Interactive, Evolutionary Textured Sound Composition

Sidney Fels[1] and Jonatas Manzolli[2]

[1] Dept. of Electrical and Computer Engineering,University of British Columbia, Vancouver, BC, Canada, ssfels@ece.ubc.ca
[2] Interdisciplinary Nucleus for Studies on Sound Communication (NICS), University of Campinas (UNICAMP), Campinas, SP, Brazil, jonatas@nics.unicamp.br

Abstract. We describe a system that maps the interaction between two people to control a genetic process for generating music. We start with a population of melodies encoded genetically. This population is allowed to breed every biological cycle creating new members of the population based upon the semantics of the spatial relationship between two people moving in a large, physical space. A pre-specified hidden melody is used to select a melody from the population to play every musical cycle. The overlapping of selected melodies provides an intriguing textured musical space.

1 Introduction

Algorithmic composition systems suggest intriguing possibilities for the production of music. They offer the potential to explore unique musical spaces with the aid of computational processes. However, they often suffer from being overly algorithmic and thus too deterministic in the types of music produced. This determinism often leads to a sense of the music lacking expression. To address this problem, our strategy adds three main interactive components to the algorithmic system: first, support for significant human interaction with the system to adjust, in real-time, expressive components; second, the addition of a constrained amount of stochasticity to the process to add complexity; and third, a method of providing control of the system at the process level rather than direct control of musical production to maintain the advantages of algorithmic composition. Our strategy is achieved by using an interactive, evolutionary approach to music composition. Rowe [14] pointed to the concept of interactive musical systems as a composition by refinement. He commented

> because the program reacts immediately to changes in configuration and input, a user can develop compositional applications by continually refining initial ideas and sketches.

In our system, a genetic algorithm underlies the musical process similar to works by Biles[1], Horowitz [5] and our recent system VoxPopuli [12]. In the research presented here, the evolutionary algorithm has control parameters for altering

the degree of stochasticity used as well as adjusting the texture of the music. The semantics of the control parameters provide a basis for creating the music interface. Specifically, the performer/composer has access to the process parameters through a semantically relevant mapping of the relationship between two objects. However, the performer does not have direct control over the specific attributes of the melody structure since these are controlled by the evolutionary procedure. The underlying structure of the process is described in section 3. Since the performer controls a process rather than the details of the music, we have also developed techniques for visualizing the process itself to assist the performer in guiding the melodic structure. Thus far, we have been able to generate interesting and rich melodies through experimentation.

2 Related Work

In recent years, Evolutionary Computation has been developed to mirror biological evolution. It is useful to create algorithms and structures of higher levels of complexity. Both biological and simulated evolutions involve the basic concepts of a genotype and a phenotype, and the processes of expression, selection, and reproduction with mutation. Miranda [11] suggested:

> a plausible approach to alife-like musical models and simulation is to consider music as an adaptative system of sounds used by a number of individuals in a certain community, engaged in a collective music making experience.

In the research presented here we use an evolutionary paradigm to control an interactive compositional process in which the genotype is described by a melodic line represented by MIDI parameters. The phenotype is the resultant sound produced by the computer in real time. In biological terms, expression is the process by which a phenotype is generated from a genotype. The selection process developed here selects melodies from the population closest to a hidden melody and plays them. The hidden melody does not affect the evolution of the population, instead, it is only used to determine which melody is heard. The idea of using hidden melody to guide an evolutionary musical process was inspired by a quotation from the XIII century written by Guido d'Arezzo[17] (p. 123):

> to find an unknown melody, most blessed brother, the first and common procedure is this: You sound on the monochord the letters belonging to each neume, and by listening you will be able to learn the melody as if from hearing it sung by a teacher.

Our approach mimics this old musical procedure using a relation between two objects as the paradigm for a monochord and letters of a Gregorian Chant.

One of the first applications of genetic algorithms (GAs) to sound design can be found in the work of Roads [13] in which evolutionary strategies are used to regulate parameters of a granular synthesis process. In this process, thousands of parameters are combined in a complex way to produce a desired sound result.

