Genetic Algorithms: The Crossover-Mutation Debate

A literature survey (CSS3137-B) submitted in partial fulfilment of the requirements for the Degree of Bachelor of Computer Science(Special) of the University of Colombo.

Nuwan I. Senaratna
(Index Number: 7096; Registration Number: 2002s8423)

November 15, 2005

Abstract

Crossover and mutation are two of the most important genetic operators found in genetic algorithms. There has been much debate as to which of these is practically and theoretically more effective. This literature review highlights the principal milestones of this debate. The conclusion we reach is that there is no evidence to show that either operator is better than the other, and that both operators have their own useful qualities. We also highlight some important new trends that we think might influence the debate in the future.

This literature review will be particularly useful for those who would like to theoretically study the relative roles of the crossover and mutation operators and for those who practically work with genetic algorithms and would like to learn more about how best to utilize crossover and mutation.

Categories and Subject Descriptors: A.1 [**General Literature**]: Introductory and Survey - *Literature Review*; F.1 [**Theory of Computation**]: Computation by Abstract Device - *Self-modifying machines, Genetic Algorithms, Unbounded-action devices*; G.1 [**Mathematics of Computing**]: Numerical Analysis - *Optimisation, Global optimisation, Gradient methods, Simulated annealing, Genetic Algorithms, Stochastic programming*; G.3 [**Mathematics of Computing**]: Probability and Statistics - *Markov processes*; I.2 [**Computing Methodologies**]: Artificial Intelligence - *Philosophical foundations, Applications and Expert Systems, Automatic Programming, Learning*; I.6 [**Computing Methodologies**]: Simulation and Modelling - *General*.

General Terms: Genetic Algorithms, Evolution, Crossover, Mutation, Chromosome, Gene, Allele, Locus, Genome, Genotype, Phenotype, Convergence, Fitness, Crossover Point, Schema, Defining Length, Order, Building Block, Noise

Additional Key Words and Phrases: Schema Theorem, Building Block Hypothesis, Exact Models, Implicit Parallelism, Evolutionary Algorithms, Evolutionary Strategies, Macromutation, Single-point Crossover, Two-point Crossover, Multipoint Crossover, Uniform Crossover, Adaptive Mutation, Hitchhiking, Positional Bias, Nave Evolution, Random Search, Gradient Search, Hill Climbing, Simulated Annealing, Markov Chain Model

Contents

List of Figures

List of Tables

1 Introduction

1.1 Preamble

Genetic algorithms is one of the most interesting and intriguing fields of study in computer science. They have been practically used to solve many different types of search and optimisation problems in many different fields, most of which have resisted attack from conventional solution methods. This has led to a general notion that "something" in genetic algorithms (and similar unconventional search and optimisation methods), makes them superior to conventional methods. "Crossover" and "Mutation", two of the most important algorithmic operators found in genetic algorithms, are very good examples of these "somethings".

From the earliest studies in genetic algorithms, controversy has reigned as to which of the two is superior. This survey outlines the major milestones in this marathon debate.

1.2 Overview of the Milestones

The Schema Theorem was probably the first attempt at theoretically analyzing genetic algorithms. Consequently, it formed the basis of most initial debate regarding the relative worth of crossover and mutation. Milestones 1 to 5, analyze and review how this theorem and related theoretical work influenced the Crossover-Mutation Debate.

As genetic algorithms were practically applied more widely, it became apparent that the Schema Theorem and other early work were not sufficient. Practical applications spawned a wide range of new techniques and variants on existing techniques in genetic algorithms as well as other competing methods, which behaved in ways not wholly and accurately explained by early theoretical work. In Milestones 6 to 9, we take a look at some of these new practical problems that genetic algorithms were used to solve, how genetic algorithms were adapted to solve these and how other competing algorithmic methods also worked to solve these.

In an attempt to better explain these new findings and techniques, many mathematical frameworks and models were devised. In Milestones 10 to 12, we describe three such mathematical frameworks that very strongly influenced the Crossover-Mutation Debate.

In spite of all this theoretical work, many fundamental questions about genetic algorithms, especially those pertaining to the specific "roles" of crossover and mutation, were still unanswered. A higher level look seemed called for. In Milestones 13 to 16, we look at some modern theories and "high level" ideas that have emerged regarding genetic algorithms and their working.

Following these milestones, we look at three possible conclusions to which the "Crossover-Mutation Debate" might now have reached: "win-lose", "lose-lose" and "win-win". Finally, we quote some of the important new trends that are likely to influence the debate in future.

1.3 A Note for the Reader

The reader must be familiar with basic genetic algorithms theory and associated terms. If not, he or she is recommended to read any of the many books introducing the subject, for example[16]. It is also useful if the reader is familiar with basic genetics and associated terms. For quick reference in genetics, the reader is recommended[7]. The milestones are not all independent sections that may be read separately. Most of the them are continuations of the ideas or claims put forward in the previous milestone. Hence, it is important that they are read in the given order.

2 Schema Theorem and related work

2.1 Milestone 1: The first genetic algorithms

During the early 1950s many scientists began pioneering genetics and evolution-based computation processes. John Holland is generally credited as having "invented" genetic algorithms. "Invented" seems to be a slight over-statement as several contemporary scientists such as Bremermann, Box, Bledsoe, Reed, Tooms, Friedman and Baricelli independently came up with several similar "inventions"[16]. However, it was Holland who really popularised genetic algorithms. He was also the first to study evolutionary phenomena, such as adaptation, in order to develop ways in which such phenomena might be applied to computer applications. Holland's first genetic algorithms were direct applications of his observations in natural evolution.

Although these early genetic algorithms were very simple and practically trivial, they did provide valuable insight into the workings of genetic algorithms. However, at the time there was no theoretical framework to mathematically explain these workings. This resulted in many problems:

- Most test results were explained by vague ideas inspired by intuition and gut-feeling, rather than specific knowledge.
- It was difficult to make unbiased and accurate predictions regarding runs of GAs.
- It was difficult to quantify findings in general.
- Most early claims regarding genetic algorithms were highly debatable and seldom universally accepted.

One of the most important of such "debatable" claims was Holland's claim that crossover was more important than mutation. He, initially, attempted to justify this by pointing out that genetic algorithms with high crossover probabilities and low mutation probabilities, tended to perform better. However, this evidence was of a purely empirical nature. Nor were scientists sure how to compare the figures, as it seemed a case of "comparing apples and oranges". The claim did not have credible evidence to support it. However, nor was there evidence to disclaim it. This was the beginning of the "Crossover-Mutation Debate". Computer scientists were split. Some supported mutation, while others crossover.

Irrespective of what they agreed with and what they disagreed with, most agreed on one thing. This problem was not going to be cleared up on empirical evidence. It was vital that some mathematical base for modelling, explaining and predicting genetic algorithms be developed. The Schema Theorem was probably the first of such developments.

2.2 Milestone 2: The Schema Theorem

The Schema Theorem[16] develops as follows:
If there is no crossover or mutation, the expected number of instances of schema H at time $t+1$, $E\left[m(H,t+1)\right]$ is:

$$E\left(m(H,t+1)\right) = \sum_{\forall x \in H} \frac{f(x)}{\bar{f}(t)} = \frac{\hat{u}(H,t)}{\bar{f}(t)} \cdot m(H,t) \tag{1}$$

Here, $m(H,t)$ is the number of instances of H at time t. $\hat{u}(H,t)$ is the observed average normalised fitness of H at time t. Selection assumed to be done using the roulette wheel technique, hence the expected number of offspring of an arbitrary organism x is $\frac{f(x)}{\bar{f}(t)}$ where $f(x)$ is the fitness of x and $\bar{f}(t)$is the average fitness of the population at time t.

Considering the effect of crossover (single-point crossover, as Holland initially did), the probability that H will survive crossover, $S_c(H)$, may be given as:

$$S_c(H) \geq 1 - p_c \left(\frac{d_H}{l-1} \right) \tag{2}$$

Here, p_c is the crossover probability. A mating instance of schema H is said to survive under crossover if one of its offspring also belong to schema H. d_H is the defining length of H and l is the length of each bit string in the considered population.

