Sonomorphs: An Application of Genetic Algorithms to the Growth and Development of Musical Organisms

Gary Lee Nelson
TIMARA Department
Conservatory of Music
Oberlin, OH 44074
(440) 775-8223
email: gary.nelson@oberlin.edu

Introduction

Oxford zoologist, Richard Dawkins[1] has described his journeys through a genetic cyberspace where he models the evolution of organisms he calls "biomorphs." Each biomorph is a small graphic image drawn by a recursive subdivision algorithm that is driven by a numeric vector. Dawkins likens this vector to a genetic code. In each new generation, biomorphs are bred by causing small random mutations in the vector. A set of "daughters" is born from which he selects a single parent for the next generation based on subjective visual criteria (see Figure 1). Clicking the mouse on one of the daughters makes it the mother of the next generation. Clicking on the mother biomorph (in the center of Figure 1) breeds fourteen new sisters.

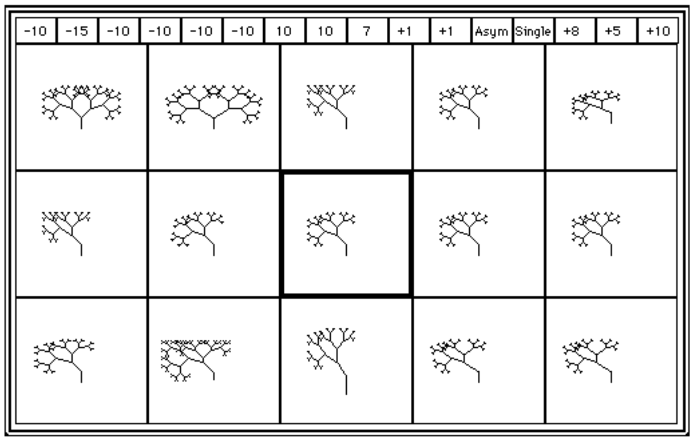


Figure 1. Dawkins' Breeding window (breeding with one parent)

Dawkins' thesis is that complicated organisms are not the result of chance or conscious striving toward a long term goal. Rather, they arise from the accumulation of small changes that assure the survival of the fittest. The process immediately reminded me of the means that composers use to search the hyperspace of musical constructs to find just the right choice for a particular moment in a piece.

[1]Richard Dawkins, *The Blind Watchmaker*, W. W. Norton, New York, 1987 (ISBN 0-393-30448-5)

This research is part of my general interest in mathematical models for musical composition. It overlaps my studies of fractals, chaos, cellular automata, artificial life, iterated function systems (IFS), quaternions, Lindenmayer systems and chaos. The common thread in this work is the recognition that complex structure can emerge from repeated application of a limited set of simple operations.

A major goal of these studies is to find optimum methods for structuring musical organisms. These methods must be rich enough to produce a broad range of musical expressions yet simple enough to be effective in real time environments for composition and performance. An equally important goal is the discovery of something about the nature of subjective musical choice so that aspects of the process can be automated in programs for algorithmic composition.

In paralleling Dawkins' research in the sound domain, I coined the word "sonomorph" to describe elements of musical compositions whose size ranges between a motive and a phrase. I have used a variety of strategies for constitution of the genetic code and selection criteria. The result is a body of pieces where musical phenotypes evolve through a combination of reproduction and genetic mutation. The genetic code that is passed to each generation is interpreted with an algorithm that generates musical structures. The genetic code is mapped onto musical parameters and presented to the composer for subjective aural evaluation. The sonomorphs that "survive" become the progenitors of future populations.

Later in this paper I will describe a model for evolving rhythmic patterns as a case study of one of the first genetic algorithms I have applied to music. At the end, I will suggest some further developments and directions. First, we need a short general introduction to genetic algorithms.

Genetic Algorithms

Genetic algorithms are based roughly on natural evolution. Individuals in a population are represented by a string of numeric or symbolic parameters that hold the genetic code or genome of the organism. Each parameter symbolizes a locus on a chromosome. The value of each parameter encodes the allele[2] at that locus.

Genetic algorithms evolve a population by examining the genome of each individual and assigning a fitness score that is based on the suitability of the organism for performing a task or fulfilling a function. Successful organisms survive to become the parents of the next generation by a process of breeding and mutation. Figure 2 shows the steps in a typical genetic algorithm.

1. Create random initial population
2. Evaluate each individual
3. Select parents
4. Breed
5. Add children to next generation
6. Replace parent generation with children
7. If end condition not satisfied repeat from step 2

Figure 2. Flow chart of a genetic algorithm.

[2]either of a pair of contrasting characteristics inherited according to the Mendelian law.

The following is only a brief description of the components of a genetic algorithms. I refer the reader to David Goldberg's excellent textbook on the subject for a more rigorous exposition.[3]

Genome Encoding

The classical example of genome encoding is the bit string. Individuals in a population are represented by a string of some specified length. The bits, taken singly or in groups of fixed or varying size, convey the genetic information relative to each allele.

Other encoding schemes use strings of integers or real numbers. Each number holds the current state of a locus on the chromosome. Dawkins uses this method for his genomes in *The Blind Watchmaker*. (See top of Figure 1.)

