Algorithmic Composition as a Model of Creativity

Bruce L Jacob

Advanced Computer Architecture Lab, EECS Department, University of Michigan,
Ann Arbor, MI 48109-2122 USA. e-mail: blj@eecs.umich.edu
http://www.eecs.umich.edu/~blj/algorithmic_composition/

Abstract. There are two distinct types of creativity: the flash out of the blue (inspiration? genius?), and the process of incremental revisions (hard work). Not only are we years away from modeling the former, we do not even begin to understand it. The latter is algorithmic in nature and has been modeled in many systems both musical and non-musical.

Algorithmic composition is as old as music composition. It is often considered a cheat, a way out when the composer needs material and/or inspiration. It can also be thought of as a compositional tool that simply makes the composer's work go faster. This article makes a case for algorithmic composition as such a tool. The 'hard work' type of creativity often involves trying many different combinations against each other and choosing one over others. This iterative task seems natural to be expressed as a computer algorithm. The implementation issues can be reduced to two components: how to understand one's own creative process well enough to reproduce it as an algorithm, and how to program a computer to differentiate between 'good' and 'bad' music. The philosophical issues reduce to the question *who or what is responsible for the music produced?*

1. Introduction

The question *what is algorithmic composition* is analogous to the question *what is artificial intelligence*. Both are hotly debated. Neither are answered easily. Both ask fundamental questions about the authorship of ideas.

Algorithmic composition is the application of a rigid, well-defined algorithm to the process of composing music. It is frowned upon by 'traditional' composers because it is often used as a means to expand one's musical palette. The explicit message is that algorithmic composition is invalid as a methodology, but the implicit message is that the music produced is in some sense unlike what the composer would have produced without help. In other words, music produced by algorithmic composition is considered somehow inferior not because it was produced by an algorithm, but because it is someone else's music—it belongs to the *designer* of the algorithm, and not to the *user* of the algorithm. One can avoid this criticism merely by implementing one's own algorithm.

To further support the algorithmic approach, the only difference between a composer's creative methodology and some algorithm that approximates it is that the composer can exhibit much more flexibility. An algorithm by definition is rigid, whereas creativity often breaks rules. But there is something to be said for following the rules through to completion. This is what the algorithm does quite easily but what a human composer will often avoid—we like to break rules in the name of creative license. Though it is difficult to follow the rules through to completion, there is a reason why well-defined structures are used to create music, art, sculpture, dance, poetry: in order to follow the rules, one will often devise creative solutions that might not otherwise have been chosen. Therefore, a computer is well suited to the task of creativity-through-hard-work, as it is incapable of stepping outside of the well-defined structures.

However, we are still left with the question of authorship. To return to the artificial intelligence theme, if an algorithm faithfully represents an artist's creative process, what is the difference between music produced by the artist and music produced by the algorithm?

1.1. The Goal of Algorithmic Composition

Creativity sometimes arrives in a sudden warm embrace, leaving one with a giddy sense of inspiration, vision, and purpose—resulting in a moment of clarity that is both inexplicable and undeniable. Other times, one starts with a vague creative seed to spend countless hours of revision and rethinking to hammer out a work of blood, sweat, tears, but mostly frustration.

In short, creativity comes in two flavors: genius and hard work. While the former may produce more 'inspired' music, we do not fully understand it and therefore have a slim chance of reproducing it. The latter resembles an iterative algorithm that attempts to achieve some optimal function of merit, and is therefore more easily realizable as a computer program.

We therefore have defined a clear goal for an algorithmic composition system: to reproduce the composer's creative methodology when the composer is in 'hard work' mode. The result is a system tailored for that particular composer (call him A); if another person were to use the system to produce music, they would be composing A's music, not their own.

1.2. Philosophical Digression

This brings us to an interesting digression on the difference between music composition and music recognition. When an algorithm written by a composer produces music that is not exactly what the composer would produce, the composer filters it—he culls out the parts that conflict with his own personal tastes.† Is this composition? How minimal must the changes be in order for the algorithm-produced music to be considered *composed* (by the algorithm or the composer)? How extensive must the changes be before one's role as algorithm-designer

† We will assume for the moment that we are not dealing with Cage here.

moves from composer to editor? At what point is the composer merely *recognizing* music that he likes? Is there a difference?