The inequality results because the *RHS* is a lower bound for the survival probability. This is because in the derivation we have only considered the destructive effect of crossover (that is, the possibility of the schema being destroyed). Similarly, the probability that H will survive mutation, $S_m(H)$, may be given as:

$$S_m(H) = (1 - p_m)^{o(H)} \tag{3}$$

Here, p_m is the mutation probability. $o(H)$ is the order of H.

By considering the effects of crossover and mutation on the population (substituting equations (2) and (3) in (1)), we get the following equation. This is known as the Schema Theorem.

$$E\left(m(H, t+1)\right) \geq \left(\frac{\hat{u}(H,t)}{\bar{f}(t)} \cdot m(H,t) \right) \times \left(1 - p_c \left(\frac{d_H}{l-1} \right) \right) \times \left((1-p_m)^{o(H)} \right) \tag{4}$$

Note: The Schema Theorem has been expressed in different notations. For example, Grefenstette[8] combines the effects of crossover and mutation into a single disruption effect term, and also expresses the theorem for an infinite population.

The Schema Theorem loosely implies that schema most likely to survive are short ($d(H)$ is small), are of low order ($o(H)$ is small) and have average observed fitness greater than the average fitness of the population as a whole ($\hat{u}(H,t) > f(x)$). Early exponents of this theorem including Holland himself strongly believed that crossover was the major source of power in genetic algorithms. Hence, the influence of mutation was often ignored and the theorem expressed without the $S_m(H)$ term.

Further, they developed on the fact that "short, lower order schemas" had a high chance of surviving, in such a way to imply that, such schemas recombined to form more and more optimal solutions. They described the working of a genetic algorithm as a process where short, lower order, fit schema were selected, recombined with crossover to form strings of increasingly high fitness. This idea was best expressed in, what is known as, the "Building Block Hypothesis".

2.3 Milestone 3: The Building Block Hypothesis

The "Building Block Hypothesis" was put forward by David Goldberg. It states:

> "Given any low-order, short defining length schema partitions, the genetic algorithm is expected to converge to the schema in that partition, with the highest average fitness"[6].

At its core is the idea of a building block. A **building block** is any short, low order schema with high fitness. Goldberg claimed that, "...Just as a child creates a magnificent fortress through the arrangement of simple blocks of wood, so does a genetic algorithm seek near optimal performance through the juxtaposition of short, low-order, high-performance schemas or building blocks..."[6].

Now, if we forget the concept of building blocks and consider individual organisms independently (that is, independently of the other organisms that belong to schemas they do not belong to), we could just as well state, "..... so does a genetic algorithm seek near optimal performance through the juxtaposition between individual high-performance binary strings...". So why bring in the concept of "building blocks" and schemas? The answer lay in what was known as **Implicit Parallelism**[16].

For a binary string of length l, we have 2^l possible binary strings but 3^l possible schemas. A single binary string belongs to many schemas. Hence, when we crossover just two binary strings, a much larger number of schemas are operated upon in parallel. This was considered one of the factors that contributed to the special "power" of the genetic algorithm.

Both the power of building blocks and Implicit Parallelism are direct consequences of crossover. Hence, the Building Block Hypothesis was considered as major evidence in favour of the idea that crossover was superior to mutation. However, the hypothesis had a serious flaw. It claimed that building blocks combined to form better strings. This claim might not seem unreasonable, since it involves attacking a complex problem by splitting it into partial solutions (divide and conquer), a strategy proven to be successful in many different cases and areas. However, it was still nothing but a claim. At the time there was no evidence to justify it. Hence, both the Schema Theorem and the Building Block Hypothesis were only partial victories for crossover in the "Crossover-Mutation Debate". Further evidence in favour of either side was yet to come.

2.4 Milestone 4: The Dynamic and Static Building Block Hypothesis

Grefenstette pointed out that the Schema Theorem might give the impression that schemas with higher actual fitness have a higher chance of surviving, when actually it is the schemas with higher observed fitnesses that survive[8]. The Schema Theorem calculates the expected numbers of instances of organisms from a given schema in the next generation, given observations carried out in the current generation. It shows that the schema that are more likely to survive are those with higher *observed* average fitness, not those with higher *actual* average fitness.

To accommodate this observation, the Building Block Hypothesis was restated as the "Dynamic Building Block Hypothesis" as:

> Given any low-order, short defining length schema partitions, the genetic algorithm is expected to converge to the schema in that partition, with the highest *dynamic* (observed) average fitness[8].

However, supporters of the original interpretation of the Building Block Hypothesis claimed that in most cases dynamic (observed) average fitness and static (actual or expected) average fitness were not very different. They, further restated the Building Block Hypothesis as the "Static Building Block Hypothesis" by replacing *dynamic* in the above statement, with *static*.

This second hypothesis was not widely accepted. In more work on these ideas, Grefenstette stated that the Static Building Block Hypothesis failed to express the true behaviour of genetic algorithms in two ways[8]:

- It failed to account for collateral convergence. That is, once the population begins to converge, even slightly, it is no longer possible to estimate the static average fitness of schemas using the information present in the current population.
- It failed to account for fitness variance within schemas. The effect of fitness variance within schemas is that, in populations of realistic size, the observed fitness of a schema may be arbitrarily far from the static average fitness, even in the initial population.

Further criticisms were on the way.

2.5 Milestone 5: The schemas are further criticized and defended

Among the direct criticisms of the Schema Theorem were the following.

- Fogel and Ghozeil criticized the Schema Theorem as failing to estimate the population schema proportions when fitness proportionate selection is used in problems with noise and other effects[2].

- The Schema Theorem itself showed that schema building was detrimental. An interesting case was deception [8].
- The Schema Theorem did not consider the **constructive effects** of crossover and mutation. Nor did it take into consideration the fact that the **observed fitness** of a particular schema changes as the population evolves. It also did not provide any indication as to how the population converges. Many empirical observations indicated various correlations between the convergence rates and the size of population. However, none of these were explained by the Schema Theorem.

However, these criticisms had a few flaws and there was much argument defending the Schema Theorem and related work. For example:

- Much of the evidence against the Schema Theorem were based on empirical results.
- The work of Fogel and Ghozeil was criticized as inaccurate models of the theorem, and hence their results were dismissed as insufficient. Poli and Radcliffe independently argued that this model gave a warped "over-interpreted" impression of Holland's theorem, and that when "modelled properly" the Schema Theorem performed very well in the face of noise[2].

Initially the Schema Theorem and the Building Block Hypothesis seemed firm evidence of the superiority of crossover over mutation. As we saw, however, this was short-lived, as various flaws and loopholes were found. The Schema Theorem was initially intended as a theoretical framework to explain the workings of genetic algorithms. In many respects it had served its purpose. For example, it was now possible to make unbiased and fairly accurate predictions regarding runs of genetic algorithms (for simple ones at least). However, this framework also created more problems and controversies. This had many reasons:

- The Schema Theorem made many assumptions and hence was fairly narrow (For example it considered roulette wheel selection, single point crossover and simple mutation).
- It could not be applied to many practical problems.

3 Practical Applications and Alternative Techniques

3.1 Milestone 6: New problems

The milestones discussed so far, principally deal with the theoretical aspect of early genetic algorithms. Unlike many mathematical "concepts" in computer science, from its earliest days, genetic algorithms were used to solve many practical problems. Like all evolutionary algorithms, genetic algorithms are essentially optimizing methods. They have been used to solve many different types of optimization problems, including[16]:

- Numerical optimizations (e.g. solving complex mathematical equations) and combinatorial optimizations (e.g. designing optimal network layouts and job-shop scheduling).
- Automatic programming, particularly to evolve programs for specific tasks and to adaptively design various computational structures such as sorting networks, neural networks and cellular automata.
- Prediction algorithms, particularly in weather forecasting
- Economics, in process models and strategy systems.
- Medicine, in modelling the immune system.
- Ecology, in modelling many ecological phenomena such as coevolution, symbiosis, resource flow and social system behaviour (e.g. insect colonies).
- Chemistry, in Macromolecular Structure Prediction, in Protein Docking and Small Molecule Conformational Search and Structure Prediction[10].

