Implications of Ad Hoc Artificial Intelligence in Music

Evan X. Merz
San Jose State University
Department of Computer Science
1 Washington Square
San Jose, CA. 95192.
evan.merz@sjsu.edu

Abstract

This paper is an examination of several well-known applications of artificial intelligence in music generation. The algorithms in EMI, GenJam, WolframTones, and Swarm Music are examined in pursuit of ad hoc modifications. Based on these programs, it is clear that ad hoc modifications occur in most algorithmic music programs. We must keep this in mind when generalizing about computational creativity based on these programs. Ad hoc algorithms model a specific task, rather than a general creative algorithm. The musical metacreation discourse could benefit from the skepticism of the procedural content practitioners at AIIDE.

Introduction

At MUME 2013, the group discussed whether it would be possible to recreate David Cope's EMI. Attendees pointed out the controversial position of Cope's work in the musical metacreation discourse due to the ad hoc elements in his program. Al Biles also explained how his live performance version of GenJam differs from a traditional genetic algorithm. Some attendees argued that the ad hoc elements of his program meant the end result wasn't a genetic algorithm at all. These two discussions resonated with my own work using artificial intelligence in music. As musicians, we usually place the creation of an artifact above algorithmic purity. In the scruffy versus neat dichotomy of the larger artificial intelligence community, this makes most of us scruffies (Nilsson 2010, 334). In this article I will review the ad hoc properties of several applications of artificial intelligence to music generation. Understanding the ad hoc elements in these algorithms is important for musicology and to temper the rhetoric about musical metacreation.

What is Ad Hoc?

WordNet lists three definitions for ad hoc.

(adj) often improvised or impromptu

(adj) for or concerned with one specific purpose

(adv) for one specific case

The definitions that apply in this paper are the latter two. This paper is not concerned with aleatoric or improvisatory methods in software, but with algorithm changes that are not generalizable to other settings. How are ad hoc methods used in music software, and what do they mean for the fields of musical metacreation and computational creativity? In this paper I considered three aspects to decide if a change is an ad hoc modification or not.

First, some ad hoc methods are required for software to be music software. A C++ library that simulates swarm intelligence isn't a music tool. There must be a baseline translation from non-musical information to musical information. The underlying space in which the algorithm operates must be converted to a music space in some way. This type of non-generalizable change is unavoidable any time a general algorithm is applied to a specific domain.

Second, ad hoc methodology is a gray area. Some modifications may be more or less ad hoc. Some changes may be ad hoc in one context while they may not be ad hoc in another context. Consider a neural network that only examines pitches in the C-Major scale. This is an ad hoc limitation in the case of all music, but a natural side effect of the input when considering only nursery rhymes.

Third, there is no single definition of a paradigmatic algorithm. There are many specialized versions of most successful algorithms. If an algorithm is ad hoc, then it is not generalizable outside of specific cases. Simulated annealing, for example, could be considered a modification

of greedy random search that avoids local minima. However, since simulated annealing does not apply limitations on the domain or context of application, it cannot be considered an ad hoc modification of greedy random search. If simulated annealing was modified so that the cooling factor was based on the read position in a piece of music, then that would be considered an ad hoc modification in the context of this paper.

In this paper, ad hoc changes are domain dependent modifications of the associated paradigm of artificial intelligence, as defined by texts such as the Floreano and Nierhaus. This paper focuses on specific ad hoc modifications in three well-known algorithmic composition systems, as well as my own work.

EMI

In *Computer Models of Creativity* (CMMC), David Cope explains his Emmy software, and spends a chapter explaining his earlier creation, EMI. Both programs are based on Cope's version of an augmented transition network, which he calls an association network. Cope's association networks are complete, weighted, directed graphs.