In a third category we find genomes represented symbolically. Each locus contains a token that may stand for a function to be executed or an action to be performed. The genome can be used to describe an individual's behavior in an environment. Food foraging actions and criteria for selection of a mate might be coded this way. In algorithms for game playing or machine learning the genome may even be a computer program and the alleles small pieces of code. The success of the organism in solving a particular computing problem determines its fitness score.

Fitness Tests

Fitness tests vary with the goals of the genetic experiment. In the bit string model, fitness might be determined simply by viewing the bits as an unsigned integer and using the value of that integer as the fitness score. Alternatively, we might add the 1's in the string and reward individuals with the highest or lowest sums or sums within a particular range.

My genetic model of evolving rhythmic patterns uses a bit summing test. In this case the bit string is interpreted as a series of equally spaced pulses. If a bit is on, a note is articulated; if a bit is off, a rest is made. Selecting for high bit counts evolves rhythms that tend toward high density. After a few generations the individuals play notes on nearly all of the pulses. Selecting for low bit counts creates an opposite tendency toward thinner rhythmic texture and finally the silence of extinction. In simple fitness tests like these there is a rapid conversion toward the idealized state implied by the test.

Another classic genetic experiment involves the simulation of a population of ants who are given a food foraging task. Bits of food are placed in a grid and each ant is given a period of time or a number of moves to search the grid. The ants who bring the most food back to the nest are selected for breeding future generations and the population evolves into better foragers. The genome represents a set of actions (move forward, turn, recognize food, etc.) that an ant can take in its search. Some models provide the ants with simulated eyes and feelers along with a memory of successful series' of actions.

[3]David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, New York, 1989 (ISBN 0-201-15767-5).

Richard Collins' dissertation[4] provides another general introduction to genetic algorithms along with case studies of his many "Ant Farms." It is one of the wonderful books I have found while looking for something else.

Breeding

Once the most suitable parents survive the fitness test, breeding the next generation begins. Again there are variants of the breeding algorithm. Some algorithms depend on the number of parents chosen for a child of the next generation. The obvious choice from a human perspective is two parents.

In the two parent model the bits are combined using a crossover method. To begin the breeding, one of the two parents is chosen by a coin toss. The bits in each parent's genome are scanned in parallel. When the first pair of bits is considered, a coin is tossed again. If it is tails, no crossover occurs and the first bit for the child's genome is taken from the parent that was chosen in the very first coin toss. If the coin turns head up, a crossover occurs and a bit is taken from the opposite parent. Figure 3 illustrates breeding of two parents with three crossover points.

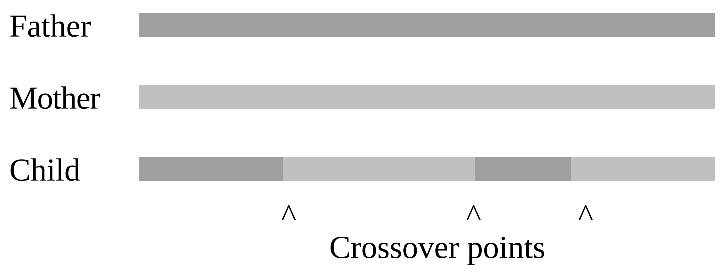


Figure 3. Breeding with three crossovers.

Figure 3 shows that selection began with the father and crossed over at the points indicated by the arrows.

In computer simulations, the coin is replaced by a probability function with a variable weight. If the weight favors tails, few crossovers are made and longer strings of bits are taken from each parent. The result is the inheritance of larger sequences of genetic code and greater dominance of features from one parent or the other. If the weight favors heads, crossover is more frequent and shorter substrings are passed to the next generation.

[4]Richard J. Collins. *Studies in Artificial Evolution,* Ph.D. dissertation in Computer Science, University of California at Los Angeles, 1992.

Dawkins uses a three parent model in his "Triangle" operation (see Figure 4).