This emergence of new practical uses for genetic algorithms did not directly influence The "Crossover-Mutation Debate". However, indirectly, it had many implications. The new problems resulted in new strategies for solution, not only in genetic algorithms but also in competing techniques. The same problems were tackled using several different techniques. When one technique produced a solution that was similar to that produced by another, scientists looked for similarities and patterns. When one technique provided a better solution, scientists scrutinized differences between techniques. Crossover and mutation were questioned in a whole range of different ways, and like the algorithms themselves, the operators themselves were forced to evolve.

3.2 Milestone 7: Competing algorithmic rivals

Competing algorithmic methods can be viewed from two aspects with respect to genetic algorithms. Firstly, they can be viewed as independent methods that have their own advantages and disadvantages over genetic algorithms. Secondly, they can also be viewed as siblings or cousins of genetic algorithms with similar mechanisms, only implemented to different extents.

Let's compare some of these methods, relative to genetic algorithms. (It is assumed that the reader is familiar with these methods. For explanations refer [16].)

Method	**Known Population Size**	**Selection**	**Crossover**	**Mutation**
Random Search	Entire population[a]	None	None	None
Gradient Search	1[b]	Trivial[c]	None	Yes[d]
Hill Climbing	1[b]	Trivial[c]	None	Yes[d]
Simulated Annealing	1[b]	Trivial[c]	None	Yes[e]
Evolution Strategies	Variable[f]	Yes	None	Yes[g]

Table 1: Competing methods compared

[a] Since the optimum could be any member of the population, all the members of the population must be known. A Random Search is like a genetic algorithm where the initial population contains all possible organisms. Hence, it is of the maximum possible size (For example if the organisms are encoded into binary strings of length 20, the population size will be over 1,000,000).

[b] A single binary string is chosen at random as the "current binary string"

[c] Selection is trivial since the population consist of one individual.

[d] Each bit in the "current binary string" is systematically mutated from left to right.

[e] Like Gradient Search and Hill Climbing, the "current binary string" is systematically mutated from left to right. However, the use of a downhill probability operator increases the effect of mutation.

[f] The nature of the population in Evolution Strategies (and also selection) is as in genetic algorithms

[g] Since mutation is implemented in Evolution Strategies, just as it is in genetic algorithms.

Something that all these methods have in common is that they do not utilize crossover. This in it self might hint that mutation had something superior to crossover. However, the real evidence came from the analysis of the performance of evolution strategies.

3.3 Milestone 6: Evolution Strategies and the "Genetic Algorithms are different" Debate

Genetic algorithms are a specialization of what are generally known as evolutionary algorithms[2]. Evolution Strategies and genetic programming are the two other principal forms of evolutionary algorithms. Genetic programming consists of evolving a program body, where every part of the program is a characteristic of some gene.

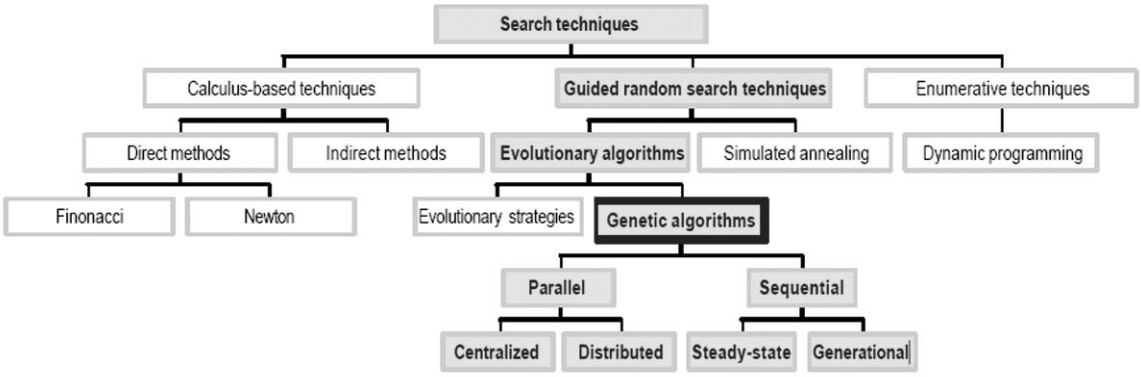

Figure 1: Family Tree of Search Techniques (Obtained from[2])

Evolution strategies are optimising methods very similar to genetic algorithms. Evolution strategies differ from genetic algorithms in two aspects. Evolution strategies use real values to represent gene organisms instead of binary strings. However, more importantly, evolution strategies use only mutation. There was much evidence to show that evolution strategies were superior to genetic algorithms.

- Evolution strategies are usually quicker and more efficient than genetic algorithms[25].
- Rechenberg, who pioneered evolution strategies, pointed out that "mutation" was a key evolutionary operator and should, hence, be pursued in other types of evolutionary algorithms.
- Many studies in evolution strategies also showed that mutation was a more sophisticated operator than crossover in that its performance parameters were dependent on the nature of the population.
- Sophisticated versions of evolution strategies, with adaptive mutation rates, proved several orders of magnitude more efficient in many function optimisation tasks.

These observations not only highlighted the role of mutation as an operator, but also showed that mutation had been a much ignored and misunderstood concept and that much of its potential remained untapped.

This sparked a fresh round of the "Crossover-Mutation Debate". Proponents of mutation argued that since mutation was the dominant operator in evolution strategies and that since this method seemed superior to genetic algorithms (where crossover was "favoured" as a supposedly dominant operator), mutation should be used much more in genetic algorithms. Opponents of mutation argued that, although mutation might be suitable for evolution strategies, genetic algorithms were an "inherently different mechanism" and that analogy was not valid.

This statement that genetic algorithms were an "inherently different mechanism" initiated the "Genetic Algorithms are different" Debate. The interesting fact was that, although mutation seemed, in the case of many other optimisation techniques, the dominant operator, the genetic algorithm was the only technique where crossover might be realistically implemented. Hence although evaluating other methods provided much insight into crossover and mutation, it seemed neither fair nor valid to compare them on these grounds. Crossover was individual to genetic algorithms and any validation of it must be done in the context of genetic algorithms - not other methods. Hence, like the "Crossover-Mutation Debate", the "Genetic Algorithms are different Debate" was widely open. Both debates were argued in parallel, sometimes colliding and intermixing.

The first blow to the "Genetic Algorithms are different" idea came from the Fogels.

3.4 Milestone 8: "Crossover provides no significant benefit"

David Fogel who continued on the work of L. Fogel, came up with some very interesting claims[5]:

- Adaptation does not require sophisticated evolutionary operators to occur successfully. Evolution towards optimality is inevitable. This implies that operators play a minor role in genetic algorithms. It is the actual process itself that delivers the goods.
- The link between parent and offspring generations must sufficiently be preserved in order to ensure that positive advances in adaptations (successful moves towards the optimal organism) are preserved (that is, we must make sure that child organisms inherit their parents fit qualities). Crossover does not always maintain this "genealogical link".
- Crossover is a generalization of several mutations performed at once. Hence, it could have no significant advantage over mutation, since it is a form of mutation itself. Conversely, mutation is a more specific and specialized version of "macromutation" and hence could have relative advantages. Consequently, Fogel claimed that mutation results in more efficient searches.

However, the most interesting and controversial of Fogel's claims is probably the link he made between genetic algorithms and other evolutionary algorithms. He said that genetic algorithms were developed by a group of people who considered genetic algorithms *fundamentally different* from other evolutionary algorithms. He claimed that all evolutionary algorithms are essentially optimization techniques and that all optimization techniques essentially climb hills, that is, they traverse the population landscape in whatever direction that might lead to a peak or optima. He showed that the only way genetic algorithms were different from other evolutionary algorithms was in its own particular operators. These (mutation, crossover, etc.) were all instances of some common "ancestor" operator and were fundamentally the same. Similarly, all evolutionary algorithms are instances of some common "ancestor" optimization mechanism that inherit most of their better capabilities from that ancestor mechanism, and hence, were fundamentally the same.

3.5 Milestone 9: New techniques

Crossover and mutation had so far been viewed as operators. Under this view, the performance of crossover and mutation had been analysed by analysing specific crossover and mutation operators. Conversely, they can be viewed as **mechanisms** - mechanisms that are implemented using operators. This is a more "high level" view and leads to a more broader understanding of crossover and mutation and their relative performance. For example, so far, we have viewed crossover in terms of the "single-point crossover" operator. However, crossover can be extended to contain other possible "crossover" operators. The emergence of such "New Operators" drastically changed views on genetic operators.