To take this a step further, suppose the 'algorithm' is a machine devised by the composer that beats on piano keys more or less randomly. The composer records key-banging sessions to edit them later and stitch together the portions that make up a good piece (in his estimation). What is the role of the composer here? Is the composer *composing* music or merely *recognizing* music? Again, is there a difference?

One step further: here, the composer travels to the nearest algorithmic composition store and purchases a machine that beats on piano keys more or less randomly, as well as a nice tape deck for recording and editing the session. He takes the machine home, records a session, and edits it down to a piece he calls *Rhapsody in absentia*. Whose music is this? To whom can we attribute the results? To the piano-punching machine? To the designer of the machine? To the user of the machine who recognized that certain passages were viable compositions? Or is the *Rhapsody* merely non-music, sounds that do not count as music because no human actually envisioned them before they were heard—although this definition would seem to be a categorical cop-out.

If we simply look at the two diametrically opposed viewpoints we unearth much of the issue. One viewpoint is that art is communication and so anything one communicates to another is art; art is the physical equivalent of "hey—look at this." A piece of driftwood found on the beach, once recognized as interesting to the beholder, is art, even though the person who takes the driftwood home and mounts it on the wall is not responsible for any of its characteristics, including its shape, its color, its texture, etc. This person did not stand at the seaside and dangle the wood in saltwater for several decades in order to get the desired results, he merely discovered the wood and said, "hey—look at this."

The other viewpoint is that only what comes directly from the hands of the artist is of the artist. This is the viewpoint that has caused many of the world's Rembrandts to devalue (to name just one of many artists) because some of the work was done by assistants, not by the master. Where the other viewpoint considers vision more important than toil, this viewpoint considers toil more important than vision.

One viewpoint discounts all algorithmic composition, another embraces it. There are no answers here, this is merely an aside to suggest that the more closely an algorithm reflects a composer's methodology, the less question there is that the work is authentic and of the composer.

1.3. Evolution of the Role of the Computer

Computers are not new to music composition; Max Mathews was developing sound synthesis programs at AT&T in the 1950's [Roads 1989]. For decades, the computer has been used to compose music, typically via probabilistic or stochastic methods [Hiller 1981,

Laske 1981, Englert 1989]. Articles by Lorrain and Jones [Lorrain 1980, Jones 1981] provide excellent descriptions of the mathematics involved. Until the mid-eighties, much of the attention was focused on timbres, rather than the structure of the music composed (e.g. [Truax 1982, Risset 1985, McNabb 1981, Vaggione 1982]). This is understandable, given the state of the art—the computational limits of the computers and the relative lack of rich sound sources. Also, given the limits of early computers, it seems reasonable that much of the composition algorithms were centered around notes, e.g. Markov chains giving next-note probabilities, rather than around higher-level structures such as phrases or thematic material. Note-based composition is much less computationally intensive, but with its simplicity comes an organizational tradeoff. A note-based method works at the level of trees and is hard-pressed to create a forest, except as a collection of trees.

Today, with samplers, sample-based synthesizers, and orders of magnitude more compute cycles readily available, more attention can be placed on structured composition. There are many examples of complex composition algorithms which produce 'correct' music, most of them centered around databases of style descriptions or rules [Cope 1987, Cope 1992, Friberg 1991, Widmer 1992]. On the other side, the computer is still often used as a fast randomizing agent; popular generation programs M and Jam Factory both use Markov chains to create lines which are statistically close to what the user inputs [Rowe 1993, Walker 1994, Zicarelli 1987]. Another tendency is to use the computer as an accompanist who listens to what is being played and responds appropriately in real-time. Here, the human input is used to generate rules on which the machine will base its output. This is seen in such programs as Cypher [Rowe 1993] and IBL-Smart [Widmer 1994].

As computers get more powerful, they become capable of handling increasingly complex tasks. Accordingly, as time goes on, composers have used computers to more closely model the creative process. This has happened not just in music composition, but in an entire range of creative pursuits from analogy-making to the creation of typefaces [Hofstadter 1995]. This is an inevitable trend, but it is also vital to the validation of algorithmic composition and computer-generated art in general. The more closely we can model our creative processes, the more the computer becomes a simple tool for artistic creation, and less a replacement for inspiration.

