1. Introduction

Composing music is a creative process, and like any creative process, it can be described in terms of an aesthetic search through a space of possible structures that satisfy the requirements of that process; in our case, creating interesting music. Compositions tend to exhibit varying degrees of structure, where the composer labors by crafting and, evolving initial ideas into satisfying end products. Richard Dawkins, in The Blind Watchmaker (Dawkins, 1986), describes lucidly how objects of apparent high structure (living organisms) can come about via evolutionary principles rooted in natural selection. He also extends this notion to a form of Universal Darwinism (Plotkin, 1995) whereby the creative generation of intellectual ideas themselves derive from such a process of iterative refinement of competing ideas, or "merges" as he likes to call them.

The success of employing evolutionary techniques in artistic endeavours is well exemplified by the work of William Latham and his Muator and Form Grow systems for evolving 3D computer sculptures [Latham et al., 1992]. His results can only be described as extraordinary, surreal and distinctly organic. Latham observes, "The machine has given me freedom to explore and create complex... forms which previously had not been accessible to me, as they had been beyond my imagination."

Inspired by the work of Latham and Dawkins, GeNoNator is an attempt at transferring these evolutionary concepts to the domain of music making. Its main "evolutionary" mechanism is the Genetic Algorithm (GA) (Goldberg, 1989), which has proven to be a very effective way of blindly seeking out acceptable candidate structures from large search spaces. In GeNoNator, the GA is used to guide the composer through the search space of possible compositions.

2. Background

GeNoNator was originally conceived as a Masters thesis in 1992 [Thywissen, 1993], written in C++ on a SGI. It built off an experimental application called Muator [Thywissen, 1992] - a simple MIDI-based tool for evolving tone-rows using an iterative fractal approach. At the time only Horner [Horner et al., 1991] had detailed the use of GAs in computer assisted composition, specifically their use in thematic bridging.

Since then, their popularity in computer music in general has grown - now found in various applications from instrument design [Horner, 1993], to searching for near-optimum synthesis parameters for switched instruments [Horner, 1995], and other computer assisted composition systems [Jacob, 1995]. For those unfamiliar with GAs and how they are typically used in composition, the following is a brief introduction.

GAs operate on populations of chromosomes (genotypes), each a codified competing "solution" whose genetic content is mapped onto the problem domain via a domain-specific representation scheme (RS). This RS allows candidate genotypes to be decoded into domain specific solution candidates (phenotypes) and then be assigned a relative fitness score via a so-called fitness function. In traditional GA usage, fitness is an objective function that evaluates how closely a given phenotype approximates (or attains) an ideal solution in the problem domain.

Typically, the GA starts by generating an initial random population of chromosomes. This population then undergoes evolution where pairs of chromosomes are picked to exchange genetic material via a process called crossover. Crossover is the mechanism that allows offspring to inherit and combine genetic material from their parents. Fitter chromosomes tend to be selected for crossover over inferior ones in a way that is proportional to chromosomal fitness. Another genetic operator is mutation where chromosomes occasionally can have a randomly selected gene altered, thus adding new genetic material to the gene pool. Through this iterative process of crossover and mutation, less fit genetic material gradually dies and fitter genetic material propagates to subsequent generations. This continues until sufficiently "superior" chromosomes evolve which marks the point of termination of the GA.

In computer music, the challenge is to find a representation scheme which maps the chromosome makeup to musical entities such that music can be "evolved". The objective fitness function is usually reinterpreted as subjective where instead the composer's taste and judgment dictate the relative success rate of competing musical structures represented by the chromosomes. As Biling (1994) observes, this is often the "kettleuck" in a GA-based system as GAs typically operate on relatively large populations of candidate...
chromosomes, each of which must be evaluated by the listener. One way to minimize the population explosion is to concentrate on defining higher level generative processes that are parameterically controlled, it is then these parameters which are evolved by the GA. The generative processes thus define boundaries to the composition's parameter space, within which the GA can evolve interesting parameter combinations. This is certainly the case with Latham's work [1992], where the user is typically given only 9 items to select between at each iteration of evolution, yet each of these nine items already exhibit a high degree of pre-defined structure (as opposed to purely random starting conditions). Other computer music approaches effectively also do this: where the generative processes drive either melodic development [Rederick 1995], rhythm patterns [Horowitz, 1994] or other well-defined structures such as fitting a melody to a pre-selected chord sequence and song structure [Biles, 1994]. The problem with this is that the composer still needs a degree of analytical skill in deriving these generative processes in the first place. However, once in place, anyone can theoretically act as "composer" and evolve music within the predetermined form space.