Following this original work, a series of articles related to the production of sound synthesis using GAs to deal with the complexity of an automated synthesis process were created [7][8][2].

In recent literature, GAs have also been applied to produce evolving trajectories of musical material [6]. Biles [1] presented a genetic algorithm-based model that mimics a student learning to improvise jazz solos under the guidance of a human mentor. In Horowitz's [5] development, an interactive system uses GAs to develop a criterion for distinguishing rhythmic patterns, producing a large number of variations. We have also studied applications of GAs to interactive composition [9] [10]. Evolutionary Computation is also increasingly employed in Computer Graphics to create scenes and animations Sims [16]. In these applications the rules are learned by the system through its interaction with the user, as described by Fogel [4].

3 Composition Process and Melody Representation

The overall system is shown in figure 1. There are three distinct sub-systems: the Music Cycle, the Biological Cycle and the Visualization sub-systems described below. The Music Cycle sub-system is responsible for selecting which melody will be played and when. The Biological Cycle sub-system is responsible for updating the population melodies. The Visualization sub-system is responsible for providing the performer with a sensible view of the evolutionary process and the musical output. Each of these sub-systems is described below.

Each sub-system operates on the melody population shown in figure 1. The melody population is composed of genetically encoded melodies. The notes are the genes in our genetic algorithm. A note's genetic structure consists of these components:

1. MIDI note number;
2. velocity of the note;
3. duration of the note;

The genome for a given melody can be of any length but we currently limit it to 12 notes. Likewise, the population can be of an arbitrary size; we currently use 20 members. Thus our population consists of 20 members, each of which is 12 notes long. Note that other components can be included to the genetic structure for a melody. As we are using only MIDI based music synthesis, other note parameters could be MIDI note controls such as the program change, channel number and panning parameters.

In addition to the melody population, we also encode a special, single melody we call the "hidden melody". Normally, it is not reconfigurable during the composition process and is established *a priori*. The hidden melody is used by the Musical Cycle Sub-system for selecting which melody to play from the melody population as described in 3.1.

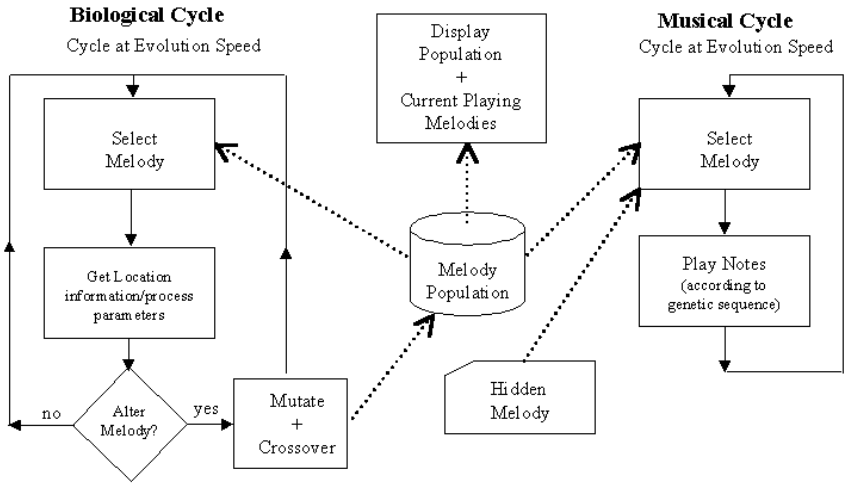

Fig. 1. Three processes run concurrently to control the musical process. The first is the Musical Cycle process which continuously selects melodies from the population closest to the hidden melody and plays them. Depending upon the cycle time, the melodies may overlap. The second is the Genetic Cycle process which continuously updates members of the population using mutation and cross-over. The third process provides visualization of the current melodies.