2. The *variations* System

While this article is a generic argument for algorithmic composition, it uses examples that come from a specific system. In order to better understand the examples, this section describes the system briefly. For a more complete description, see [Jacob 1995] or the author's webpage.

2.1. System Overview

variations is an algorithmic composition system that attempts to model the 'hard work' ethic of creativity. The system was designed to reproduce very closely the creative process that this author uses when composing music, which is very similar to writing a canon, except that the variations on themes need not be isomorphisms. The human creative process allows one to 'make up' new themes whenever they are needed or wanted. Creativity is difficult to define, let alone program into a computer, so the system makes no attempt at thematic generation—it only produces variations on existing material. However, since the themes can be variations upon variations, the net result is often indistinguishable from creating new, barely related thematic material.

The system works at the level of motives, which simplifies the organization of the music. It is much easier to create structure in a piece when one works at a higher level than it is when working at the level of notes—it is hard to force a next-note-probability model into creating structured pieces riddled with thematic development. In the *variations* system, order is imposed on the music produced by definition; by ensuring that all of the notes used in the resultant piece belong to phrases related to each other through transformation, a certain amount of thematic development is inherent, almost inevitable, in the music. This allows the system to pay more attention (and computer cycles) to harmonic progression.

The compositional algorithm:

1. Define a number of primary themes (motives) to be used in the composition.
2. Compose phrases by creating motives and adding them one by one to the phrase. At each step, judge the quality of the resultant phrase and remove the last motive if the combination is unsuitable.
3. Create motives by selecting at random from the primary themes and motives already in the phrase, and producing variations upon the selection.
4. Once there are a large number of phrases, join them together into larger frameworks.

The primary software components are the COMPOSER and EAR modules. The COMPOSER produces material and the EAR filters out whatever is 'bad.' It is a producer-consumer paradigm, popular in music composition systems. One side produces music, the other side consumes it and critiques it, affecting the future output of the producer.

Figure 1 illustrates the process. The COMPOSER takes as input a number of melodies, and composes a phrase from them motive by motive. The COMPOSER creates music by producing a variation on a previous motive and layering and sequencing it with the other motives. Each motive is either a copy of a primary motive, a variation upon a primary motive, a copy of a previous motive, or a variation upon a previous motive. At each step, the resultant phrase is tested by the EAR and given a yes/no grade. If the EAR likes the piece, the composition process continues. If the EAR does not like the piece, the COMPOSER deletes the motive and tries a different variation.

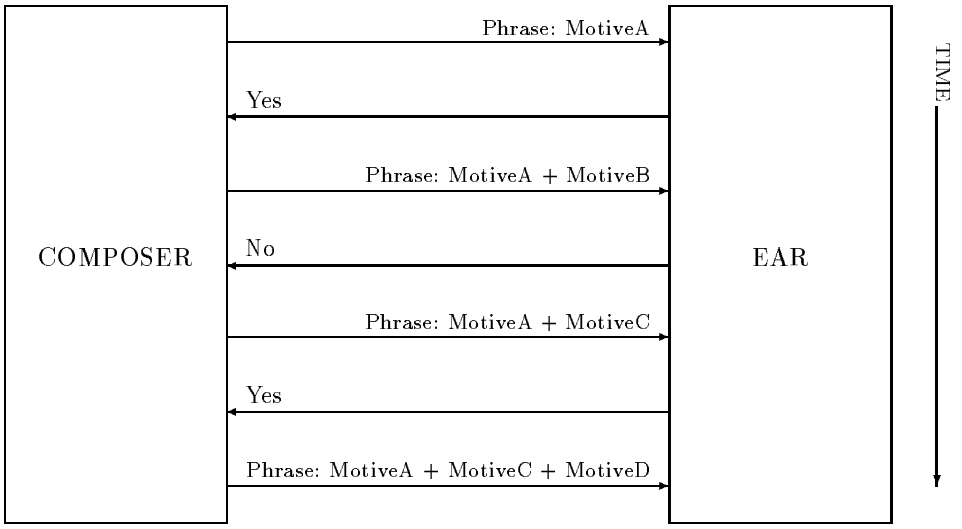


Figure 1. The producer-consumer relationship between the COMPOSER and EAR modules.