There is a second approach that is worth briefly mentioning, and which is to somehow model the composers taste itself, thus freeing him/her from having to laboriously judge a large population. One proposed method of doing this is to use a neural network to try to deduce a pattern of taste, trained by observing past user judgments [Thywissen, 1993, Biles, 1994]. Another approach is to assist chromosome fitness evaluation by allowing the user to define rules that, when satisfied, contribute to the chromosome's fitness score [Thywissen, 1993].

3. Evolving Music with GeNotator

GeNotator is an attempt to define a comprehensive framework for musical evolution. Figure 1 gives a high level view of its principal philosophy. In it, we can picture a composition as an amorphous whole of interrelated parts where each plays an important role in the compositions overall structure. These parts may be melody, rhythm, structure, harmony, instrumentations or form. "Blind" composition involves applying the genetic algorithm to aspects within this amorphous whole such that permutations of the composition space can be evolved.

In terms of its place in the tradition of computer assisted composition, GeNotator is best described as a hybrid-algorithmic composition tool which, on the surface, resembles any mainstream MIDI-based compositional GUI environment with piano-roll editing and an "arrangement" editor.

However, to complement this is a framework with GA at its heart, for evolving music. Before going into detail about this framework, it is worth discussing briefly how GeNotator distinguishes between two different types of user/composer.

3.1 "Meta-composer" vs. "Gardener"

GeNotator operates at two distinct interactive levels, which are best described in terms of user sophistication: the meta-composer vs. the gardener (to borrow terminology from Latham). Both, incidentally, could be the same person. The meta-composer tends to have a more analytical understanding about the form and structure of a composition and what he/she wants to achieve, and uses the GA to assist the creative process by generating and refining ideas. Contrast this with the gardener who escapes from analytical thought entirely (i.e., is not permitted to think in terms of structure definition), and instead only needs to know what he/she likes. This is related to the observation that people tend to be far more sophisticated in listening than in creating music ("I don't know anything about music, but I know what I like").

Both types of user have differing interfaces to a given composition. A typical scenario involves the meta-composer "publishing" a compositional structure genotype; the gardener is subsequently a consumer who interactively evolves a published meta-composition into any number of phenotypes - figure 2.

3.2 Internal Organization of GeNotator

GeNotator uses a hierarchical model to represent any given composition. It also has a simple grammar scripting language that can be used to generate instances of hierarchically organised compositions. The smallest compositional element is the phrase, which is defined as a sequence of possibility concurrent note events. Collections of phrases can be organised into systems, and systems arranged as children to other systems to any level of abstraction - figure 3. A piano-roll editor and percussion map editors allow editing of phrases, while a hierarchic "arrange window" allows editing of systems.

A phrase-evolver applies the GA to phrase development, and a system-evolver applies the GA to system development. In this way, the composer can target the GA at different levels within a composition, either breeding new structures, or developing existing ones further within the form space. The GA lives in a...
Figure 3 - Principal architectural components of Geniator.

Additionally, the GA may be applied to a defined musical grammar in order to explore the syntactic permutations permitted by it. The grammar scripting language contains some basic musically useful syntax and can refer to existing phrases and systems as well as user-defined and parametrically controlled event generators. An event generator is an algorithmic process that takes input parameters and generates output in the form of phrases and systems.

3.3 Evolving Phrases

In Geniator, a phrase is a list of notes - each described in terms of a pitch, a relative start time, a duration, and a loudness value. Notes may occur sequentially or concurrently within any given phrase, and a phrase is typically associated with an instrument, although with percussive phrases, each note maps to a different percussive sound in the traditional MIDI manner. Phrases are thus ideally suited for representing melodic, harmonic/choral, and rhythmic structures.

In melodic evolution, the GA can operate on several different aspects of a given melody, each represented by a population of chromosomes. These aspects are: pitch content, rhythmic content and dynamics content - figure 4.

Figure 4 - Independent evolution of different aspects of a phrase.

In this way, the composer can either evolve all three collectively, or just concentrate on one particular aspect of the melody. Geniator allows evolution to be applied to previously recorded material by using the source to generate a population of mutated instances which are then the basis for further evolution. It can also take several "favorable" parent chromosomes and splice them to generate hybrids.

If the composer is operating with large populations, fitness rules can be activated to augment the subjective fitness function, whereby chromosomes automatically accumulate points if they fulfill a particular active rule. The composer can then filter out low scoring chromosomes and concentrate only on evaluating the high scoring ones. Typical pre-defined rules specify desired keyscales/modes, note durations, note density and interval constraints.