> "Association networks are initially empty databases in which a user's input is placed, and in which all discrete entries of that input are connected to all other discrete entries. Networks react to input by placing each discrete entry into a node and assigning weights to each of the connections between this entry's node and all of the other nodes in the network. Association networks assign weights in both directions on a connection; that is, a weight from node X to node Y, as well as a weight from node Y to node X. These weights initially derive from proximity and similarity" (Cope 2005, 274)

Cope's association networks are not passive. Each vertex can incorporate a bit of code that it can run in certain situations. For this reason, Gerhard Nierhaus uses the more precise term augmented transition network to describe Cope's data structure.

> "In an augmented transition network (ATN), the TN is extended in a way that allows specific instructions, conditional jumps or also whole sub-programs to be assigned to the edges." (Nierhaus 2010, 122)

In EMI, Cope uses association networks to implement musical recombinancy. Recombinancy is a technique whereby parts and patterns discovered in a piece are shuffled and recombined to form a new composition. In the case of EMI, many pieces are autonomously broken down, analyzed, and recombined according to the information derived from the analysis. Pieces are broken down into patterns at several layers of specificity and generality. He calls these patterns signatures, earmarks, and unifications (Cope 2001, 109). Signatures are patterns that reoccur in a single piece. Earmarks are patterns that indicate the form of a piece. Unifications are repeated patterns that hold a single piece together. Cope sees this layered recombination process as an analogy for the creative process in humans.

Cope's data structures are virtually identical to basic Markov models (Nierhaus 2010, 68) with only three exceptions. First, the edges are weighted, rather than probabilistic. This difference is significant because Cope tries to avoid decision-making based on random selection (Cope 2001, 129). Second, there are many augmented transition networks used in EMI. There is one for pitches, one for phrases, one for signatures, one for earmarks, and so on. Cope is never specific about how these models are combined in the music generation step. It's clear that the process is iterative, but otherwise we are left in the dark.

The major ad hoc modification in EMI is the methods that are triggered when specific transitions occur. The program can trigger subroutines when specific transitions in the network are used. These subroutines may trigger other networks, or may trigger music specific code. Without looking at the code itself, it's impossible to list all the methods, but in *Virtual Music* Douglas Hofstadter indicates that some of the code is domain specific.

> "Voice-hooking would be the requirement that *the initial note of the melodic line of fragment f2 should coincide with the next melodic note to which fragment f1 led in its original context… texture-matching...* is basically the idea that *the notes in a chord can be moved up or down pitchwise by full octaves and can be spread out time-wise so as to match some preexistent local pattern*" (Hofstadter 2001, 45)

By employing logic specific to notated music, Cope encodes musical knowledge into the algorithm itself in an ad hoc way.

GenJam

GenJam evolves phrases of music that are combined to form a jazz solo over a given chart. GenJam works in three modes, learning, breeding, and demo. These three modes break up the genetic algorithm into three parts.

In learning mode, GenJam works with a human mentor to discover good measures of music and combine them into phrases. In learning mode, measures and phrases of music are probabilistically generated based on the chord progression for the piece, and scales that are usually acceptable over a given chord. Each snippet is rated by the human mentor, who acts as a fitness function. The

measures are not transformed at all in learning mode, they are simply rated.

In breeding mode, a population of 48 measures is selected from the rated measures. The lowest-rated half of the measures are discarded. Then 24 new measures are generated by using genetic modifiers on the remaining measures. Unlike most genetic algorithms, GenJam employs a large set of genetic modifiers, including the traditional mutation and crossover, as well as several genetic modifiers that are based on musical transformations. The latter category includes reverse, invert, transpose, and sort.

> "In an effort to accelerate learning by creating not just new, but better offspring, these 'musically meaningful mutation' operators violate conventional GA wisdom that genetic operators should be 'dumb' with respect to the structures they alter." (Biles, 1994, 135)

Demo mode is the performance mode. In demo mode, GenJam simply selects and combines phrases that have been deemed fit in breeding mode.