2.2. Motive-Oriented Organization of the Composer

The system organizes and manipulates material at the level of motives, so that thematic development is inherent in the music produced. A motive-oriented internal representation forces an implied structure on the compositional process; it allows for easy coherent manipulation of the material so that the resultant thematic variations are neither trivial nor unrecognizable.

Each motive contains the following:

- a list of **pitches**, where the possible values are 1 (tonic), 2, 3, 4 (subdominant), 5 (dominant), etc.
- corresponding lists of **rhythms** and **articulations**
- **key** information: the tonic (including which octave it is in), whether the key is major or minor
- **timing** information: offset in measures from previous motive
- **part** information: which instrument

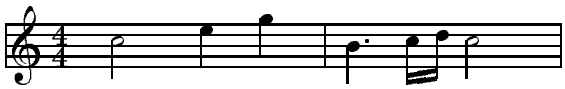
The chosen form encourages transformation: transposition is a simple matter of changing the tonic or adding the same number to every member of the pitch list. Changing the rhythm is just as simple: dividing the elements of the rhythm list by 2 turns every quarter note into an eighth note, every eighth into a sixteenth and so on. Similarly changing the numbers in the articulation list will make a legato phrase sound more staccato.

Since the system begins with a small number of defined motives, resultant compositions will consist entirely of motives closely related to each other. Each phrase contains self-

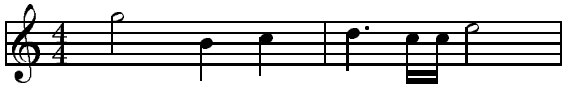
similar components, and all phrases based upon the same primary motives are related to one another. The component motives are all related by transformation, and the relations span from the obvious, such as echo or transposition, to the very subtle.

3. The Composition Process as an Algorithm

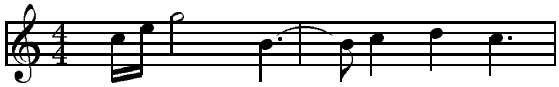
The goal of this section is to demonstrate through examples that composing-by-hard-work is inherently algorithmic. The composition process described in the previous section is not specific to a computer program; the process is quite similar to the creative processes of a human. Phrases are built up from motives one by one. Motives are composed by arbitrarily picking from a few themes, modifying them slightly, and adding them to the work-in-progress. For example, we start with the following well-known theme (typeset by Daniel Taupin in his MusicTEX manual):



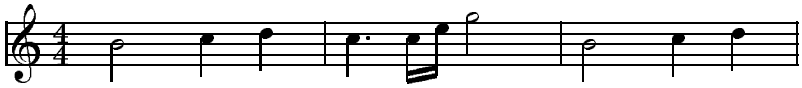
If the length of the variation is to be seven notes, then the following would be a possible result: †



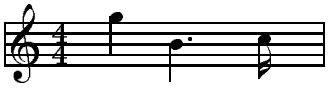
The following would be just as viable:



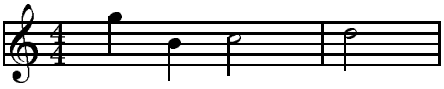
If the length is to be longer or shorter than seven, we can get simple variations such as this:



or this:

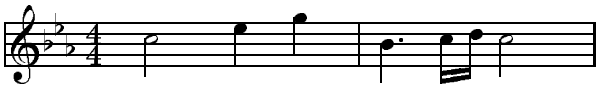


We can also see more dramatic changes. We can choose to run pitches forward and rhythms backward:

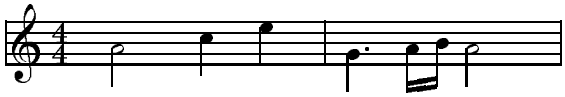


† Please note that the inclusion of a time signature is simply to make the rhythms of the phrases easier to read. Some phrases will not fill out the last measure exactly, but do not have implicit rests at the end.

