2006

Genetic music

Ryan Becker

MS Project Proposal

Genetic Music

Ryan Becker
(rdb9723@cs.rit.edu)

Department of Computer Science
Rochester Institute of Technology
May 17, 2005

Chair

Prof. Joe Geigel _____

Reader

Prof. John A. Biles _____

Graduate Coordinator

Prof. Hans-Peter Bischof _____

Table of Contents

Abstract

Describes the application of several Genetic Algorithm (GA) and Genetic Programming (GP) algorithms to algorithmic music composition (inspired by Karl Sims' work with images). Each of these algorithms fall under the category of Interactive Evolutionary Computing (IEC), which has been effective at solving a variety of problems sharing many of the same difficulties facing algorithmic music composition. An application implementing these IEC algorithms is described, loosely modeled after Karl Sims' work, which includes a user interface and general software framework supporting the application of any IEC algorithm to algorithmic music composition. The software framework makes it very easy to experiment with different algorithms or data structures, which is central to this project. Various IEC algorithms will be implemented and their effectiveness compared.

Introduction

Algorithmic music is an incredibly active area of research, which spans the fields of computer science, art, science, and philosophy. It is not a new idea, but has actually been around for as long as computers, if not longer.

History of Algorithmic Music

Algorithmic music is indeed an old concept; one could argue that even the musical dice games of Mozart [11] were really an early form of algorithmic music, for example. A pair of dice were tossed to determine a sequence of precomposed measures, which together formed a complete piece. There are many examples of similar types of musical composition techniques throughout history that could be classified as algorithmic. One even earlier technique for music composition, called “soggetto cavato” [11] and developed in the 1400s, involves mapping letters of the alphabet to notes in order to produce unique melodies. This often took the form of vowel-mapping and was used with peoples' names as well as the names of cities and towns.

These techniques are quite simplistic in nature, but are still arguably the beginnings of algorithmic music and demonstrate that people have long been fascinated with music composition that contains some type of “magical” generative component that comes from something outside themselves, either random or otherwise. This fascination continues to this day and is even more widespread, given the powerful computers that we now have at our disposal and the interesting observation that programmers often have an interest in music as well. I believe this has yet to be demonstrated by any scientific study or survey, but it seems to be true more often than not. There is also a deeper philosophical question that I believe helps to fuel interest in algorithmic music, which gets back to the essence of creativity and aesthetics.

Statement of the Problem

Perhaps one of the most difficult tasks of any algorithmic music system is producing something that has aesthetic appeal, which is a critical component in music – more than a nice ideal. Much of past research in algorithmic music seems to have focused on generating musical complexity and neglected the importance of simply sounding good. It is fairly easy to generate something that is musically complex, but generating something compelling, that tells a story, conveys an emotion, or communicates some idea, is significantly more difficult. It is this vague, subjective aspect of creativity that plagues any system that aims to generate some form of “art” or perform any task that we would generally think of as creative.

Analysis

As has been mentioned, for many years, and especially since the dawn of computing, algorithmic music has been explored in great depth. Yet, many algorithms are unsatisfactory, either because they simply don't generate good music, with aesthetic appeal, or because the musical solution space has been so limited that every generated piece of music sounds the same.

Past Approaches to Algorithmic Music

When reviewing the literature pertaining to algorithmic music we can identify two basic approaches - “derivative” and “generative.” In general, the derivative approach tends to create music that all sounds the same, while the generative approach tends to generate music that just doesn't sounds good at all.

Derivative

In the early days of computer music, many algorithms were designed to be seeded with existing pieces of music, then restructure the ideas found in the model pieces into a new composition [1]. These algorithms were purely statistical in nature and used Markov analysis, which simply creates tables representing the probability of a note falling at any given place in a piece. These types of algorithms can sometimes result in interesting variations on existing pieces of music, but results sound much like the original piece(s) of music used for content.

Another application of algorithmic music, that falls loosely under the derivative approach involves musical interpretation ([17] provides an example of this). The challenge is to create a program which can play expressively, like a human. The problem here is one of performance, as opposed to composition.