Evolving harmonic/choral structures is done in a similar way, although here phrases comprise lists of concurrent note groupings, each representing a chord. Chord progressions can be evolved, as well as the pitch content of individual chords. Again, fitness rules may be activated where, for example, acceptable chord sets can be specified.

Rhythmic/percussive evolution differs from both melodic and harmonic/choral evolution in that the GA operates on clusters of possibly concurrent note events - each a separate pattern assigned to a particular percussive instrument lasting one measure - figure 5.

Figure 5 - A manoeuvre of concurrent patterns, one for each percussive sound.

For any given rhythmic/percussive phrase, the GA evolves a range of differing parameters, from individual pattern populations, to populations of pattern groupings and instrument combinations. Fitness rules can be activated to promote chromosomes that fulfill particular required features such as choice of percussive instrument family, density level, amount of syncopation and use of down beat and general dynamics. This is similar in approach to Hecox's [1994] use of GAs to evolve rhythmic structures, in that it serves as a straightforward method of encoding rudimentary rhythmic knowledge into the fitness function. It is also a logical extension of how Geniator augments melodic and harmonic/choral evolution with rule-based fitness evaluation.

3.4 Evolving Systems

System evolution is fundamentally different from phrase evolution in that the chromosome genes encode the activation of a selection of system operators. System operators are well-defined processes that manipulate the contents of the system being evolved. Geniator currently has five pre-defined operators: transpose, retrograde, invert, augmented/minimize, and growth. The first four are interpreted with their traditional meaning; the last allows systems to grow new content by either copying mutated versions of existing systems or simply creating a fractally derived structure of empty systems which the user can populate with phrases.

Thywissen
3.5 Meta-Composition.

As already indicated, "meta-composing" sets the scene for the gardening of music. The meta-composer will typically use a combination of pre-recorded and evolved phrases and systems, together with GeNotator's grammar scripting language and parametrically controlled event generators to formulate a meta-composition with potential that is bounded by the structure defined by this combination of the grammar, event generators, and phrase content. This bounding is important in order to constrain the composition form space. GeNotator is flexible in this regard since the degree of bounded structure can either be minimal, or highly constrained, depending on what the composer wants to achieve.

Given a grammar definition, GeNotator automatically generates a chromosomal structure that maps all the choice routes in the grammar to individual genes. Added to this is a composer-aided, yet built-in, experimental parameter and any pre-defined systems/phrases. Collectively, this packaging is the genotype structure definition (Figure 2) and is "publishable" for interactive consumption.

3.6 Gardening Music.

Gardening music involves taking a genotype structure definition and feeding it into GeNotator's gardening interface in order to interactively search through its form space without regard for internal structure. Again, the GA plays an important role in this search and the user has the option to evolve the composition at differing levels of abstraction. At the highest level, GeNotator can be asked to generate a small number of phenotypes These may vary quite widely since every aspect of the genotype is subject to genetic manipulation by the GA. However, this is often the desired effect in order to get maximum initial variability. Here, GeNotator also allows the user to select two or more parent phenotypes for cross-fertilization.

For more interactive control, the user may ask GeNotator to list individual phrases, event generators, and system/phrases in the genotype structure definition, and selectively evolve them as the meta-composer would. So if a particular melody or rhythmic structure is of interest, it can be targeted and selectively evolved.

4. Conclusions and Future Work

In so far that GeNotator provides a comprehensive approach to musical evolution within a hybrid compositional environment, there is much yet to explore. In terms of musical output, GeNotator can certainly evolve interesting music, but it is often a matter of balancing good starting conditions with a degree of form space bounding. Grammars help a lot here, as do the parametrically controlled event generators, which is really GeNotators key to allowing end-users "garden" music. The meta-composer still needs a degree of analytical epitude in order to capture an interesting genotype structure definition, and it is perhaps here that initial enhancements should be focused. One avenue of interest is to provide automated grammar generation from existing music, or to apply the GA to the evolution of grammars themselves. Other planned enhancements include allowing the user to extend the built-in fitness rules via a simple, ad-hoc definition language. The same goes for the event generators and system operators, which should ideally also be user-extensible (currently they are hard-coded).

Finally, GeNotator's gardening interface could be separated out into a standalone application. With the growth of the Internet, combined with the fact that many PCs now come with fairly sophisticated sound cards with built-in MIDI, a GeNotator gardening interface could manifest itself as a Web browser plug-in. This would allow users across the net to exchange, combine and mutate favourite musical phenotypes and genotypes within GeNotators framework.

References