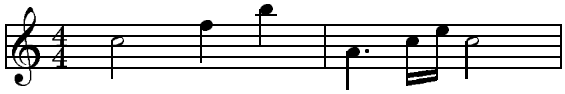
We can change the key:



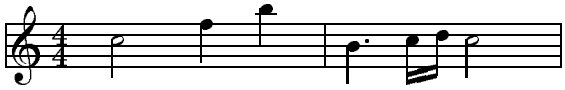
We can transpose (note the lack of accidentals):



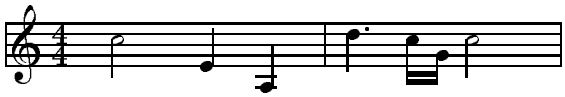
We can multiply all pitch values by 1.5 and round:



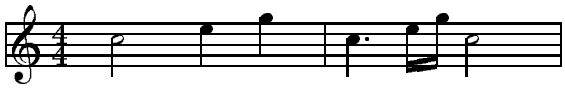
We can multiply all pitch values by 1.5 and truncate:



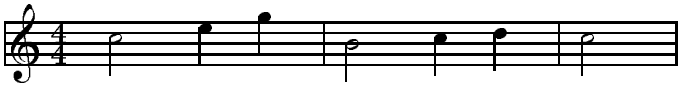
We can multiply the pitch values by -2.0:



We can choose to use just a subset of the pitches (maintaining order):

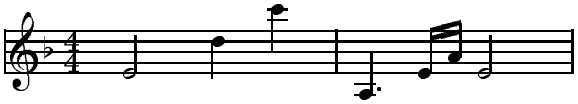


We can choose to use just a subset of the rhythms (also maintaining order):

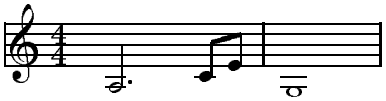


When we start combining the changes with each other, we get more complex behavior. The variations begin to look less like the original theme and more like the weighted-probability characteristics of Markov processes for the choice of notes, rhythms, and articulations. However, it is still clear that there is a relationship between the motives produced.

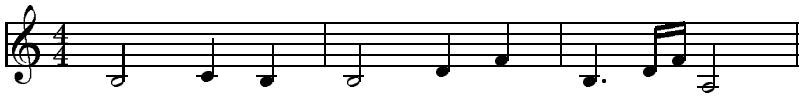
For instance, we can multiply the pitch values by 3 and change keys:



We can choose a shorter list, multiply the rhythm values by 2, choose different subsets of pitches and rhythms, and transpose:

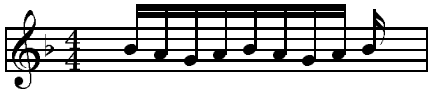


We can choose a longer list, translate within C major, and mismatch the rhythms:

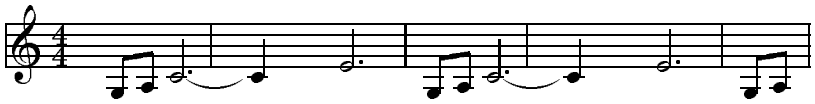


Things get very interesting when we perform variations on the variations. Each variation on a theme is akin to a poor-quality photocopy or musical recording—with each iteration, the resemblance to the original fades. After several iterations the original is almost indistinguishable. If we only consider the last three examples, which were fairly normal variations to begin with, we get motives that barely resemble the original theme.

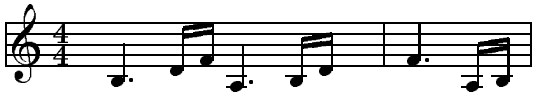
For instance, we can modify the first theme by dividing the pitch values by 4, selecting a subset of the pitches, selecting a different subset of rhythms, and transposing:



We can modify the second theme by repeating the entire theme while mismatching the pitches and rhythms:



We can modify the third theme by repeating subsets of the pitches and rhythms:



These modifications are all classic variations used in canons and fugues to vary thematic material. There is nothing special or inspired about the examples—they are merely provided to demonstrate that there is an extremely large number of possible modifications that can be done to a piece of music, many more possibilities than a human would want to explore, or even *could* explore in a reasonable time. The beauty of using a computer is that it *can* explore the space.