3.1 Musical Cycle Sub-system

The Musical Cycle Sub-system is responsible for selecting which member of the current population to play and when. A Musical Cycle consists of the following two steps:

1. Select the melody to play.
   The selection of the melody is done by choosing the melody closest to the hidden melody. We determine distance by calculating the mean square difference between gene sequences of two melodies. Mathematically, this is expressed:

$$D = \sum_{i=1:all\_notes} \sum_{j=1:all\_note\_structures} (g_{ij}^a - g_{ij}^b)^2 \quad (1)$$

   where g_{ij}^a is the note structure j of note i for the one gene and g_{ij}^b is for the other. Remember that a gene has MIDI information substructures which are combined to form the melody genome.
2. Play the melody.
   All the notes are scheduled by the system to play according to their duration parameter as specified in the genes. Duration is specified relative to each note's previous note as a power of 2 ± 1 delay units of a fixed tempo. As the duration of each note is a function of the genes, the overall duration of each melody in the population is different. Once scheduled, melodies play autonomously and the musical cycle continues without waiting.

The Musical Cycle repeats at a parameterized frequency called the music cycle time allowing melodies to be overlapped or spread out. When overlapped the music acquires complex texturing whereas when they are spread out a minimalist feel occurs.

By using the distance from the hidden melody as a selection criterion, the melodies have some gravity towards an a priori melodic structure providing some root to the sound produced. However, the biological process, described in section 3.2, provides stochastic variation from the hidden melody to give interesting deviations. Furthermore, the amount of variation is controlled by the performer to provide expressive possibilities. Remember, the hidden melody does not affect the evolution of the population; rather, it determines which member of the current population is heard.

3.2 Biological Cycle Sub-system

Each cycle of the Biological Cycle Sub-system determines a new member of the population based on a currently selected member using a round-robin strategy. The two processes implemented thus far are mutation and crossover applied to MIDI note number, duration and velocity. We chose these as they are the most typical genetic processes for modeling. Additional processes are left for future work.

In mutation, each gene in each genome in the melody has a random chance of being changed to a new value according to the *Mutation Probability*. The

performer's interaction with the system sets this parameter dynamically, as discussed in section 4. Mutation provides a degree of randomness in the melody population. This allows the population to wander freely in melody space facilitating freshness in the melodic structures. The amount of wandering in the melody space (genotype variation) is controlled by the Mutation probability. The deviation that is actually heard (phenotype) is mediated by the hidden melody, thus any extreme mutations from the hidden melody are unlikely to be heard. To prevent large variations and maintain note alignment, we constrain note number mutations to a circular two-octave scale and durations to powers of 2 (± 1). Mutations can cause large variations in a gene's value. We constrain the genetic variations to moderate this. When mutation causes a note to move outside a two-octave scale, it is reassigned a value inside the restricted two octaves. This is accomplished by considering the scale as circular; thus, any note number higher than that of the top of the two octaves is circled around to the bottom of the scale (and vice versa).

For crossover, two additional melodies from the population are selected as mother and father melodies. For each gene, with a given *Crossover probability*, crossover will occur. Crossover is performed by replacing the genes (notes) with one from the mother or father melody stochastically. If a gene is not selected for crossover the original is left untouched. Thus, after crossover, depending upon the Crossover Probability, the new melody is a mixture of original notes plus some notes selected randomly from either the other or father melodies. The Crossover Probability is determined from performer interaction with the system, as discussed in section 4. Crossover causes mixing of the current population to produce offspring that are related (at least in genotype) and thus sound similar when played. Like mutation, genotype variation is controlled by the Crossover Probability but the phenotype is still mediated by the hidden melody, so large variations from the hidden melody are likely not to be heard.

4 Interaction Framework

In our current direction we attempt to map performer interaction semantics to the evolutionary semantics provided by the process control parameters. Mapping semantically related control parameters reduces cognitive load, increases intimacy, and is the basis for direct manipulation interfaces[15]. We do this by mapping the distance between two objects to the mutation probability and the angular deviation formed by the vector joining the two objects with a reference angle to the Crossover Probability. This mapping is important since it makes the manipulation of melodic variation more predictable, easier to learn and enhances intimacy with the system.