Generative

Also popular during the beginnings of research into computer music were algorithms which were solely generative, with no human element. This is based on the assumption that computers can generate completely original music. This is by far the most ambitious form that algorithmic music takes. It is possible to use Markov chains in a purely generative way, by skipping the analysis step and simply establishing some probabilities by filling in the Markov tables, which is precisely what Hiller did in 1957 when he composed what is often considered the first original piece of music composed with a computer, the “Illiac Suite for String Quartet” [1].

For a long time, the focus of generative music was on creating original musical elements, with little thought given to how the resulting music was actually perceived. Simply the idea that a computer could generate a complex piece of “music” by itself was a novel idea and fascinated researchers. Not only was the human response to music neglected, but nearly everything known about traditional music composition and music theory was discarded. This was a radical difference from the derivative crowd, who based everything on existing music and left very little room for deviation.

Need For A Human Component

Based on the results of many years of research in computer music, it's apparent that we are incapable of coaxing computers to generate compelling music aside from human intervention [24], [18], [13].

> “No algorithm can generate meaningful music from scratch. Knowledge of the desired musical style, or a set of artificial rules, or a training set of existing music is at least needed” [18].

As soon as a human element is introduced, whether it be a single musical idea, single piece, large number of pieces, or any other human element, the chances of success dramatically improve. Still, it is difficult for a computer generated piece of music to convey anything that the provided music elements didn't already contain, and thus, it is difficult to arrive at something “fresh” or “original.”

High Dimensional Solution Space

Another of the difficulties inherent in music is the high dimensional solution space. There are simply too many variables in music for a straight forward algorithm to handle in a reasonable amount of time, if at all, which is why many of the successful algorithms limit the solution space. These variables include time, pitch, duration, loudness, meter, and tempo, among others. These variables create an infinite number of possible solutions, or pieces of music, and this high dimensional solution space is both difficult to describe and navigate.

Hypothesis

With this problem in mind, let's suppose there *were* a system that could generate interesting original music. What might it look like? The two most difficult problems were the high dimensional solution space and the need for a human component. I believe Interactive Evolutionary Computing (IEC) is the most promising technique and has the potential to solve both of these problems, but before examining how IEC may be applied to the problem at hand, we must define Evolutionary Computing (EC) and describe its various manifestations.

Overview of Evolutionary Computing (EC)

EC describes the set of algorithms and techniques based on the theory of biological evolution. It is incredibly effective at finding solutions to problems with a high dimension problem space. It has been used to solve, or at least approximate solutions, to many NP problems, including the traveling salesman problem, among others. The idea has been around since the 1950s [2], but really solidified when Holland developed the Genetic Algorithm (GA) in 1962 [16]. The GA has become probably the most popular and successful of this class of algorithms. The GA's success is due, in a large part, to the publication of the classic Goldberg text [15], which made GAs palatable to the common man (or at least the common engineer), explaining in concrete terms how the GA may be applied to a variety of real-world problems.

Genetic Algorithms (GA)

In a GA, a “population” of candidate solutions is “evolved” toward better and better solutions to a problem with hopes that each successive generation will yield more “fit” solutions. The process of “evolution” involves “mutation” and “crossover” (ie. mating) operations, designed to mimic the phenomena described by the theory of biological evolution. A critical component of the GA is the fitness measure, required in order to model Darwin's “survival of the fittest.” From each generation, the most “fit” candidate solutions, or “individuals,” have a greater likelihood of persisting and reproducing, to form the next generation. One other critical component of the GA is the encoding of the

solution, and the definition of the mutation and crossover operations are necessarily dependent on this encoding.

Genetic Programming (GP) [20] is a compelling variation of the GA, in which computer programs are the individuals being evolved, and fitness is determined by how well the computer program performs the desired task. This has been shown to be highly effective at solving a surprisingly wide variety of programming problems.

Interactive Evolutionary Computing (IEC)

EC seems to be the perfect solution to the high dimensional solution space problem, but what about the critical human component? IEC is a special branch of EC which includes an interactive human element, essentially providing feedback throughout the generative process. Based on our analysis of the problem, this type of algorithm would address the two most critical issues.