4. Algorithmic Compositions: Variations on a Theme by Mozart

There is a search space that is composed of all possible variations on a theme; the size of this space cannot be overestimated. To give a feel for its extent, a dozen examples of phrases that have been derived from the original theme are presented. They have not passed any sort of quality test, they are merely presented as they were generated by the algorithmic composition system *variations*.†

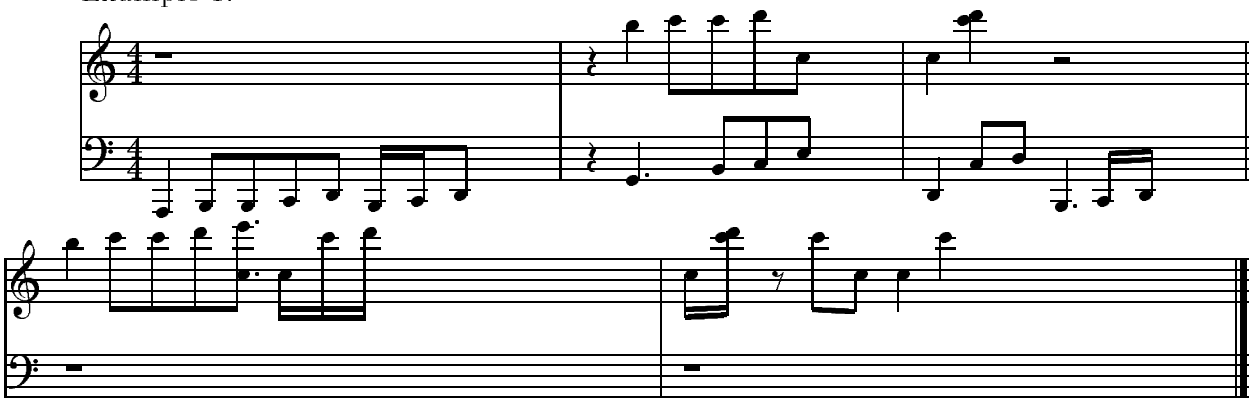
The original theme is recognizable in some of the extracts, in others it seems com-

† They happen to be sections of pieces that neither broke MIDI2TEX nor required extensive editing of the generated MusicTEX macros.

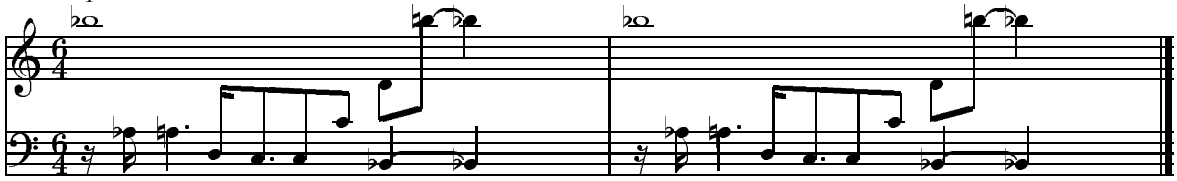
pletely absent. Nonetheless, all of the themes were produced using the methods described in the previous section, based solely on the two bars of Mozart's melody. And even when the relation between the extracts and the original theme is unclear, the relation between the extracts produced is still evident.

These examples also provide an illustration of the blurred distinction between simple variation and originality. If one did not know to look for the seed theme in these extracts, one would likely never find it. The reason is that the system is allowed to create variations on the variations, and variations upon them—so that very quickly the piece moves far from the original music source. What is this, then, if not creativity?

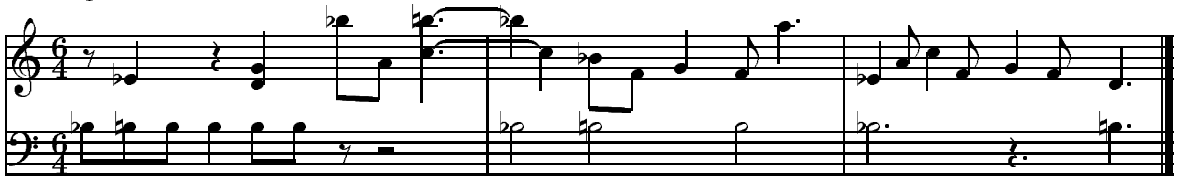
Example 1.