Although this new high-level view might be given as a reason for the emergence of new operators, the real reason why new operators emerged was that the conventional operators had inherent defects. For example[16]:

- Crossover was generally credited as having the ability of generating new higher order schemas with high fitness from lower order schemas. However, the generation of new higher order dependent highly on the nature of their potential component lower order schemas (**positional bias**).
- Short, lower order schemas had a high probability of surviving, even if certain members of that schema were unfit, as long as other members were fit. There was a tendency for unfit gene patterns to get "hitched" onto fit patterns and hence survive (**hitchhiking**).
- Single-point crossover treats different positions with bias (**preferential treatment**).
- It was noted that in many practical problems, it was advantageous to select certain points known as "hot spots" for crossover. However, single-point crossover did not make allowances for this.

- In order to keep building blocks intact, extra operations such as inversion and reordering [1]were needed.

Among the new forms that emerged were two-point crossover, multipoint crossover and uniform crossover.

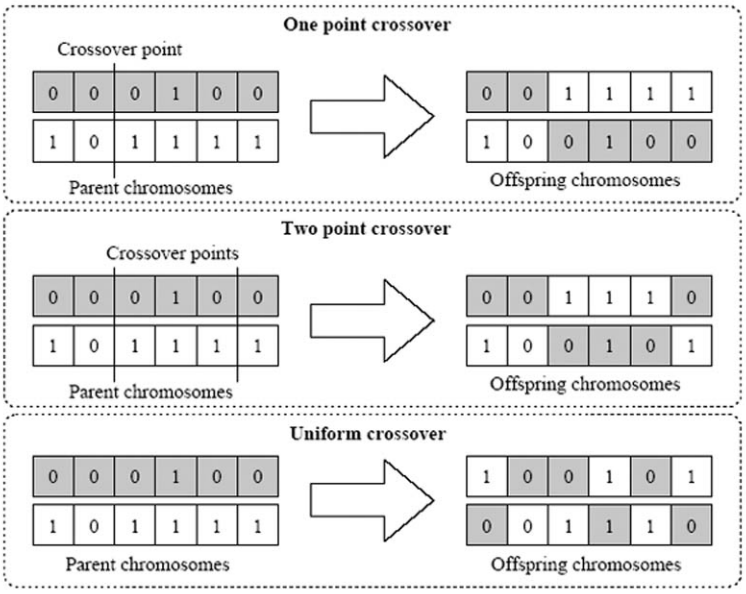

Figure 2: Forms of Crossover (Obtained from[20])

Of these, uniform crossover was the most interesting and promising. In[13] De Jong and Spears list three comparatively positive virtues of uniform crossover:

- The disruption of hyperplane sampling under uniform crossover does not depend on the defining length of the hyperplanes. This reduces the possibility of representation effects, since there is no defining length bias.
- The disruption potential is easily controlled via a single parameter (*see* Disruption Theory). This suggests the need for only one crossover form (uniform crossover), which is adapted to different situations by adjusting this parameter.
- When a disruption does occur, uniform crossover results in a minimally biased exploration of the space being searched.

Like crossover, new mutation techniques also emerged. Since application of mutation was much simpler, the emphasis was on how mutation parameters should be best utilized. Many researchers noticed that the optimal mutation rate was much dependent on the state of the population, especially its variance. Hence, many new techniques for adapting mutation rates during the execution of the genetic algorithm were developed.

There was also much emphasis on running generic algorithms *without* crossover (**Naive Evolution**). In fact there are many practical problems that were better solved this way. For example, in[24], Spears and Anand say that for neural network modules and their control circuits, genetic algorithms without crossover, perform much better than those with crossover. It is interesting to note that there are many cases in nature where complex organisms have evolved without any crossover whatsoever (e.g. Bdelloid rotifers). Indeed, biologists consider mutation, not crossover, as the main source of “raw material” in evolution[1].

The emergence of these new crossover and mutation operators, not only drastically expanded general views on operators, they also complicated the "Crossover-Mutation Debate". From one side, continuous generalizations as to what exactly was meant by a crossover or mutation "mechanism" resulted in higher and higher level views. From the other side, more and more involved operators resulted in an urgent need for more involved frameworks to analyse and compare them. It was becoming very apparent that without some rigorous mathematical work the "Crossover-Mutation Debate" as well as genetic algorithms in general would stray in all directions.

4 Mathematical Frameworks and Models

4.1 Milestone 10: Exact mathematical models

As we saw, a rigorous mathematical framework was the need of the day. The Schema Theorem was a start, but it was hardly sufficient. For example, it did not provide a "long term" view of the performance of the algorithms. By long term, the author means, a view that provides insight into the working of a genetic algorithm throughout several generations (possibly throughout its entire execution).

In 1991, Vose and Liepins solved a genetic algorithm with an **Exact Model**[26]. An Exact Model of a genetic algorithm captures every detail of it in the form of mathematical operators. For simplicity they considered a genetic algorithm with an infinite population (hence, sampling error was zero). Each of the possible binary string representations an organism can take is represented by the integer that the binary string encodes (e.g. 0110 is represented by integer 6).

The population at time (generation) t, is represented in column matrix, $\vec{p}(t)$:

$$\vec{p}(t) = \begin{pmatrix} p_0(t) \\ p_1(t) \\ ... \\ p_N(t) \end{pmatrix} \tag{5}$$

Here *0,1...N* are the integers that represent the binary strings. If we encode the genetic algorithm with binary strings of length l, then $N = 2^l - 1$. $p_i(t)$ is the proportion of organisms in the population coded as the binary string represented by integer i at time (generation) t.

The normalized fitnesses are similarly represented in column matrix $\vec{s}(t)$:

$$\vec{s}(t) = \begin{pmatrix} s_0(t) \\ s_1(t) \\ ... \\ s_N(t) \end{pmatrix} = \frac{F \cdot p_i(t)}{\sum_{j=0}^{N} F_{j,j} \cdot p_i(t)} \tag{6}$$

Here, $F = \begin{pmatrix} f_0(t) & 0 & ... & 0 \\ 0 & f_1(t) & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & f_N(t) \end{pmatrix}$. $f_i(t)$ is the unnormalized fitness of the binary string represented by integer i. Note that F can be considered as an operator that represents selection.

The goal of an Exact Model was to come up with an operator G, with the property $\vec{s}(t+1) = G \cdot \vec{s}(t)$. If only selection was considered then, since $E[\vec{s}(t+1)] = F \cdot \vec{s}(t)$ and an infinite population is considered, $\vec{s}(t+1) = F \cdot \vec{s}(t)$ and simply $G = F$. The effect of crossover and mutation was mimicked in the mathematical structure $\mathcal{M}$ which consisted of elements r, such that,

$$E[p_k(t+1)] = \sum_{\forall i,j} s_i(t) \cdot s_j(i) \cdot r_{i,j}(k) \tag{7}$$

.

The combined effect of $\mathcal{M}$ and F was such that $G = \mathcal{M} \circ \mathcal{F}$. Vose and Liepins showed that,

$$r_{i,j}(0) = \frac{1-p_m}{l} \cdot \left[\eta^{d(i)} \cdot \left(1 - p_c + \frac{p_c}{l-1} \sum_{c=1}^{l-1} \eta^{-\Delta_{i,j,c}} \right) + \eta^{d(j)} \cdot \left(1 - p_c + \frac{p_c}{l-1} \sum_{c=1}^{l-1} \eta^{-\Delta_{i,j,c}} \right) \right] \quad (8)$$

Here $\eta = \frac{p_m}{1-p_m}$, $d(X)$ is the number of one bits in the binary string that represents integer x and $\Delta_{i,j,c} = d(\text{“rightmost } c \text{ bits of } i\text{”}) - d(\text{“rightmost } c \text{ bits of } j\text{”})$.

By mathematically manipulating equation (8) , they found expressions for the other $r_{i,j}(k)$'s.