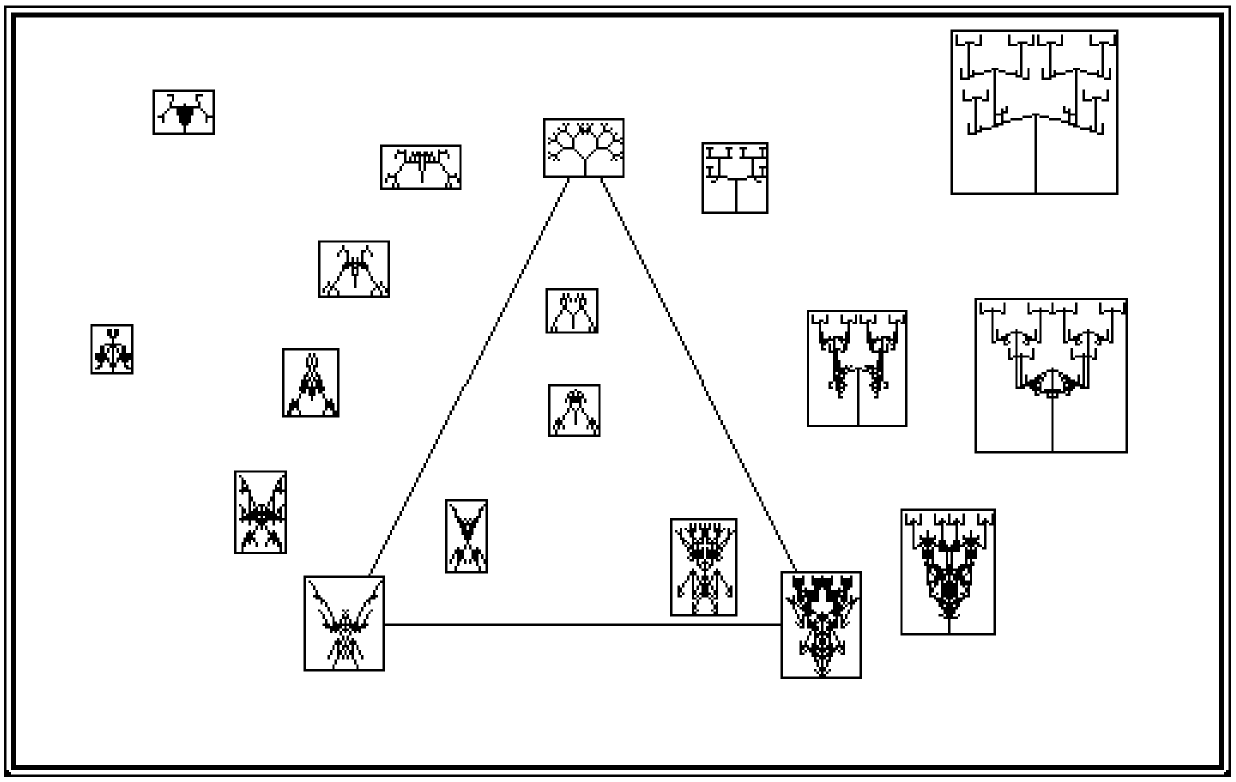


Figure 4. Dawkins' Triangle window (breeding with three parents).

In this model, the same three parents are used for all children of the next generation. When the mouse is clicked in the screen, the genome for a child is constructed as an average of the parents' genomes weighted inversely by the distance of the mouse click from each parent.

Single parent models also exist. The operation screen in Figure 1 shows Dawkins' implementation of this model. Since no breeding is possible, the crossover stage is skipped and children evolve entirely by mutation of the single parent's genome.

Mutation

Like crossover, mutation is controlled by probability. In the bit string genome, the probability of mutation determines the likelihood that a bit will change state from zero to one or vice versa. As the bit string is scanned, a weighted random function decides whether to alter the state of each bit. High probabilities reduce the effectiveness of the parent selection step and tend toward random evolution. If the mutation probability is low it can perturb the conversion effect that the fitness test encourages without altering the general flow of evolution.

Mutation in Dawkins' parametric model is carried out by incrementing or decrementing the values of a few parameters. These parameters determine the lengths of lines, turning angles, symmetry, and number of branches. Each child differs from its parent by small steps in only a few genetic features and each sibling undergoes a slightly different mutation.

Dawkins uses a variable mutation weight and gives the users of his program the option of including the weight as part of the genome. This gives the possibility that the amount of mutation between generations can increase or decrease so that the rate of evolution will appear to accelerate or decelerate.

Implementation

My genetic experiments are carried out for the most part with **MAX**.[5] **MAX** is an objected-oriented language for programming interactive musical processes. It has a graphic user interface where functional objects can be linked together in larger systems. It accepts input from the Macintosh keyboard, the mouse, and instruments with MIDI capability. In addition to MIDI output, **MAX** can control compact and video disk players, display still and animated graphics, and communicate with digital signal processing chips. **MAX** also permits the installation of user written objects in C so that functionality can be extended beyond the already large set of standard features.

The model I have chosen to describe here is a relatively simple one. A bit string is used to define a rhythmic pattern. A one means play a note; a zero means don't play.

Population

The population in this experiment consists of an array of individuals arranged on a square grid of variable size. The grid is divided into variable sized "islands." Figure 5 shows a 16x16 grid with sixteen 4x4 islands.

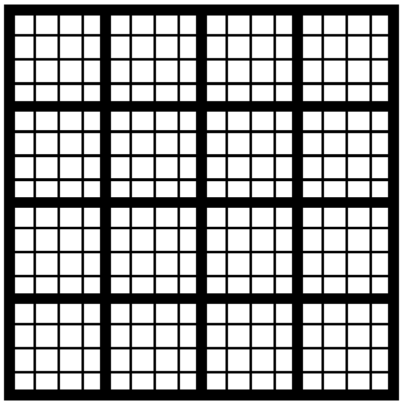


Figure 5. A population of 256 individuals living 16 per island.

The genome for each individual is a one dimensional array of 32-bit integers. The length of the genome is variable. The population is initialized by filling all of genomes with pseudo-random numbers.