IEC has proven to be remarkably effective in creative or artistic processes. Karl Sims perhaps did the most to forward this method when he applied it to computer graphics [22]. He applied it to several problems, including animation, but the technique that remains most popular today, perhaps for its effectiveness and straight forward implementation, is Genetic Images. Genetic Images provides an excellent example of IEC.

The basic idea of Genetic Images is quite simple: a set of images are presented to the human and one or more favorites are chosen. These select images are then used as the basis for the next generation. Mutation and crossover operations – from GAs – are then applied to produce a new set of images. This process repeats until the algorithm produces an image the user finds appealing and decides to stop. This is essentially a GA where the human becomes the fitness function.

For something as subjective as music, there is no way to produce a standard analytical fitness function capable of effectively representing a person's individual aesthetics. Even if we could model something as complex as a person's aesthetics, it would only reflect one particular individual. What about other people, or what about when a person's aesthetics change? Aesthetics are dynamic and can change as frequently as a person's mood. It is in domains such as this that IEC works best and thus Karl Sims' successful Genetic Images provides an excellent starting point and inspiration for a Genetic Music application.

Existing Research

Applying IEC to music is not a new idea; several people have approached Genetic Music, from various angles. Both standard GA [5], [9], [21], [26] and GP [19] techniques have been used and some projects have even used both together [25]. There is much to learn from what has and hasn't worked in using IEC with music and it would be foolish not to take into consideration both the mistakes and advances of past research.

Perhaps the most serious difficulty with Genetic Music is the universally acknowledged human fatigue problem. This is a potential risk for all IEC algorithms, because the population size must be small enough for a human to evaluate in a reasonable amount of time – most Genetic Image applications use a population of around 9 individuals, for

example. The problem is that when the population is reduced to numbers that small, the number of generations required to generate a satisfactory solution almost always dramatically increases. It's simply not possible for a human to evaluate millions of generations, which is not an uncommon number in many applications of traditional GAs.

Because the human fatigue problem is so critical, research in Genetic Music either directly or indirectly focuses on solving this issue. In order to lessen human fatigue, the total time required to find a good solution must be reduced. This means that either the population must converge in fewer iterations or the efficiency of the interactive human component must be improved. In other words, either the algorithm itself or the user interface (UI) must be made more efficient. It helps to keep this in mind when examining what others have done, because it is what motivates nearly all existing research.

In an attempt to solve the human fatigue problem, several people have tried to model trends in user selection with a Neural Network (NN), with mostly inconclusive results. [9], [19], and [25] describe attempts to do this. At best, the trained NN is sometimes able to produce results almost as well as the user, but is unable to consistently do so [19]. In [25] the NN is more consistently able to produce fairly good results, greatly reducing time to converge, but the results lack diversity. Modeling human aesthetics is a very difficult problem and existing attempts seem to oversimplify, as [9] explains: "...humans listen to music in complex and subtle ways that are not captured well by simple statistical models."

Some people have focused on a particular part of music, while others have tried to do everything, including rhythm, melody, harmony, and various instruments. [5] aims to generate melodies. [26] and [12] set out to generate music consisting of multiple instruments and parts. [25] focuses on generation of rhythms. It is decidedly more difficult to handle multiple instruments and parts of music. The solution space becomes significantly larger with each added part, so often more intelligence is required for the algorithm to be successful. It is important to remember this when evaluating various studies.

The most successful algorithms to date find creative ways to narrow the solution space. For example, Al Biles' GenJam [5] focuses on jazz, and does it very well. Of course, GenJam is not very useful for less tonal music, or other styles, but this is not a problem because it was never intended to do anything but jazz. GenJam restricts the solution space in several key ways. First, the chord progression for a piece is given, and melodic material is mapped onto tried and true jazz "scales." Since the beginning of jazz, musicians have been improvising solos based on the particular "scale" that goes with each type of chord. This mapping ensures that a "wrong note" is never played, and because it is based on techniques used by every jazz musician, it sounds right (and even hip sometimes). One other way the solution space is reduced is by limiting the number of unique notes to 14. This allows music events to be represented as 4 bit numbers, because 0 and 15 represent a rest and a hold, respectively.