GenJam is not a standard genetic algorithm in several ways. First, the fitness function is a human rater. This presents a significant algorithmic bottleneck, but is a process that has been used successfully in several genetic algorithm art projects (MacCallum 2012). This is not an ad hoc change in most contexts, since the fitness function can be whatever function accurately captures the salient aspects of the domain. A human rater could be applied to genetic algorithms in many domains. Second, there are two types of populations. There is a population of measures and a population of phrases. Again, this is not an ad hoc change, this is just combining two genetic algorithms into one process. Third, the genetic modifiers are musically, rather than biologically inspired. This is an ad hoc modification that strictly applies to music. The standard genetic modifiers of mutation and crossover can generate enough variation to find musically valid output (Nierhaus 2010), however, the traditional operators would take more generations of evolution, and a larger population. The musically inspired genetic modifiers are a required modification due to the bottleneck of a human rater. By using musically inspired genetic modifiers, Biles encodes musical information in the algorithm just as Cope did in the subroutines in EMI.

Despite all of these modifications, it is clear that GenJam includes all four standard ingredients of a genetic algorithm (Floreano et al. 2008, 13). It has a genetic representation of the solution, a population, genetic modifiers, and a type of fitness function. GenJam is similar to Cope's EMI because both programs are ad hoc modifications of algorithms that have been successful in other domains of artificial intelligence. In both cases, the composers considered the musical output more important than adhering to algorithmic purity.

Since the 1990s, GenJam has been heavily modified by Biles. In a demonstration at MUME 2013, GenJam played live with an improviser. The improviser's performance was bred with individuals who were previously bred for that tune, then genetic modifiers were applied, and the results were recombined into a new solo (Biles 2013). This modification is an even greater diversion from a standard genetic algorithm because Biles removed the fitness function entirely. Rather than use a fitness function, he chose to use musically-informed genetic modifiers, which can only generate music that is valid to its context.

WolframTones

WolframTones is a project initiated by Stephen Wolfram and Peter Overmann in the early 2000s to apply cellular automata to music generation.

Cellular automata are models of discrete dynamic systems. Physicists such as John von Neumann, Stanislaw Ulam, and Konrad Zuse used cellular automata to model magnetism, crystal structure, and population growth and decay. Cellular automata require six basic ingredients, a cell space, a state set, a rule set, a neighborhood, initial conditions, and time (Floreano et al. 2008, 102). There are many types of cellular automata that can be defined using these properties.

WolframTones is a publicly accessible music generation tool that allows a user to specify parameters such as genre, instrumentation, pitch mapping, and rhythmic pattern. The program searches for a cellular automaton type and rule that satisfies the user parameters. After picking an automaton, the program flips the two dimensional output of the automaton on its side, then uses that data as pitch and duration information for each instrument selected by the user. The automaton data is mapped to note events using rules relating to genre, rhythmic patterns, and instrumental roles.

The ad hoc elements of WolframTones enter through the transformations applied to the automaton to account for the genre. Wolfram indicates that a subprogram searches for a valid rule set. The exact structure of this search is unclear. The output of the one-dimensional automaton is rotated ninety degrees, then strips of the output are used as a data source for each instrument. A parameter called height determines how many rows of the output are assigned to each instrument, thus how many possible pitches can occur in the resulting part. Specific values that occur in the output are then mapped to a scale or to a rhythmic pattern. WolframTones uses rules about instrument roles to distribute selected automaton data to selected voices.

> "Different Roles in effect represent different algorithms for picking out features in the automaton pattern. WolframTones includes many such algorithms; different ones are typically chosen for different musical styles. None plays no notes. Polyphonic plays all notes that satisfy certain basic WolframTones criteria; it typically plays many notes at a time. Lead n plays one note at a time, allowing several choices for how the note should be picked out from black cells in the underlying pattern. Chords n plays a few notes at a time. Bass n plays one note at time, placing it at a lower pitch level." (Wolfram Research n.d.)

The "certain basic WolframTones criteria" are unspecified, but it's clear that this represents a significant part of what the application does. Is the intelligent behavior then resulting from the cellular automaton or from the rules applied to it?

Rules also come into play to pick rhythms as well.