[5]Miller Puckette and David Zicarelli, OpCode Systems, 3950 Fabian Way Suite 100, Palo Alto, CA 94303

Global Controls

The initial state of the "world" is set by means of global controls (the God function?). Grid dimension and genome size are needed to generate the initial random population. Island size and walk length govern the children's field of search for parents. Probabilities for migration, crossover, and mutation characterize and influence evolution through the generations. Figure 6 shows a **MAX** patch that provides slider controls for these variables, displays their current values, and communicates those values to the rest of the environment.

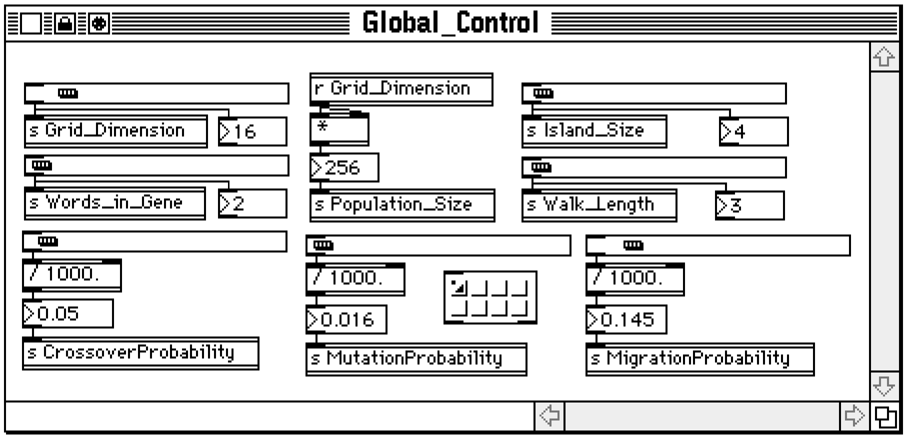


Figure 6. Global controls for initialization, parent selection, and breeding

Migration

The concepts of islands and migration were not mentioned in the earlier introduction to genetic algorithms. This model was suggested by one of Collins' Ant Farm experiments.[6]

Islands serve as breeding colonies. During the selection of parents, search and evaluation of strength is limited by island boundaries. Migration may occur just before parent selection and breeding. Using a variable probability individuals may exchange places with one of their eight immediate neighbors.

My migration algorithm tests for obvious degenerate cases. Individuals are prevented from migrating within an island. This would leave the gene pool of an island unchanged. A simple serial method of test and exchange could cause an individual to migrate several times in a generation and wind up very far from home. This wandering is prohibited with a rule that prevents an individual may from moving more than once in a generation. The grid, like the islands, is toroidal so individuals near the edge of the grid can migrate to the opposite side. This is no flat earth model.

[6]Collins. *Studies in Artificial Evolution*

Selection of Parents

Two grids are used to store genomes: one for the current generation (the parents) and one for the next generation (the children). In preparation for breeding, an empty grid like the one shown in Figure 5 is created with a square to hold each new child. Empty-gene children search for potential parents by performing a random walk around their island. They start with the parent who occupies the place on the parent's grid that corresponds to their own square on the children's grid. Figure 7 shows the **MAX** patch that performs parent selection.

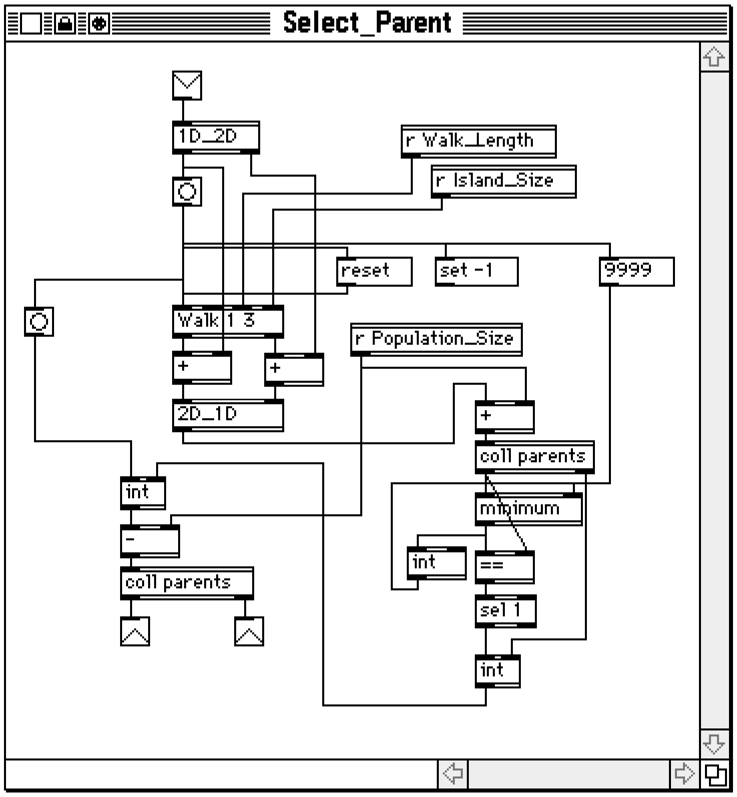


Figure 7. MAX patch for parent selection

During the walk, the child can only move to squares on the grid that are immediately adjacent. With each step there are eight possible directions. Islands are toroidal so an attempt to walk off the edge results in a wrap around to the opposite side of the island. As they walk, children take note of the strengths of the parents they encounter. At the end of the walk they select the parent with the greatest strength. The children are fickle. If there is a tie, children favor the parent encountered nearest to the end of the walk.