The UI is still a difficult problem, and there are apparently no easy solutions. GenJam allows the user to provide positive and negative feedback on measures and phrases (a phrase consists of 4 measures) as they are played back in full, one at a time, during the training phase. There are 48 unique phrases, and 64 unique measures with GenJam. Of course, because there are no wrong notes, there are likely to be half decent individuals in

even the first generation. This is an example where, because of some intelligence given to the algorithm, the number of generations will be significantly reduced. Because of this, it doesn't really matter if the user needs to listen to every single individual, because fit individuals are likely to appear fairly quickly. More conventional approaches adopt a UI similar to that used for Genetic Images, displaying some graphical representation, often a score, and allowing individuals to be played on demand. A favorite may then be selected as with Genetic Images (although it is more common to allow multiple favorites with Genetic Music). There is much room for improving the UI in Genetic Music applications.

Most, if not all, projects include a significant amount of intelligence in the algorithm, often ensuring the music complies with "good music theory," endowing the algorithm with some idea of what is "musically interesting," requiring a specific meter, and limiting the range of the melody, among others. This is done to lessen the fatigue problem and ensure that the algorithm converges on something in a reasonable amount of time. Music has many more dimensions than images and pleasing music is arguably a narrower category than pleasing images (a fairly general random approach works quite well with images). It is clear that developing a good balance in the amount of "smarts" is critical to the success of any Genetic Music algorithm.

Synthesis

There are a variety of questions that music raises, as a application domain for IEC: How is each individual represented? Can music be graphically rendered as a score or as something else? When will individuals be played and how much of each individual will be played? These questions are probably dependent on the length of each individual and the number of unique individuals which exist in each generation. Is the user an amateur musician, professional, or non-musician? How much effort can be expected from the user? What level of smarts should the GA be endowed with? On one extreme is a completely random GA and on the other extreme is a GA with mutation and crossover operations based completely on music theory. In terms of scope, what are the variables in music and is it feasible to let the GA adjust them all? Finally, should Genetic Programming be used, or a standard GA (ie. should we evolve music directly, or introduce a level of abstraction and evolve a music generation program, in the same way Genetic Images works)?

These are a few of the questions that motivate my project, some of which I hope to explore and hopefully answer, in light of the patterns and trends in existing research. My goal is to implement three different approaches to Genetic Music. The first two will be fairly standard GA and GP approaches, but the final algorithm, while a GP approach, will be something more original, based on mathematical curves. It is almost completely based on the Genetic Images algorithm, but has never been applied quite so directly to Genetic Music. I will implement these GP and GA approaches to Genetic Music, drawing on ideas from Genetic Images and existing research in Genetic Music. In this way, I will develop a working knowledge of the field of Genetic Music, by implementing the principle algorithms and techniques employed (in addition to my own idea), gaining a practical understanding of what works, what doesn't work, and why. It should be noted that there are no real standard approaches to Genetic Music, thus even the 2 algorithms that are not entirely original will include a good deal of my own ideas. There are some

clear general trends, but I will be working out many of the details myself.

While my ideas will likely change over the course of this project, it helps to begin with some initial ideas concerning the overall software framework, musical parameter set, genome definition for the GA, solution generation (from phenotype to music), function set for GP, and UI.

Software Framework: A software framework will be built, taking full advantage of polymorphism to support any conceivable evolutionary algorithm. It should be incredibly easy implement a new algorithm. The basic control of the GA or GP should be abstracted out so that each new algorithm can focus on the algorithm itself. Consider a population class that contains instances of polymorphic individual classes, implementing crossover and mutation functions. If the goal was to only implement a single algorithm, perhaps it wouldn't matter so much for the software framework to facilitate the addition of new algorithms, but because the goal is to implement at least 3 different algorithms, it's important for the framework to be easily extensible.

GA/GP Parameter Set: These might include such things as pitch, velocity (loudness), note duration, meter, tempo, key, and mode (major, minor, etc). I will begin with pitch and note duration, hopefully incorporating velocity later, and possibly others, as time permits.

GA Genome Definition: I will begin with simply a string of notes, containing pitch, duration, velocity and whatever other parameters are to be modified. I might also consider some type of tree data structure – perhaps a 4 layer tree containing song, phrase, measure, and event levels, similar to GenJam.