We have experimented with two different ways of interaction. The first technique is to use two cubes as the objects. These cubes are actually the receivers of a 6 degree-of-freedom tracker (Polhemus Fastrak). The position and orientation of the receivers are measured and used to compute the distance between them

and the angle relative to a horizontal line. Figure 2 shows a person using the Fastrak sensors with the system.

Fig. 2. Photo of person playing with the system using the Fastrak.

In the second technique the two objects are people. The positions of the people are tracked using a local positioning system (LPS) developed in-house at the University of British Columbia. The LPS system uses infrared-based active badges and camera modules for tracking the position of moving objects. The idea behind using the interaction of two people to manipulate the genetic algorithm comes from thinking about the semantics of how two people interact with each other and their environment. We take the physical proximity of two people and map this to the mutation probability; thus, if they are close together the population becomes relatively stable with little mutation to explore the melody space. However, if the two people move farther apart, the mutation probability increases which in turn introduces new genetic information into the population.

The relative angle between the two people and the wall sized visual representation of the population (see section 5) is related to the crossover probability. Thus, the more two people interfere with each other's view of the screen the more likely it is that the offspring of the current crossover mixture will contain genes from the mother and the father melodies. Thus, the two performers can

interact with each other while the underlying evolutionary process controlling the melody will relate to their relationship.

The additional parameters, such as tick frequency, biological, and musical cycle times, are adjustable in real-time using a graphical user interface (GUI) (see figure 3). We plan to map object manipulation semantics to these parameters as well in a future version of the system.

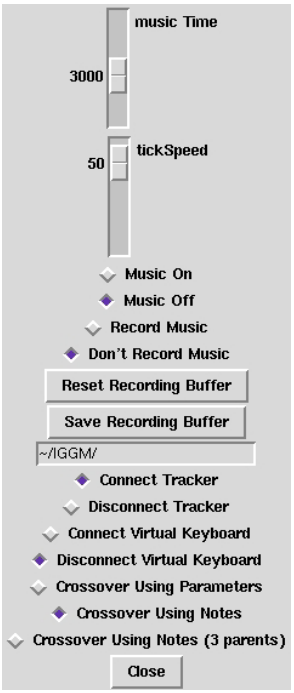

Fig. 3. Graphical User Interface for the system.

In initial experimentation we discovered that the system provides a reasonable amount of controllable melody space. The use of the hidden melody always keeps the navigation reasonably well constrained so that melodies playing concurrently work well together. Further, the ability to adjust the musical cycle time enables flexibility in the texture of the music. The ability to control the biological time allows the performer(s) to adjust how fast the population can explore the different melody spaces possible. The overall effect is that music with interesting texture as well as interesting stochastic variation focused around a predefined hidden melody is produced. Examples of some of the music that is produced are available at: **http://www.ece.ubc.ca/~hct/IGGM**.

5 Visualization

Finally, we have implemented a mechanism for visualizing both the current melodies that play concurrently as well as the overall population. We use the same concept used by [3] for visually representing a gene sequence. We represent each note by a line segment. The length of the line is dictated by the duration of the note. The angle of the vector relative to its neighboring note is related to the two note numbers of adjacent notes. Finally, the velocity of the note is mapped to the thickness of the line. Specifically, the following formulae are used:

$$length = log_{10}(duration) + 1) * 6 \tag{2}$$

$$angle = (((noteNumber2 - noteNumber1) * 10) + 180)/(2.0 * \pi) \tag{3}$$

$$width = ((loudness/127.0) * 4.0 + 1) \tag{4}$$

Where *duration* is the number of ticks the note is sustained, $noteNumber(n)$ is the MIDI note number of adjacent notes, *angle* is in radians, *loudness* is the velocity of the note being played and *width* is in pixels.