This model resulted in some very interesting implications. It showed that if there was no selection, and only crossover and mutation (that is, $G = \mathcal{M}$), then the entire population will tend to a uniformly varied random population. Hence, they recognized F as a principally “focussing” operator and $\mathcal{M}$ as a dispersing operator. This had two very important effects on the “Crossover-Mutation Debate”[26]:

- It seemed to confirm Fogel's claim that operators played an insignificant role in genetic algorithms, and that mutation and crossover were not much different.
- Secondly, it pointed to the fact that there were several, sometimes conflicting, mechanisms simultaneously going on in genetic algorithms, of which both crossover and mutation were simultaneously a part of.

It was more than apparent that this mathematical model was one of the most significant and important milestones in the “Crossover-Mutation Debate”. However, it also had a serious flaw. It, among other things, assumed an *infinite population*. In practical situations, practical constraints (processing, memory, etc.) make sure that the size of the population is indeed “very finite”. An Exact Model that did not assume an infinite population was needed. Nix and Vose formulated just that using Markov Chains.

4.2 Milestone 11: Markov Chain Model

A **Markov Chain** is a stochastic process where the probability that the process is in a particular state at a given time t, depends only on the state of the process at a time $t-1$. In a genetic algorithm, the state of a population is dependent only on its parent population. Hence, Markov Chains proved ideal for modelling genetic algorithms.

Nix and Vose used Markov Chains to model very simple genetic algorithms[17]. All possible populations were represented as vectors P_i. The probability of a population P_i would, in the course of n generations (selection and recombination steps), be transformed in to population P_j was represented in a matrix with elements $Q_{i,j}$ where:

$$Q_{i,j} = n! \prod_{y=0}^{2^l-1} \frac{\left(\mu \cdot \left(\frac{F\vec{\phi}_i}{|F\vec{\phi}_i|} \right) \right)}{Z_{y,j}!} \quad (9)$$

Here F and $\mathcal{M}$ are as defined in the previous section. $\vec{\phi}_i$ is a vector such that the y^{th} element in $\vec{\phi}_i$ is the number of occurrences of string y in population P_i. Z is a giant matrix representing all possible populations. The i^{th} column Z is $\vec{\phi}_i$ and $Z_{y,i}$ is equivalent to the y^{th} element in $\vec{\phi}_i$.

Not only did Exact Models provide an exact “track” of the genetic algorithm performance, they were also ideal for comparison purposes. Nix and Vose showed that as the population size increased, this model converges to the infinite population model. They also used these “exact” findings to build a graphical landscape representation of the life of a genetic algorithm. This showed which populations were more reachable and which were not. They proposed a very interesting conjecture that stated that although the short-term behaviour of a population is strongly dependent of the initial population, the long-term

behaviour of the population is independent of this and strongly dependent on the landscape of which it is a part of[17].

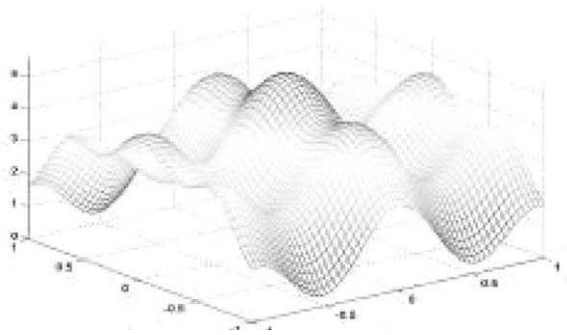

Figure 3: 3D representation of a population landscape (Obtained from[2])

The drawback of this model was that it could be used only for very simple genetic algorithms. Practically, it was very limited. However, it allowed a very detailed and precise analysis of some fundamental mechanisms. For example:

- It provided a very detailed picture of how the optimal mutation rates depend on the size of the population.
- It showed how the relative importance of crossover and mutation rates interrelate.
- It confirmed some and rejected others of the many of the empirical claims made with regard to crossover and mutation

In a sense, it elevated the view computer scientists took in analysing genetic algorithms. Hence, in spite of its obvious limitation, the Markov Chain Model was a very important milestone in the "Crossover-Mutation Debate".

4.3 Milestone 12: "No Free Lunch" Theorems

The two mathematical models discussed so far did indeed provide precise and detailed descriptions of genetic algorithms. However, their complex and constrained nature made them difficult to analyse. De Jong and Spears came up with a less precise but more analysable characterization framework for genetic algorithms. This led to many insights, including to what became known as the "No Free Lunch" Theorems[25].

In this review, we only skim the gist of the mathematical derivations. For details of the workings, the reader must refer the original text.

4.3.1 Framework applied to General Operators

First, operators were considered generally as processes to which two parameters are input (parent organisms) and two are output(child organisms). For any such operator, for any schema H_k (where k is the order of the schema), the random variable A_k represents the number of organisms that belong to H_kc in the input parameters. Similarly, B_k for the output parameters.

Then[25], they showed that the **survivability** of H_k (that is, given that at least one of the parents belonged to H_k, the expected number of children belonging to H_k) $E_s[B_k]$,

$$E_s[B_k] = P_s(H_k) + P(B_k = 2|A_k = 1 \vee 2) \tag{10}$$

Here, $P_s(H_k)$ is the probability of survival or $P(B_k = 1 \vee 2|A_k = 1 \vee 2)$. $P(B_k = 2|A_k = 1 \vee 2)$ is in the usual probability notation.

Note that *survival* was the complement of *disruption*. They also showed[25] that the number of children belonging to H_k expected to be constructed by such an operator $E_c[B_k]$,

$$E_c[B_k] = P_c(H_k) + \frac{1}{2^k - 2} \sum_{S=1}^{2^k-2} P(B_k = 2|S) \tag{11}$$

Here, we consider the construction of H_k from lower order schemas H_m and H_n of orders m and n respectively. There is no overlapping hence, $k = m + n$. There are 2^k ways of the required k alleles being supplied. Any such way is referred to as a **situation** and is denoted by $S(0 \leq S < 2^k)$. $P_c(H_k)$ is the probability of constructing a member of H_k averaged over all such S and equals $\frac{1}{2^k-2} \sum_{S=1}^{2^k-2} P_c(H_k|S)$. $P_c(H_k|S)$ is $P(B_k = 1 \vee 2|S)$ where "$|S$" denotes "given that the operation belongs to situation S".

Finally, they combined the effects of survival and construction to give $E_{c,s}[B_k]$,

$$E_{c,s}[B_k] = P_{c,s}(H_k) + \frac{1}{2^k} \sum_{S=0}^{2^k-1} P(B_k = 2|S) \tag{12}$$

Here, $P_{c,s}(H_k)$ equals $\frac{1}{2^k}[2P_s(H_k) + (2^k - 2)P_c(H_k)]$.

This general framework was then applied to individual operators.

4.3.2 Framework applied to Crossover

For crossover $E_s[B_k]$, $E_c[B_k]$ and $E_{c,s}[B_k]$ were respectively:

$$E_s[B_k] = P_s(H_k) + P_{eq}^k \tag{13}$$

$$E_c[B_k] = P_c(H_k) + P_{eq}^k \tag{14}$$

$$E_{c,s}[B_k] = \frac{1}{2^k}[2P_s(H_k) + (2^k - 2)P_c(H_k)] + P_{eq}^k \tag{15}$$

Here,P_{eq} is a measure of the probability that both parents have the same allele at a particular defining position. It will be explained further under *Disruption Theory*. $P_s(H_k)$ is as in the general case.

$E_{c,s}[B_k]$ was plotted for various types of crossover; the result was surprising. Irrespective of what the crossover operator was, $E_{c,s}[B_k]$ was only dependent on H_k and the convergence of the population, *not* on the crossover operator. Modifying the operator to increase survivability results in a reduction in constructability (and vice versa). Or as De Jong and Spears put it:"...there appears to be no free lunch with recombination."