Solution Generation: Using MIDI and the useful Java Sound APIs, note descriptions may be used to construct a MIDI sequence and rendered as audio.

GP Program Representation: I will implement at least 2 different GP approaches. The first will evolve a tree representing a function which returns musical phrases. Leaves will be actual notes and branches will be functions operating on musical phrases (a phrase may consist of only 1 note). For this approach, the function set must operate on musical notes, or at least lists. It is not purely mathematical, as with the standard Genetic Image algorithm. The other GP approach is based much more closely on Genetic Images, using mathematical functions (probably taken directly from those used for Genetic Images) of time to represent pitch, velocity, and note frequency (duration). Because these functions will be naturally continuous, function values will be rounded in order to map onto discrete pitches, velocities, and durations. One way to think about this approach, is that pitch, velocity, and duration are colors.

GP Musical Function Set: These might include the following, for example:

- append(phrase1, phrase2)
- interleave(phrase1, phrase2)
- addDegree(phrase1, phrase2)
- subtractDegree(phrase1, phrase2)
- shorten(phrase)

- lengthen(phrase)
- palindrome(phrase)
- repeat(phrase)
- reverse(phrase)
- rotate(phrase, n)
- addRelativeNote(phrase, n)
- shift(phrase, n)
- major(phrase)
- harmonicMinor(phrase)
- blues(phrase)
- wholeTone(phrase)
- pentatonic(phrase)
- tiePitches(phrase)
- transpose(phrase, n)
- restNoise(n)
- noteNoise(n)
- lowThreshold(phrase, n)
- highThreshold(phrase, n)

UI: The UI will be loosely based on Genetic Images, graphically representing each piece of music as a piano roll (dashes for notes, the length representing note duration, and the vertical location representing pitch, perhaps color representing velocity). Each piece could be played on demand and stopped whenever desired, by another action – mutation, crossover, or playing another piece, for example. In this way, the user does not necessarily need to listen to each individual in a given population, hopefully reducing the time to evaluate and determine the fittest individuals.

Project Plan

Following is my project plan, with the estimated week of completion for each goal:

- Implement Genetic Images, in order to develop a working knowledge of IEC and develop a GP framework. (week 4)
- Implement interactive GA technique. (week 5)
- Implement interactive GP technique. (week 6)
- Compare each technique. (week 6)
- Explore various degrees and types of "smarts," with the goal of having as little intelligence as possible, while still yielding good results in a reasonable amount of time. In general, the type and amount of intelligence given to the GA or GP is what

differentiates the many approaches to genetic music. (week 6)

- Explore mappings between music and graphics (and maybe other mediums) with the goal of possibly deriving alternative rendering techniques (of course, the obvious way to render music is as sound or as a score). This borders on visualization, which is an interesting problem domain in its own right, but the goal here is to improve the UI, in hopes of partially solving the human fatigue problem. (week 7)
- Possibly combine GA and GP in a hybrid approach. (?)

The project plan is subject to change, based on how the project unfolds. If a particular technique, or especially a new idea, proves interesting and seems promising, I will explore it further and perhaps drop several goals. While I may drop some goals, I still intend to study the key techniques that others have implemented, particularly the GA and GP approaches.

When my project is finished, I hope to demonstrate a solid understanding of existing approaches to Genetic Music and be able to compare and contrast them, based on my own implementation of the principle techniques. Also, in the course of working on this project, I have hopes of developing a successful original approach to Genetic Music.