Walk length is a global variable (see Figure 6). A short walk means that a child may never encounter the strongest parent on the island. A long walk raises the probability that the child will always choose the strongest parent on the island. Like the other global variables in this model, small changes in walk length can radically alter results. In the present model, children select parents with the least bits on. The walk is repeated for each child to select a second parent (see Figure 8). There is a likelihood that long walks on small islands will lead to the selection of the same strong individual as the second parent. My search algorithm prevents this by omitting the already-chosen parent from the second search.

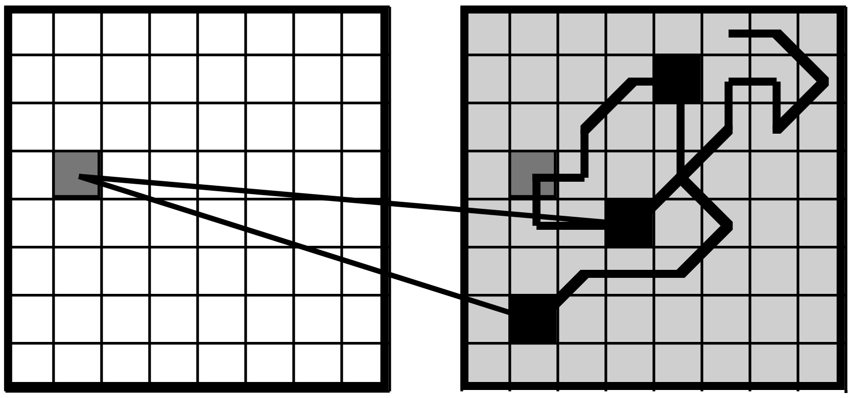


Figure 8. Random walk in search of strong parents. Strongest parents in black.

Breeding and Mutation

The breeding step in my model uses two parents whose genomes are subjected to the crossover algorithm described earlier. The resulting child is passed through the mutation function. Both crossover and mutation are controlled by global probabilities. Breeding and mutation are shown in Figure 9.

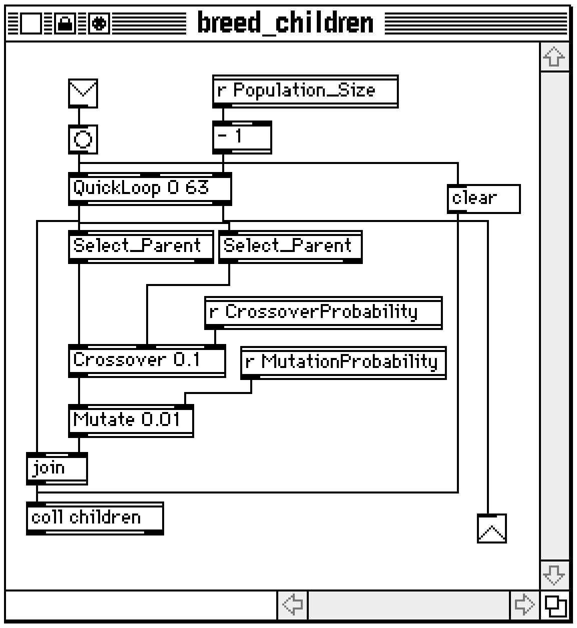


Figure 9. MAX patch for breeding and mutation

I wrote **Crossover** and **Mutate** as external objects in C. The **Walk** object in Select_Parent is part of a set of random generators that share the same seed so processes with random components can be saved and recreated.[7] The **join** object is part of a library of list manipulation objects written by James McCartney and distributed with **MAX** as "unsupported objects."

[7]The current version of y MAX implemntations of crossover, mutation and several other random processes are available via anonymous ftp at kahless.isca.uiowa.edu/pub/ftp/max/NelsonRandom.hqx.

Interpretation

With each passing generation it is possible to pause time and examine the progress of the algorithm. Remembering that the goal of this research in the creation of original musical works, examination must occur in the domain of sound.

Mapping

This is perhaps the most difficult step. Up to now we have only a mechanism to assure that complex things will happen. Rendering this complexity into music can take us down many thorny paths. At the moment, I have chosen the relatively simple interpretation that I described earlier, the articulated pulse train. To "play" a genome, the bits are iterated and each 1 produces a note. Similar genetic structures produce similar rhythms.

Pitch Material

The pitch material for this model is fixed as a sequence that matches the length of the genome. As each bit is examined, we advance in the sequence. If a bit is on, we hear the note. It is somewhat artificial to freeze one parameter in this way but it facilitates examination of similarities in long genomes.

Duration

If a note is not played on a given pulse, the duration of the previously played note is extended. The pulse is interpreted as an eighth note and the maximum extension is a half note.

Musical Examples

The examples on the next two pages (Figures 10-13), illustrate in score the evolution on one island through 30 generations. Figure 10 is generation zero, the random initial population. Time is paused every ten generations and the current state of the population is notated.