4.3.3 Framework applied to Mutation

Similar derivations were made for mutation. For mutation, $E_s[B_k]$, $E_c[B_k]$ and $E_{c,s}[B_k]$ were respectively:

$$E_s[B_k] = \sum_{\forall Q} P(Q)[(1 - p_m)^k + p_m^{|Q|}(1 - p_m)^{k-|Q|}] \tag{16}$$

$$E_c[B_k] = \sum_{\forall Q} \sum_{\forall R} P(Q \wedge R)[(1-p_m)^k + p_m^{|R|}(1-p_m)^{k-|R|}] + (1-p_m)^k + p_m^{|Q|}(1-p_m)^{k-|Q|}] \quad (17)$$

Here, p_m is the mutation probability. Q is the random variable describing the set of alleles in the second parent that does not match H_k. $P(Q)$ equals $(1-P_{eq})^{|Q|}.P_{eq}^{(k-|Q|)}$. R is the random variable describing the set of alleles in the first parent that does not match H_k. $P(Q \wedge R)$ equals $(1-P_{eq})^{(|Q|+|R|)}.P_{eq}^{(k-|Q|-|R|)}$

Unlike crossover, for mutation, operators that resulted in more disruption (that is, less survival or low values of $E_s[B_k]$) also resulted in less construction. Hence, no "No Free Lunch" Theorem existed for mutation and therefore a $E_{c,s}[B_k]$ had little meaning for mutation.

These "No Free Lunch" Theorems and the framework they were based on helped characterise crossover and mutation as having specific roles. The "Crossover-Mutation Debate" could now be argued at a much higher level.

5 A High level look

5.1 Milestone 13: Analytical reflection

With the advent of more precise mathematical frameworks to explain the behaviour of genetic algorithms, the theoretical void that was felt earlier was now filling. So much so that computer scientists now began to look at them from a higher level. There was much reflection and quasi-philosophical questions were starting to be asked. For example:

- Were crossover and mutation actually different? If so, what were the real differences between crossover and mutation? Which of these were observable?
- What did scientists mean when they said that mutation was superior to crossover or vice versa? What were the indicators of superiority?
- What "role" did these operators play in a genetic algorithm?

Some **differences** had already been observed:

- Crossover exploited existing genetic material, while mutation explored for newer material.
- Crossover proved less and less effective as the similarities between organisms increased as the population converged, while mutation became more and more necessary as variation in the population reduced.
- Crossover tended to preserve, while mutation seemed to destroy.

But did any of these observations make one operator **superior**? Genetic algorithms were, from a practical point of view, essentially search and optimization mechanisms. Hence, an operator could be considered superior if algorithms that chiefly utilize it are better than those that use others.
To better understand what is meant by "algorithms that chiefly utilize" scientists needed to better understand the **roles** played by operators in genetic algorithms[22].

Many roles were put forward. However, all these chiefly boiled down to two categories[22]:

- The Constructive role
- The Disruptive role

5.2 Milestone 14: Disruption Theory

Earlier we computed the probability that a given schema H will survive crossover. Now, suppose that crossover plays a purely disruptive role. Then, following along the lines of the derivation of the Schema Theorem, the exact probability that a given individual in the schema H will survive single-point crossover and uniform crossover, $P_c(H)$ is given by:

Crossover Form	$P_c(H)$ **Expression**
Single-Point	$1 - p_c \left(\frac{d_H}{l-1}\right)$
Uniform	$1 - p_c \cdot o(H)$

Table 2: $P_c(H)$ for various forms of crossover

Both forms of crossover here have certain aspects in common. Disruption can be controlled by manipulating the crossover. However, they also differ in certain respects. Unlike, single-point crossover where disruption is dependent on the defining length of the schema and the length of the entire binary string, disruption in uniform crossover is dependent, not on these factors, but on the order of the schema[22]. Hence, uniform crossover is not biased towards preserving schemas with short defining lengths. This, to an extent, undermines the "Building Block Hypothesis" and hence the power of crossover.

Another interesting fact about disruption due to crossover that is not directly revealed in the above equations is the fact that the various disruption rates vary as the population **converges**. Hence, in a sense disruption rates are also time dependent. As stated before, the Schema Theorem gives a "worst case" prediction for organism numbers. For example, in the case of crossover, it assumes that when a chromosome is subjected to crossover, all the bits that are crossed over are replaced by the inverted forms of those same bits. Initially, when the population is fairly random, this might be possible, and indeed common. However, as the population converges this becomes less and less likely, since organisms become more and more alike to one another. In order to understand this better, a quantity $p_{eq}(d)$ is defined. The $p_{eq}(d)$ of two schemas is the probability that they contain the same allele (bit values) at the d^{th} locus. More generally, p_{eq} is the average of all the $p_{eq}(d)$. Initially, since the population is random, the value of p_{eq} is around 0.5. However, as the population converges the value gets closer and closer to one. As this happens, the disruptive effect of crossover becomes less and less. When $p_{eq} = 1$, crossover has no disruptive influence at all[22].

With this total lack of disruption, crossover resulted in producing clones (offspring identical to their parents) and was hence redundant or unproductive. On the other hand, if crossover operators were altered to be more disruptive, then fit organisms were in danger of being destroyed. Hence, there was a need to increase **crossover productivity** without overly increasing disruption - as De Jong and Spears put it in [12], "to have the best of both worlds".

The way the behaviour of crossover operators depends on the **population size** also changes from operator to operator. This is also known to influence disruptive effects. Small populations also suffered from premature convergence more acutely than larger populations. However, De Jong and Spears argue that this is a different phenomenon to crossover productivity, and is due to the fact that small population sizes result in low information capacity, resulting in inaccurate selection sampling[12].

Now, let us similarly consider mutation. The probability that a given individual in a schema H will survive mutation, $P_m(H)$ is given by:

$$P_m(H) = (1 - p_m)^{o(H)} \tag{18}$$

Unlike in the case of crossover, this rate is independent of the nature of the population and its convergence. Hence, while disruption due to crossover declines as the population converges, disruption due to mutation remains constant (as long as p_m is held constant). Conversely, by varying p_m as a function of

$p_{eq}(d)$ we can produce a version of mutation that produces the exact disruption as crossover would over time.

This Disruption Theory swung the "Crossover-Mutation Debate" in favour of mutation. It showed two things:

- Mutation is a more flexible and manipulatable disruption operator.
- It could also be used to mimic crossover.

So was there any need for crossover? Couldn't we just get rid of crossover? Not quite. Anyway, the Disruption Theory was only part of the story. The Construction Theory had its own tale to tell.

5.3 Milestone 15: Construction Theory

The Schema Theorem essentially highlights the disruptive qualities of genetic operators. This viewpoint was not confined to this theorem. In most early theoretical works, all operators in general were considered initiators of disruption. However, later computer scientists began considering the possible "constructive" effects of operators. Instead of considering the possibilities of individuals within schemas being destroyed, they began to consider the possibility of higher order organisms being constructed from individuals within lower orders[22]. A search for a "Construction Theory" was also helped by the fact that analyzing disruptive effects alone was not sufficient for selecting suitable genetic operators for applications[12].

Let us initially consider mutation. Suppose we want to find the probability that an individual belonging to some schema is formed by the mutation of an individual of another schema. Let m be the number of alleles in the two schemas that match, and n be the number of alleles that don't match. Then, along the lines of Spears' framework discussed earlier the probability Q_m is:

$$Q_m = (1 - p_m)^m \cdot (p_m)^n \tag{19}$$

Now, let us consider crossover. Suppose we want to find the probability that an individual belonging to some schema is formed from a crossover operation on some other individual. Now, suppose that this latter individual has only some of the correct alleles required to form the former's schema. Then, the third individual that participates in the crossover must have the missing correct alleles. Using mathematical methods similar but more complex to the ones used previously, we can find the required probabilities[22].

Using such mathematical expressions, computer scientists were able to compare the relative constructive effects of crossover and mutation. The results were very interesting:

- The constructive power of mutation was symmetrical relative to p_m. Hence, mutation rate 0.5 yields the highest construction ability. Higher mutation rates retarded this ability[22]
- Earlier in the existence of the population when the population had a high level of diversity, crossover had a constructive ability much higher than mutation. Indeed, under these high diversity conditions, mutation was not at all able to achieve the levels of construction crossover achieved. Empirical results showed that crossover maintained this superior ability until the population was around 70% converged.
- This superiority of crossover was more prominent in the case of algorithms with longer binary string representations[22].
- Mutation could not simultaneously have a high construction rate and a low disruption rate. In other words, whenever parameters were formulated to encourage construction, simultaneously disruption would also be encouraged, and hence whatever was constructed had a high probability of being simultaneously destroyed.