Bibliography

1: Assayag, Gérard. Computer Assisted Composition Today. 1st SYMPOSIUM ON MUSIC AND COMPUTERS. "Applications on Contemporary Music Creation, Esthetic and Technical aspects". CORFU. October 1998.
2: Back, Thomas, Hammel, Ulrich, and Schwefel, Hans-Paul. Evolutionary Computation: Comments on the History and Current State. IEEE Transactions On Evolutionary Computation, Vol. 1, No. 1. 3-17. April 1997.
3: Becker, Ryan. Algorithmic Music: An overview of past research and a look to the future. Research Topic for Virtual Theatre (Spring 2004), Department of Computer Science, Rochester Institute of Technology. May 2004.
4: Becker, Ryan. Interactive Evolutionary Computation: With a focus on Genetic Music. Research Topic for Computer Animation: Algorithms & Techniques (Winter 2004), Department of Computer Science, Rochester Institute of Technology. February 2005.
5: Biles, John A.. GenJam: A Genetic Algorithm for Generating Jazz Solos. Proceedings of the 1994 International Computer Music Conference, ICMA, San Francisco. 1994.
6: Biles, John A.. Autonomous GenJam: Eliminating the Fitness Bottleneck by Eliminating Fitness. Proceedings of the GECCO-2001 Workshop on Non-routine Design with Evolutionary Systems.. 2001.
7: Biles, John A.. GenJam in Transition: from Genetic Jammer to Generative Jammer. Generative Art. 2002.
8: Biles, John A.. GenJam: Evolutionary Computation Gets a Gig. Proceedings of the 2002 Conference for Information Technology Curriculum, Rochester, New York, Society for Information Technology Education. September 2002.
9: Biles, John A., Anderson, Peter G., and Loggi, Laura W.. Neural Network Fitness Functions for a Musical IGA. Proceedings of the International ICSC Symposium on Intelligent Industrial Automation and Soft Computing. 1996.
10: Burns, Kristine H.. Algorithmic Composition. http://eamusic.dartmouth.edu/~wowem/hardware/algorithmdefinition.html. February 2004.
11: Burns, Kristine H.. History of electronic and computer music including automatic instruments and compositional machines. http://music.dartmouth.edu/~wowem/electronmedia/music/eamhistory.html. February 2004.
12: Felice, Fabio De, Abbattista, Fabio, and Scagliola, Francesco. GenOrchestra: An Interactive Evolutionary Agent for Musical Composition. Generative Art. 2002.
13: Garnett, Guy E.. The Aesthetics of Interactive Computer Music. Computer Music Journal, Vol. 25, No. 1. 21-33. March 2001.
14: Gena, Peter. Lejaren Hiller (1924-1994). http://www.artic.edu/~pgena/lhobit.html. February 1994.
15: Goldberg, David E.. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA. 1989.
16: Holland, John H.. Outline for a Logical Theory of Adaptive Systems. Journal of the ACM (JACM), Vol. 9, Issue 3. 297-314. July 1962.
17: Hudak, Paul and Berger, Jonathan. A Model of Performance, Interaction, and Improvisation. Proceedings of International Computer Music Conference. Computer Music Association. 1995.

18: Järveläinen, Hanna. Algorithmic Musical Composition. Seminar on content creation, Telecommunications software and multimedia laboratory, Helsinki University of Technology. April 2000.
19: Johanson, Brad and Poli, Riccardo. GP-Music: An Interactive Genetic Programming System for Music Generation with Automated Fitness Raters. Genetic Programming 1998: Proceedings of the Third Annual Conference. 181-186. 1998.
20: Koza, John R.. Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems. Technical Report STAN-CS-90-1314, Stanford University Computer Science Department. June 1990.
21: Moroni, Artemis, Manzolli, Jônatas, Zuben, Fernando Von, and Gudwin, Ricardo. Vox Populi: An Interactive Evolutionary System for Algorithmic Music Composition. Leonardo Music Journal, Vol. 10. 49-54. 2000.
22: Sims, Karl. Artificial Evolution for Computer Graphics. Computer Graphics, Vol. 25, No. 4. 319-328. July 1991.
23: Takagi, Hideyuki. Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation. Proceedings of the IEEE, Vol. 89, No. 9. 1275-1296. September 2001.
24: Todd, Peter M. and Werner, Gregory M.. Frankensteinian methods for evolutionary music composition. Musical Networks: Parallel distributed perception and performance, N. Griffith and P. Todd, eds. Cambridge, MA: MIT Press. 1998.
25: Tokui, Nao and Iba, Hitoshi. Music Composition by Means of Interactive GA and GP. Proceedings of the 2001 IEEE Systems, Man and Cybernetics Conference. 2001.
26: Unemi, Tatsuo and Senda, Manabu. A New Music Tool for Composition and Play Based on Simulated Breeding. Proceedings of Second Iteration. 100-109. 2001.