I will not describe everything that is observable in these examples. Rather, I will hit the high points and a few of the details and leave the reader to discover others.

As we look at Figure 10 and scan down the staves through each pulse, we see the common "tune" that the sonomorphs share. The first pulse is Eb, the second is A and so on. Comparing the parts we find that no two are exactly alike. This is to be expected because of random initialization of the population. Moving on to Figure 11, we find that ten generations have already resulted in a fair amount of conversion. In fact, parts 5, 8, and 9 are identical.

In Figure 12 we see that parts 5 and 9 have changed somewhat in the second and third measures and that the changes are the same in both parts. Part 8 has held its shape from generation 10 with only a small change in the first measure. Parts 3, 4 and 7 are now identical. In Figure 13 we observe significant conversion. From the last pulse of the second measure (Eb) all of the parts are the same except for part 3. Finally, we note the effect of the fitness selection for fewer bits. The number of notes has decreased in each part from an average of 20 or so in Figure 10 to about 12 in Figure 13.

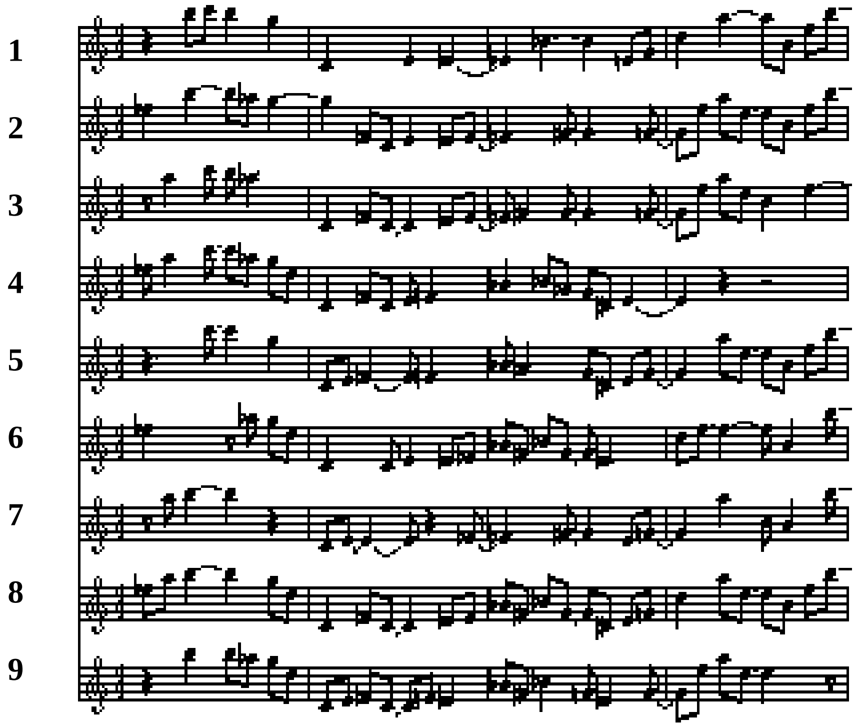