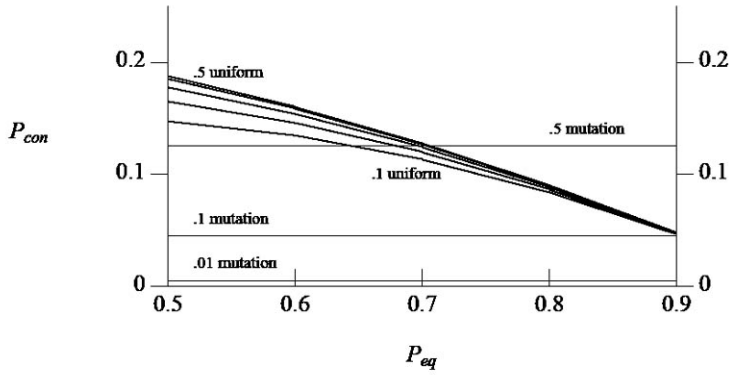

Figure 4: Construction levels achieved by crossover and mutation (Obtained from[22])

- Conversely, crossover had the ability to simultaneously achieve construction and survival (that is, make sure that disruption was low)[22].

Just like the Disruption Theory swung the "Crossover-Mutation Debate" in favour of mutation, the Construction Theory strongly supports crossover. This seemed a long awaited theoretical justification of Holland's belief that at the core of genetic algorithms was the ability for low order schema to "construct" higher order schema. Mutation was incapable of this role. Indeed, results indicated that mutation was powerless to achieve anything without some help from crossover.

5.4 Milestone 16: A High level look

So far we have viewed mutation and crossover from the point of view of what they achieve at a low level; that is, at the level of the chromosomes (of binary strings) they operate on. We have observed how they "disrupt" the nature of the strings and how they "construct" new strings from other existing strings. However, these strings are merely the low level mechanism by which the algorithms strive to achieve a high level result. What we really need to understand is how these operators perform at a high level.

What exactly do we mean by "high level"? At a high level the genetic algorithm works by traversing all possible solutions. This traversal is not done in an *ad hoc* way, but by following a certain pattern as dictated by the algorithm operators. A "high level look" at crossover and mutation may be described as "observing" the **patterns of traversal** that these operators initiate. Two important high level "patterns" are exploration and exploitation[22].

As we showed before, crossover essentially exploits inherent possibilities in the population. Mutation creates random diversity in the population by exploiting yet not-possessed genetic material. Hence, at a high level we see two roles: Exploitation and Exploration. Crossover essentially performs an exploitive role, while mutation performs an explorative role[22].

This high level point of view leads to some very interesting questions:

- Firstly, although crossover essentially performs an exploitive role and mutation an explorative role, each operator is capable of performing the other's role. What is the relative importance of these roles?
- Secondly, what is the balance between these roles that must be achieved? Many computer scientists were of the view that this aspect of genetic algorithms was generally neglected. For example, Alkatan and Abdelwahab state that most of the efforts to improve crossover operator for genetic

> programming have focused on the preservation of building blocks and that these fail to achieve the balance between the disruption and construction force of crossover [14]. Conversely, many problems arise when attempting to achieve this balance. For example, if the environment in which the organisms reside is prone to change, better solutions may tend to newly appear from time to time. In such cases, an algorithm that is in the process of exploiting an already recognized potential optimum, might have to change roles and explore the landscape for newly emerged optima. This "balance" has been often quantified in terms of the ratio of the crossover and mutation parameter values. There have been many attempts at finding such an optimal ratio.

"Exploitation and Exploration" is only one classification of operator roles. Other high level classifications also exists. For example, Sastry and Goldberg have analyzed crossover in its role as an "innovator", a "mixer" and a "schema disrupter", considering these roles both individually and composedly[20].

Some have taken the idea of a high level to philosophical extremes, drawing analogies with human lives. For example Lobo says:

> If one observes the way most people live, one has to conclude that, with some exceptions once in a while, what people usually do in life is mostly small variations of what they have done in the past. In many ways, what we do today is not much different from what we did yesterday, and that sounds a lot like a mutation kind of exploration way of life. Likewise, one could speculate that crossover corresponds to the big decisions that one makes in life. Tracing back my own life, I can clearly recognise some landmarks that were very important for my development. In many ways, those were decisions that required a bit more thought, were more unusual, more challenging, more risky, but were also the ones that yielded the highest payoff on the long run[15].

6 The Result

As we come to the end of this literature survey, one wonders where exactly the "Crossover-Mutation Debate" has led itself to and what its result is (if such a result exists!). There are several possible conclusions.

- **The win-lose result:** The crossover and mutation debate throughout had devotees who firmly believed that either one or the other operator was definitely superior. This is indeed a reasonable conclusion, because after all , for specific cases at least, it is possible to quote countless cases where one operator or the other is superior. However, these are indeed specialized and this conclusion is not valid, for more general cases.
- **The lose-lose result:** Others argued that neither crossover nor mutation had any significant effect, and that, in a sense, the debate had no winners - only two losers. This conclusion, though seems realistic, is pessimistic. It undermines much of the promising work that is in progress regarding crossover and mutation. Much of the future of genetic algorithms rests on new work on crossover and mutation. Hence, acceptance of this conclusion is, in a way, rejection of genetic algorithms as a whole.
- **The win-win result:** The most suitable result is a "win-win" result. Both crossover and mutation have many positive aspects, and though neither is perfect, both are necessary. "Winning-winning" is a better conclusion, since neither side has actually won yet, while new trends are improving both.

So it seemed that the debate is not yet over, with both teams in potential winning positions.

7 New Trends

In fact, it seems that the "Crossover-Mutation Debate" has only just begun. Genetic algorithms are still largely unexplored and not very well understood. There is much work in progress and many new areas are only just being explored. Many of these are likely to influence the "Crossover-Mutation Debate" and serve as future milestones. Some important new trends are listed.

7.1 Specialized work into the concept of fitness

In nature, a fit organism is one which survives. In practice, genetic algorithms are used to solve specific problems, and the optimally fit organism must be that which represents the optimal solution. Very often, genetic algorithms undesirably mimic nature in such a way that the organism that survives is not always the optimal one. This has resulted in much of the misleading empirical evidence that has fuelled much of the "Crossover-Mutation Debate". If the debate is be argued with better integrity, it is vital that the concept of fitness is better understood. Much work in this area is already under way.

7.2 New methods of implementing operators

Earlier, we saw how the emergence of new crossover and mutation influenced the "Crossover-Mutation Debate". New techniques for implementing operators continue to be found and investigated. In[18], Radcliffe generalises schema as **Formae**, to help exploit parallelisms better. He introduces a random, respectful recombination operator, which he claims, in addition to being useful in its own right, will also help build more complex operators.

Also, considerable work is going into how crossover and mutation might be implemented as a unified operator. For example, in[21], Sastry and Goldberg investigate a selectomutative genetic algorithm. They use a probabilistic model building process that enables the mutation operator to automatically identify key building blocks of the search problem.

In[27], Watson and Pollack represent building blocks previously identified as good building blocks, as independent, variable length individuals. The crossover operator is applied to these individuals (and not to quasi-random bit string sets). This ensures producing offspring that have the good characteristics of both parents and prevents the propagation of garbage genes (hitchhikers, etc.).

7.3 New methods of adapting parameters

As we saw both crossover and mutation perform very differently when they are slightly varied. New variants have not only radically influenced the "Crossover-Mutation Debate", they have resulted in many improvements in genetic algorithms in general. Varying the values operator parameters take has been known to drastically vary their performance. There have been several new techniques for manipulating operator parameter values and many of these are likely to influence crossover and mutation.

In[23], Spears implements a genetic algorithm such that it can choose between two-point and uniform crossover at various times, adaptively. The marked improvement in performance observed prompts Spears to further suggest the adaptive use of more operators.