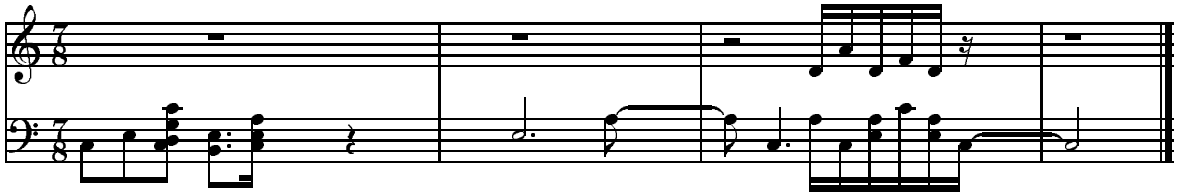
Example 2.



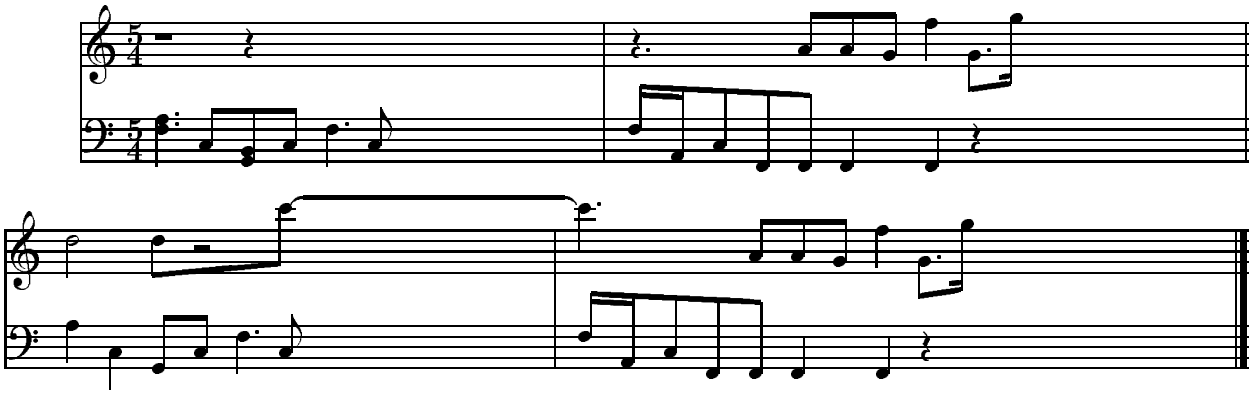
Example 3.



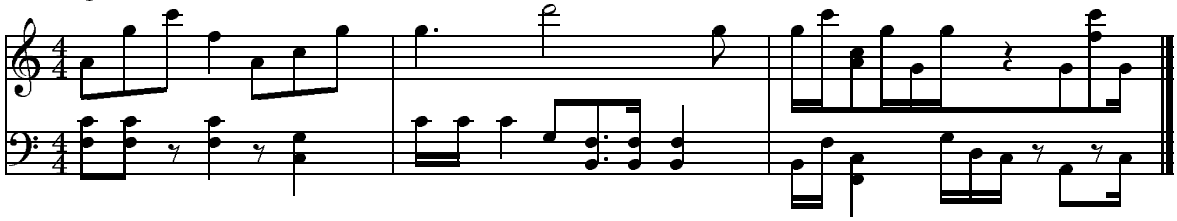
Example 4.



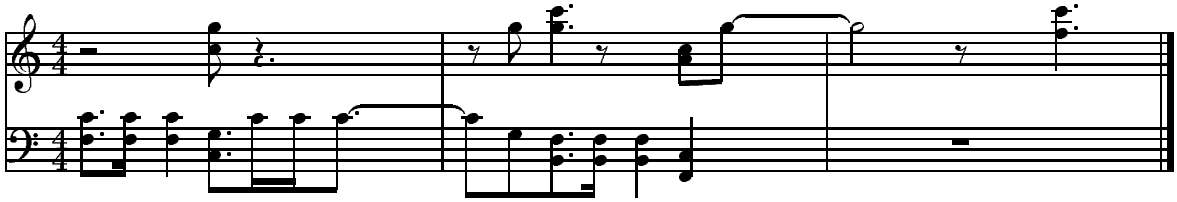
Example 5.