> "WolframTones includes a large number of algorithms for deriving drumming from underlying automaton patterns. Each algorithm in effect specifies a different procedure for determining what configuration of notes in each beat should produce what drumming. Different styles of music typically involve characteristically different drumming patterns. In WolframTones all Percussion choices are nevertheless in principle available for all styles--though 'crossing' drumming patterns can lead to unusual results." (Wolfram Research n.d.)

In the end, rules seem to dominate the data that is derived from the cellular automaton.

Swarm Music

Musical knowledge can be encoded in the algorithm itself, as in the previous examples, or it can be encoded in the data on which the algorithm operates. If a composer specifically organizes the data on which the algorithm operates in order to push the algorithm toward specific aesthetically acceptable outcomes, then intelligence is encoded in the data, not simulated by the program.

A good example of this is my application of swarm intelligence to generate sound collages. In my masters work I wrote a program called Becoming Live that mapped the boids algorithm to audio samples, and notated music. It did this by quantizing the swarm playing field into a discrete grid and measuring the number of agents in a grid space. If a space was crowded, then the associated music data was added to the output. Movement of the swarm simply triggered music data that I had previously arranged into a grid. This mapping is clearly ad hoc, as any system that navigates a two dimensional space could be substituted here for the swarm. A random walk, for example, could be used to navigate the two dimensional space. The aesthetically pleasing nature of the output is derived from the composer's input rather than the swarm actions. As long as the music data is strategically arranged by the composer, then the output will be acceptable.

Tim Blackwell's application of swarm intelligence to music generation is inherently less ad hoc. In Blackwell's mapping, the swarm moves on a three dimensional musical space of pitch, loudness, and location in measure. These parameters are less ad hoc than my mapping because they are basic properties of MIDI-based music. Further, the swarm is moved around the space based on attractors that are derived from MIDI input. In Blackwell's simulation, attractors are placed in the space based on input from a live musician. The musician's note events attract the swarm to similar note events. Blackwell's swarms are responding to another musician, whereas in my mapping, the swarm wanders randomly based on the parameters of the swarm simulation. Blackwell's swarm algorithm is otherwise a standard boids simulation (Floreano 2008, 532).

Blackwell's algorithm still contains ad hoc elements based on the use of MIDI for input and output. MIDI assumes the western chromatic scale, which assumes western instruments. This seems like a technical limitation more than a way of embedding intelligence in the algorithm, however, a swarm navigating a space defined by frequency and timbre could be applicable to a wider range of musical situations.

Conclusions

Ad hoc modifications exist in most algorithmic music programs. This should be expected to a degree, since there is a minimum level of modification that is necessary to generate musical data. Still, even the canonical examples of particular techniques, contain significant deviations that apply only to the domain of music generation. This is probably not an earth-shattering revelation to most readers, but it's worth remembering that ad hoc modifications are deeply embedded in artificial intelligence in music. If a program such as Cope's contains ad hoc elements, this does not make it an exception that is outside the mainstream, but rather brings it closer to common practice.

It is appropriate that composers care more about the end result than about the nebulous idea of algorithmic purity, but ad hoc elements must be considered when placing algorithmic composition in the larger context of computational creativity. These systems were designed to create art, not to test a hypothesis in a scientific way. As such, it is dangerous to generalize about such systems' capabilities, and their relationship to human creativity. I am extremely skeptical of the claims about computational

creativity put forward by authors such as Boden and Wolfram based on examplar systems such as those discussed here. Boden writes that "whatever it is that [EMI is] doing is clearly very general in nature" (Boden 2004, 312). Stephen Wolfram makes similarly broad claims about his WolframTones system.

> "even with the rules of a simple program, it was possible to produce the kind of richness and complexity that, for example, we see and admire in nature... We were making music that was good enough that people assumed it must have the usual human origins: we had succeeded in passing the analog of the Turing test for music." (Wolfram 2011)

I do not doubt that the output from these programs can be called creative, but in nearly every computational music program there are ad hoc modifications that cast doubt on the effectiveness of such systems in other domains. Ad hoc methods are used to model a specific task rather than the general functioning of the brain. These programs may reflect how people create specific forms of music, but it seems unlikely that they tell us much about creative dance, sculpture, web comics, or even music from other traditions.