Figure 10. Generation 0

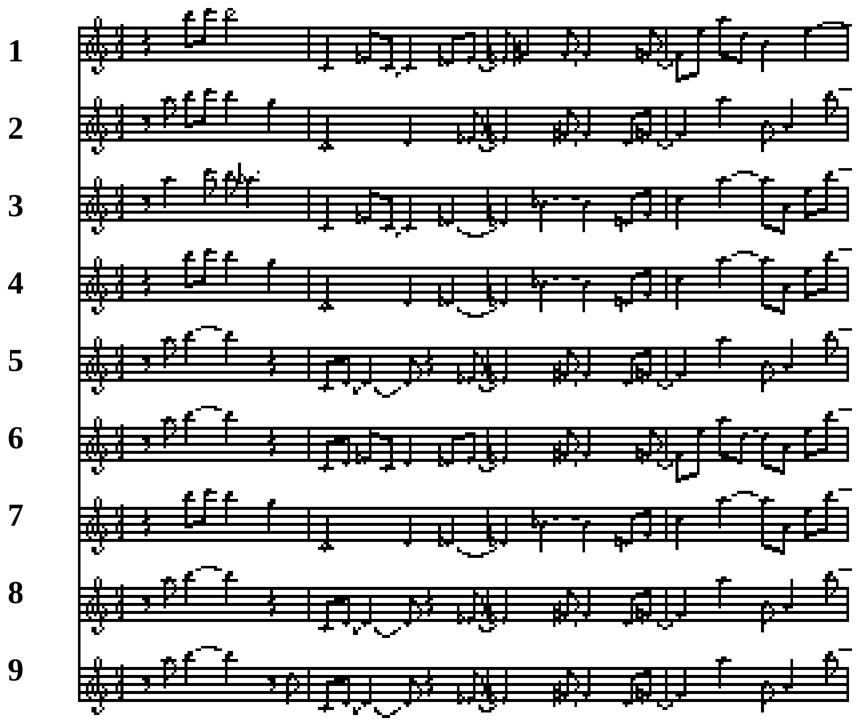


Figure 11. Generation 10

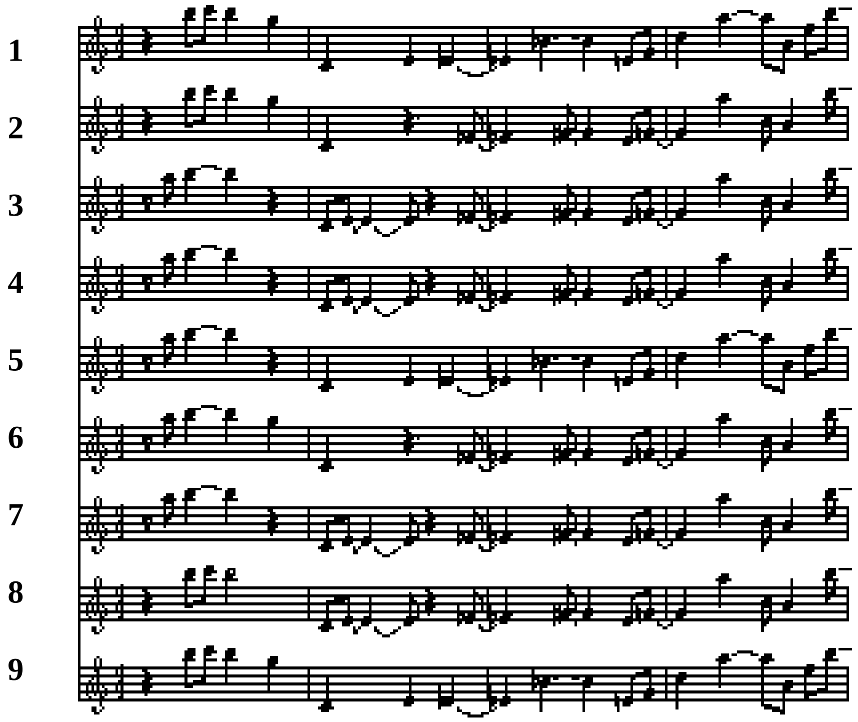


Figure 12. Generation 20

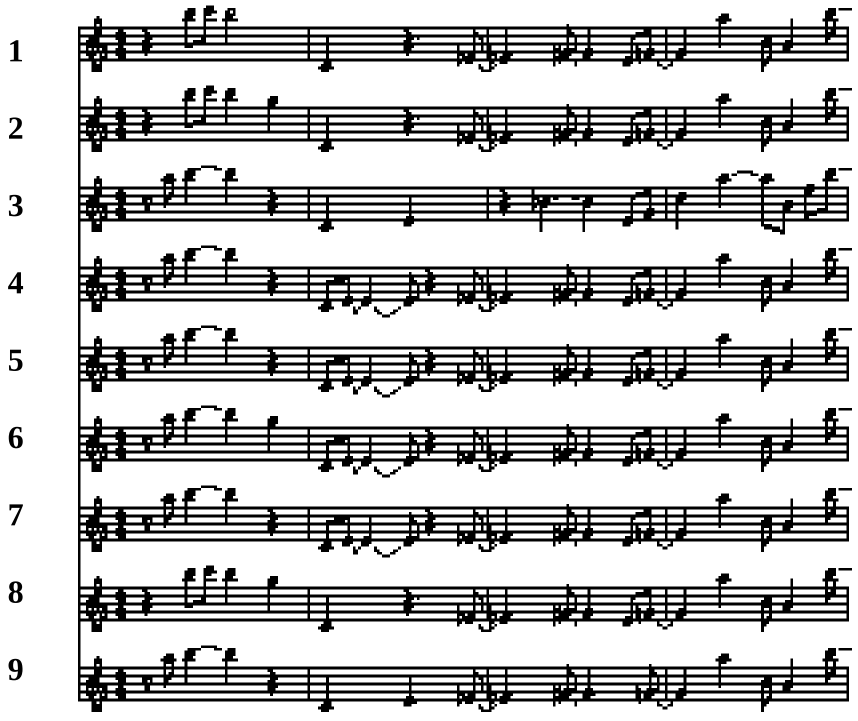


Figure 13. Generation 30

Subjective Selection

The power of *The Blind Watchmaker* program is the ability to make subjective selection of parents with vaguely defined criteria. In the present work we assume the selection process to be based in some kind of informed esthetic.

My construction of a mechanism for subjective selection with musical examples like those in Figures 10-13 is only partial complete. Music presents a difficulty that we do not find in Dawkins' model. With *The Blind Watchmaker* program we can compare and evaluate an entire generation with a single glance. If all of the sonomorphs in a population were presented at once the cacophony would make it impossible to distinguish let alone choose the stronger individuals. Presenting sonomorphs sequentially taxes the memory even in populations as small as the one described above.

The provisional solution is a combination of sonic and graphic feedback from the experiments. During the pause between generations the controls in Figure 14 are available.