The visualization can be displayed either on a typical computer monitor or on a large projection screen. The large projection is useful when two people play music with the system. This is because the dynamics of the two people's interactions map to the control parameters, requiring them to have a large amount of space in which to move. Currently, the space we use is about 5m X 4m. Screen shots of some the melodies in a population are shown in figure 4. The hidden melody is shown in figure 5. Remember, the member of the population selected to play each musical cycle is the melody from the population that is closest to the hidden melody. We are currently investigating other possible visual representations.

6 Summary and Future Work

In summary, we have developed a system that allows a performer(s) to control an underlying evolutionary process which in turn creates music. We have encoded melodic structure as a genome and have defined a number of genetic operations that can be applied to a population of melodies. We have mapped some of the relationship semantics between two objects to control semantically related operations in the evolutionary cycle. This allows performers to manipulate the musical space by adjusting the process parameters, rather than directly controlling the musical output. This process occurs in real-time and is suitable for performances. At each cycle of the musical loop, the melody closest to a predefined hidden melody is selected. This melody is played concurrently with other melodies in play. The music produced has stochastic variations related to the distance between two objects; the mixing of the population is related to the perceived interference of two objects. The music also has interesting textures and structure.

Here are our future plans:

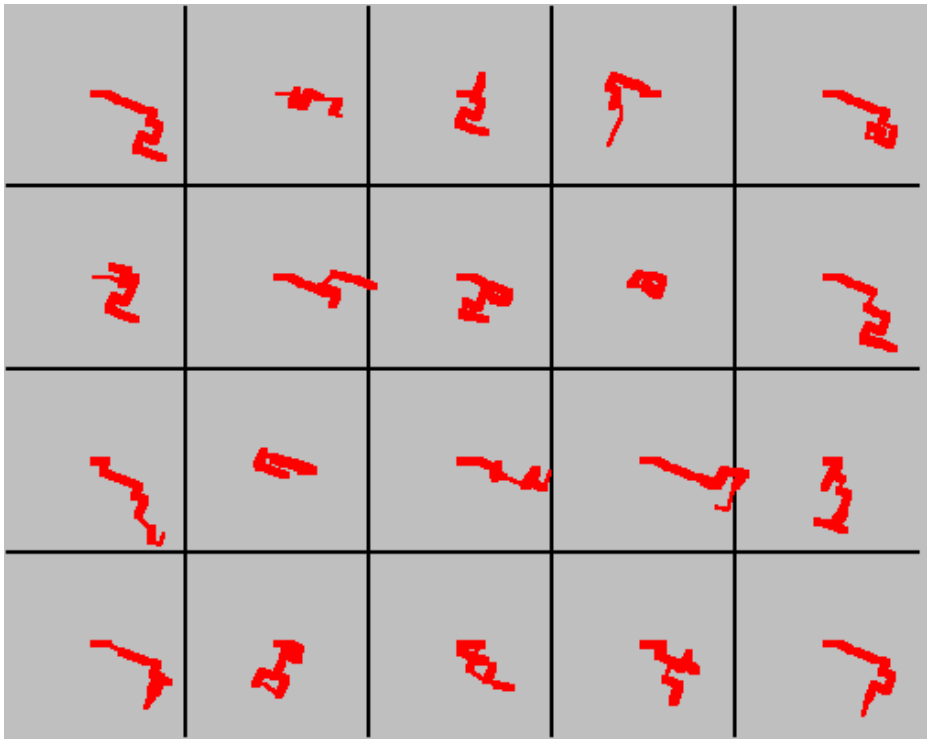

Fig. 4. Screen shots of the stick visualization of various melodies. These melodies were the result of a small amount of mutation (10%) away from the hidden melody for illustration.

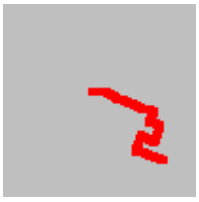

Fig. 5. Screen shots of the stick visualization of a hidden melody that we have used. The closest melody from the melody population in figure 4 would be chosen to play every cycle time.