Also, a chief reason for inconclusiveness in the "Crossover-Mutation Debate" was the fact that operators were unable to perform independently of the problem at hand. Many new methods have proved otherwise. For example, the use of "Labelled mutation parameters" (that is, where each gene has a self adapting label that assigns it its own mutation rate) has resulted in algorithm implementations that are independent of the problem landscape, and hence work equally well for a wide range of problems[9].

Similarly, Draidi and Kharma in[3] investigate *Parameterless* Genetic Algorithms (genetic algorithms that do not need manual tuning of their parameters). They analyse three categories of such: Deterministic, Adaptive and Self-Adaptive. Their conclusions are interesting: Although they find that different problem-situations require different parameterless methods, they find that, overall, it is the simplest methods that are most successful generally.

7.4 Influences from Genetics and Ecology

Geneticists and Ecologists have developed many theories to explain the roles of crossover and mutation. These have not yet been completely applied to genetic algorithms. Also, many variations of mutation and crossover studied in genetics and ecology might be applied to genetic algorithms.

Computer scientists are also incorporating aspects of natural occurrences into genetic algorithms. For example, in natural organisms we see "generation gaps" within populations. This concept of overlapping generations has been used in genetic algorithms known as "steady state" genetic algorithms.

In[11] De Jong and Sarma study generation gaps. They study the effect of these on the variance of populations, methods of deleting offspring in order to control variance and various factors that compound their theories. They develop their arguments using mutation-crossover-free genetic algorithms and then go on to discuss implications in real genetic algorithms.

7.5 Statistical mechanics models

The Exact Models we considered either assumed an infinite population or posed severe restrictions on the nature of the population. Statistical mechanics methods, such as those used in physics can be suggested as a compromise. The idea here is not to track the exact behaviour of every individual in the population, but to provide exact statistics about macroscopic statistics about the population, such as mean fitness and mean number of individuals in a particular schema. Such methods are likely to provide some very interesting insights into the relative performance of crossover and mutation.

7.6 Application of related mathematical works

The use of the Markov Chain Model to model genetic algorithms was an example of the practical application of a well-studied and developed mathematical model. In addition to the model itself, much prior insight gained regarding Markov Chains was applied to genetic algorithms. There are many other mathematical models that may be similarly applied to model varying aspects of genetic algorithm. For example, there have been attempts to model genetic algorithms and genetic operators in terms of groups[19].

Also, in[4], Fischer and Wegener apply GAs to solving Ising Model problems. The Ising Model is a network model used to study the theory of ferromagnetism, and was considered a typical example with a clear building block structure, suited well for two-point crossover. It was earlier believed that genetic algorithms (chiefly based on crossover) outperform by far evolution strategies (which are based only on mutation), in solving such problems. However, Fischer and Wegener show that mutation performs much better than was previously believed. This highlights how important it is not to overestimate the apparently obvious.

7.7 Specialized work into the high level roles of crossover and mutation

Even though much work has been done regarding the specific roles of crossover and mutation, exactly what these roles are has not been completely understood. More work in this area is required and can be

expected.

8 Conclusion

Although all 16 milestones reviewed have resulted in much knowledge and wisdom regarding genetic algorithm, none have provided us with any evidence as to who is the winner of the "Crossover-Mutation Debate". Nor, is it my belief that any of the quoted "new trends" will result in sufficient evidence to conclude the "Crossover-Mutation Debate" at some future date. Indeed, it is more likely that they will complicate matters further. But then again, the "Crossover-Mutation Debate" arose from an effort to better understand genetic algorithms, and these new methods are likely to result in many new findings, practical uses and ideas. In other words, in the future, we are likely to encounter many more milestones than the mere 16 quoted here.

The "Crossover-Mutation Debate" has no outright winner. Neither crossover nor mutation can be proclaimed superior to the other. Both have their own useful qualities and are likely to prove very useful in theoretical studies as well as practical applications in the future.

9 Acknowledgements

I'd like to express my thanks to my parents, my lecturers at the University of Colombo and my fellow students, for all their support, advice and assistance.

I would specially like to thank Mr. Malik Silva, Lecturer-in-charge/Literature Review, for the guidance he provided throughout the writing of this literature review, and for the pertinent and instructive comments he gave on the draft report that preceded this final report.

References

[1] David Beasley, David R. Bull, and Ralph R. Martin. An Overview of Genetic Algorithms. *University Computing, 15(2) and 15(4)*, 1993.

[2] Mehrdad Dianati, Insop Song, and Mark Treiber. An Introduction to Genetic Algorithms and Evolution Strategies. 2003.

[3] Fadi Draidi and Nawwaf Kharma. Review and Empirical Comparative Study of Parameterless Genetic Algorithms. *Concordia University's DEC LAB Projects*, 2001.

[4] Simon Fischer and Ingo Wegener. The Ising Model on the Ring: Mutation versus Recombination. *Genetic and Evolutionary Computation - GECCO 2004*, 2004.

[5] D. B. Fogel and J. W. Atmar. Comparing Genetic Operators with Gaussian Mutations in Simulated Evolutionary Processes Using Linear Systems. *Biological Cybernetics*, 1990.

[6] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. 1989.

[7] N.P.O. Green, G.W. Stout, and D.J. Taylor. *Biological Science*. 1995.

[8] John J. Grefenstette. *Deception Considered Harmful*. 1992.

[9] Pitoyo Hartono, Shuji Hashimoto, and Mattias Wahde. Labeled-GA with Adaptive Mutation Rate. *IEEE CEC*, 2004.

[10] Gareth Jones. Genetic and Evolutionary Algorithms. 1997.

[11] Kenneth A. De Jong and Jayshree Sarma. Generation Gaps Revisited. *Proceedings of the Foundations of Genetic Algorithms Workshop*, 1992.

[12] Kenneth A. De Jong and William M. Spears. An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms. *Proceedings of the First International Conference on Parallel Problem Solving from Nature, Dortmund, Germany*, 1990.

[13] Kenneth A. De Jong and William M. Spears. On the Virtues of Parameterized Uniform Crossover. *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991.

[14] Aghiad kh. Alkatan and Ashraf H. Abdelwahab. New Crossover Operator Mechanism for Genetic Programming. *12th International Conference on Artificial Intelligence Applications.*, 2004.

[15] Fernando G. Lobo. A philosophical essay on life and its connections with genetic algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, 2001.

[16] Melanie Mitchell. *An Introduction to Genetic Algorithms.* 1996.

[17] A.E. Nix and M.D. Vose. Modelling genetic algorithms with Markov chains . *Annals of Mathematics and Artificial Intelligence No. 5*, 1992.

[18] Nicholas J. Radcliffe. Forma Analysis and Random Respectful Recombination. *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991.

[19] Jonathan E. Rowe, Michael D. Vose, and Alden H. Wright. Group Properties of Crossover and Mutation. *Evolutionary Computation, 10(2)*, 2002.

[20] Kumara Sastry and David E.Goldberg. Analysis of Mixing in Genetic Algorithms: A survey. *IlliGAL report no. 2002012.*, 2003.

[21] Kumara Sastry and David E. Goldberg. Designing Competent Mutation Operators via Probabilistic Model Building of Neighborhoods. *IlliGAL report no. 2004006*, 2004.

[22] William M. Spears. Crossover or Mutation? *Proceedings of the Second Foundations of Genetic Algorithms Workshop*, 1992.

[23] William M. Spears. Adapting Crossover in Evolutionary Algorithms. *Artificial Intelligence Center Internal Report No.AIC-94-019, Naval Research Laboratory, Washington, DC 20375*, 1995.

[24] William M. Spears and Vic Anand. A Study of Crossover operators in Genetic Programming. *Proceedings of the Sixth International Symposium on Methodologies for Intelligent Systems*, 1991.

[25] William M. Spears and Kenneth A. DeJong. Dining with GAs: Operator Lunch Theorems. *Proceedings of Foundations of Genetic Algorithms*, 1998.

[26] M.D. Vose. Modeling simple genetic algorithms. *Proceedings of the Foundations of Genetic Algorithms Workshop*, 1992.

[27] Richard A. Watson and Jordan B. Pollack. Combination and Recombination in Genetic Algorithms. *technical report CS-00-209, Dept. Computer Science, Brandeis University*, 2000.

Note: Where a certain work has been referenced in another reference, the original reference has been cited.