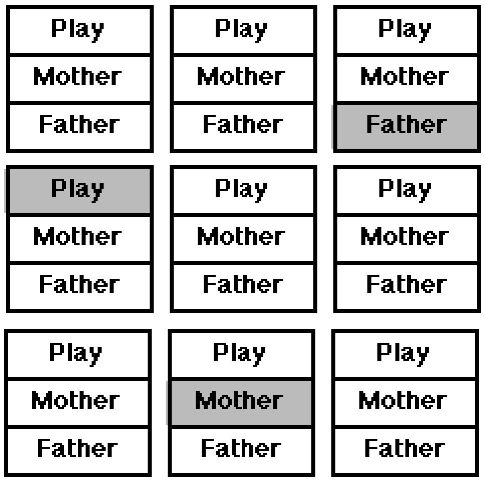
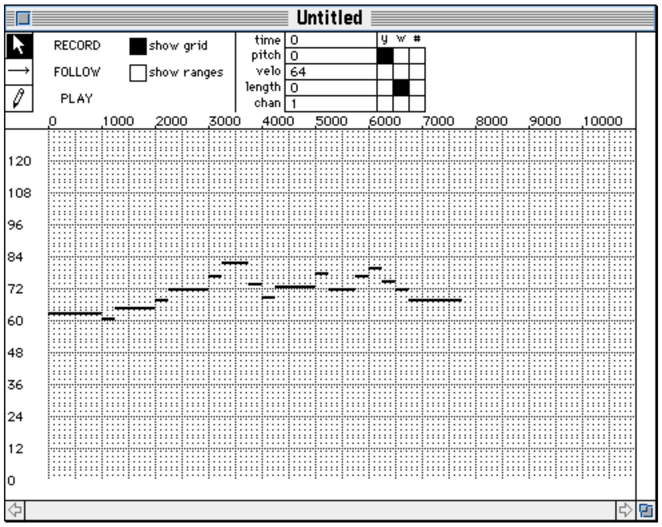


Figure 14. Control panel for subjective selection of parents.

These controls provide a multiple toggle. In the nine clusters of buttons only one of each kind may be selected at a time. Selected buttons are shown in gray. When a "**Play**" button is pressed the corresponding sonomorph begins to sound and its graphic representation is brought to the fore (see Figure 15). If another **Play** button is selected, the sounding sonomorph stops playing and the one newly selected begins. As we audition the sonomorphs we can select "**Father**" and "**Mother**" buttons to indicate our choice of parents for the next generation. When we are happy with the parents we have selected we create a brood of nine children who become the subject of the next fitness test.

To help in the audition process, I use a visual aid in the form of the **MAX** object **explode**. **explode** is another unsupported object that comes with **MAX** but it is robust and quite powerful. It can be used as a sequencer/recorder and presents its contents in a form of "piano roll" notation (see Figure 15). The contents of **explode** can be written as a MIDI file and exported to **Finale** or other notation software for printing in the traditional form of Figures 10-13. This is a slow process that is not very useful for a large number of auditions. Obviously, a future version of **MAX** will require a realtime notation interface.



time 0
pitch 0
velo 64
length 0
chan 1

Figure 15. Sample graphic output from MAX' explode object.

Further Work

Work with mathematical models like genetic algorithms is almost overpowering. The breadth of the field is so great that it is difficult to focus for very long on simple examples and the extraction of basic principles. The rhythmic articulation model that I have used as an example, however interesting, is probably not a powerful tool for making large compositions. The operations are certainly too limited and too simple to make sophisticated musical utterances.

In addition to expanding my models to include subjective selection, I am exploring other interpretations of the genomes. An interpretation that describes traversal through multiple musical dimension has proved promising. In this interpretation, successive bits are grouped to form small integers. Each integer represents one of a set of operations on a current point in musical space. If we define such a point with a set of ordinates for pitch, start time, loudness, spatial location, and any of the many parameters that combine to create timbre we can think of sonomorphs as living in a multidimensional environment.

As operations are extracted from the genome, points are transformed through the environment. One operation might mean simply "get louder." Another could mean decrease the sharpness of attack on the amplitude envelope of a synthesizer. Many of the operations would result in sound events (notes) while others would modify parameters that would be used to make a subsequent sound.

Finally, I hope to integrate genetic algorithms with the other mathematical models I mentioned in the introduction. In particular, the multidimensional environment that I described in the two preceding paragraphs was inspired by my reading about Lindenmayer Systems.[8]

[8]Przemyslaw Prusinkiewicz and Astrid Lindenmayer. *The Algorithmic Beauty of Plants*, Springer-Verlag, 1990 (ISBN 3-540-97297-8).

Conclusion

I have presented a summary of my preliminary exploration of genetic algorithms. In spite of the technical focus of this exposition, I must tell my readers that the musical results are the most important part of this exercise. If a mathematical model or a particular method of interpretation or mapping persistently produces music that I regard as ugly or poorly formed, I discard the model.

I'm often asked why I persist with this kind of research and why I take the time to describe my techniques. If I did not believe in the fertility of this ground I would make my music in more conventional ways. If I did not believe in the importance of sharing the results of my labor as both process and product, I would just compose music and keep my methods to myself.   I hope that this paper will encourage others to begin or continue work with mathematical models for musical composition. I will be most interested to hear their results and, when possible, to share ideas in person.

Numbers don't make art. People do. It's all in the choice of the tools, the colors, the textures and the rendering.