One of the great aspects of MUME 2013 and MUME 2014 is the integration with the AIIDE conference, where we are intermingled with other procedural content practitioners. Musical metacreators could benefit from the type of rhetorical discipline advocated by Gillian Smith in *The Seven Deadly Sins of PCG Research*. Musical metacreators must strive to answer the questions she poses in a quantifiable way.

> "How meaningfully different are the pieces of content from each other? Are there certain kinds of content that simply cannot be created? Are there certain kinds of content that the generator seems to be biased towards creating?... What were the assumptions made when creating the generator - do you assume a particular art style, or range of acceptable... values?" (Smith 2013)

Smith's questions can be easily applied to music. Musical metacreators should be held accountable for supplying more than just hand picked examples. Counter examples where an algorithm fails should be part of every technical report. How is GenJam tied to the jazz idiom? Does Cope's EMI model structures in non-western music? How does Blackwell's swarm music perform in a more structured style of music? If these questions are left unanswered, then it is incumbent on musicologists to supply these answers. Claims made by software creators must be examined by researchers who understand both the code and the underlying algorithms.

Finally, we must consider the question of why successful algorithmic composition programs rely so much on scruffy, ad hoc techniques. Music is tied to culture and tradition as well as mathematical systems. Is algorithmic purity desirable, or even compatible with the goals of algorithmic composition as a field?

References

Biles, John A. 1994. GenJam: A Genetic Algorithm for Generating Jazz Solos. In Proceedings of the 1994 International Computer Music Conference. Accessed May 14, 2014. http://igm.rit.edu/~jabics/BilesICMC94.pdf.

Biles, John A. 2013. Straight-Ahead Jazz with GenJam: A Quick Demonstration. In Musical Metacreation: Papers from the 2013 AIIDE Workshop.

Blackwell, Tim. n.d. Swarm Music: Improvised Music with Multi-Swarms. Accessed May 8, 2013. http://igor.gold.ac.uk/~mas01tb/papers/SwarmMusicImprovisedMusicWithMultiswarms.pdf.

Boden, Margaret A. 2004. *The Creative Mind: Myths and Mechanisms*. 2nd ed. London: Routledge.

Cope, David. 2005. *Computer Models of Musical Creativity*. Cambridge, MA: The MIT Press

Cope, David. 2001. *Virtual Music: Computer Synthesis of Musical Style*. Cambridge, MA: The MIT Press.

Floreano, Dario and Claudio Mattiussi. 2008. *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. Cambridge, MA: The MIT Press.

Hofstadter, Doublas. 2001. Staring Emmy Straight in the Eye. In *Virtual Music: Computer Synthesis of Musical Style* ed. David Cope. Cambridge, MA: The MIT Press.

MacCallum, Robert M., Matthias Mauch, Austin Burt, and Armand M. Leroi. 2012. Evolution of Music by Public Choice. In Proceedings of the National Academy of Sciences of the United States of America.

Nierhaus, Gerhard. 2010. *Algorithmic Composition: Paradigms of Automated Music Generation*. Germany: SpringerWienNewYork.

Nilsson, Nils J. 2010. *The Quest for Artificial Intelligence: A History of Ideas and Achievements*. New York: Cambridge University Press.

Princeton University. 2010. About WordNet. Accessed July 1, 2014. http://wordnet.princeton.edu/wordnet/citing-wordnet/.

Smith, Gillian. 2013. The Seven Deadly Sins of PCG Research. Accessed July 1, 2014. http://sokath.com/main/the-seven-deadly-sins-of-pcg-papers-questionable-claims-edition/.

Wolfram Research. n.d. Composition Controls. Accessed August 10, 2014. http://tones.wolfram.com/about/controls.html.

Wolfram, Stephen. 2011. Music, Mathematica, and the Computational Universe. Accessed July 1, 2014. http://blog.stephenwolfram.com/2011/06/music-mathematica-and-the-computational-universe/.