1. to experiment with alternative visualizations;
2. to allow additional control of MIDI parameters to evolve in the genome, such as program change, MIDI panning, etc.
3. to use a MIDI controller or a keyboard to change the Hidden Melody in real-time;
4. to increase the complexity of a melody by including multiple notes and improve the rhythmic representation allowing irregular subdivision;
5. to increase the types of evolutionary processes available, and
6. to further experiment with the mapping of two and three object relationship semantics to the semantics of evolutionary control.
7. to evaluate the various mappings between the interaction space

Evaluation of the system is complicated by the fact that there are no well defined objective criteria for performance. However, one technique we are developing is to use distraction tasks while people are controlling the music. The more distraction that people can tolerate while controlling the system suggests that the interface may support automatic behavior. In this mode of operation, it should be easier to express oneself. Of course, user surveys may be used in conjunction with this method.

Evolutionary processes are excellent candidates for automatic composition as well as real-time exploration of melodic variation. Further, since a process model underlies the creation of the music the performer only indirectly controls the music. The advantage is that process semantics can be mapped to performer semantics. This facilitates creating consistent cognitive maps between manipulation and navigation in the musical space, thus decreasing cognitive load and enhancing intimacy with the interface.

References

1. J. A. Biles. A genetic algorithm for generating jazz solos. In *Proceedings of the 1994 International Computer Music Conference, (ICMC94)*, pages 131–137, 1994.
2. N. M. Cheung and A. Horner. Group synthesis with genetic algorithms. *Journal of the Audio Engineering Society*, 44(3):pp. 130–147, 1996.
3. Palle Dahlstedt. A mutasynth in parameter space: Controlling sound synthesis by interactive evolution. personal communication, 2001.
4. B. Fogel. Evolutionary computation - toward a new philosophy of machine intelligence. *IEEE Press*, pages 46–47, 1995.
5. D. Horowitz. Generating rhythms with genetic algorithms. In *Proceedings of the 1994 International Computer Music Conference, (ICMC94)*, pages 142–143, 1994.
6. B. L. Jacob. Composing with genetic algorithms. In *Proceedings of the 1994 International Computer Music Conference*, pages pp. 452–455, 1995.
7. G. Johnson. Exploring the sound-space of synthesis algorithms using interactive genetic algorithms in g. a. In *Proceedings of the AISB Workshop on Articial Intelligence and Musical Creativity*, 1999.
8. J. R. Koza, F. H. Bennet III, D. Andre, M. A. Keane, and F. Dunap. Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on Evolutionary Computation*, 1(2), 1997.

9. J. Manzolli, A. Moroni, F. Von Zuben, and R. Gudwin. A evolutionary approach applied to algorithmic composition. In *Proceedings of the VI Brazilian Symposium on Computer Music (Rio de Janerio, Brazil)*, pages 201–210, 1999.
10. J. Manzolli, A. Moroni, F. Von Zuben, and R. Gudwin. An evolutionary approach applied to algorithmic composition. In *Proceedings of the VI Brazilian Symposium on Computer Music*, pages 201–210, 1999.
11. E. R. Miranda. *Composing Music with Computers*. Oxford (UK): Focal Press, 2001.
12. A. Moroni, J. Manzolli, F. Von Zuben, and R. Gudwin. Vox populi: An interactive evolutionary system for algorithmic music composition. *Leonardo Music Journal*, 10:pp. 49–54, 2000.
13. C. Roads. Genetic algorithms as a method for granular synthesis regulation. In *Proceedings of the 1993 International Computer Music Conference*, 1994.
14. R. Rowe. *Interactive Musical Systems*. Cambridge, Massachussetts(USA): The MIT Press, 1993.
15. Ben Schneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (Second Edition)*. Addison-Wesley, 1992.
16. K. Sims. Interactive evolution of equations for procedural models. *The Visual Computer*, 9(9):466–476, 1993.
17. O. Strunk. *Source Readings in Music History*. Vail-Ballou Press, 1950.