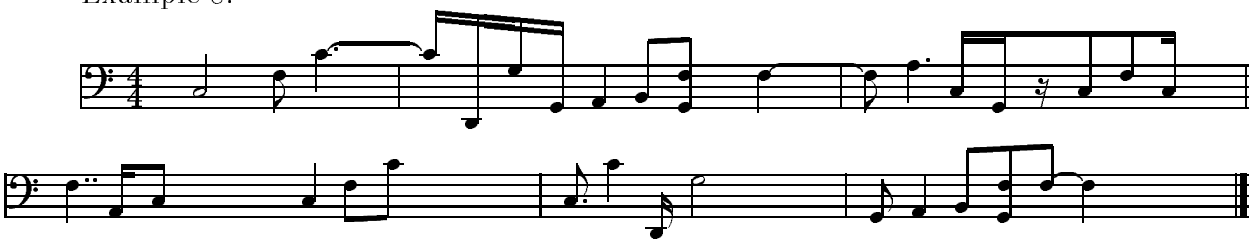
Example 6.



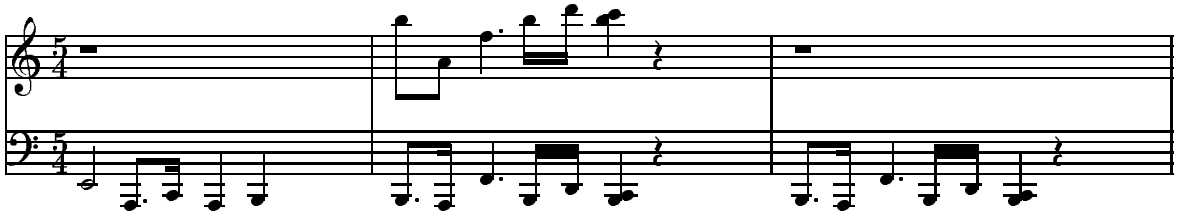
Example 7.

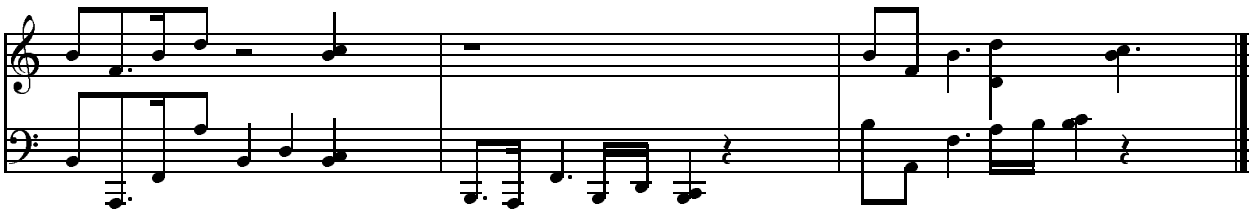


Example 8.

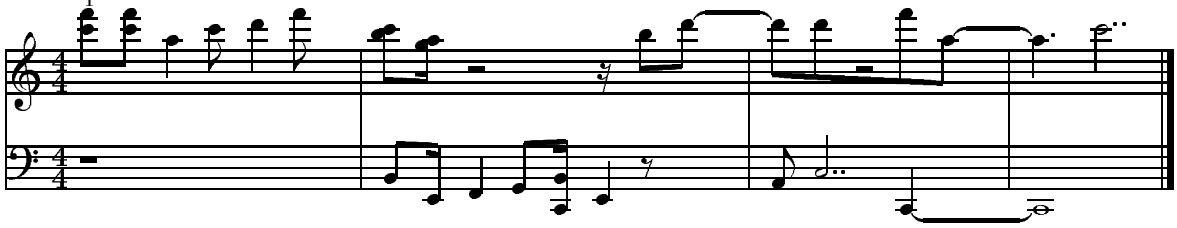


Example 9.

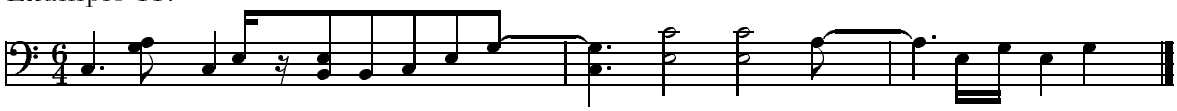




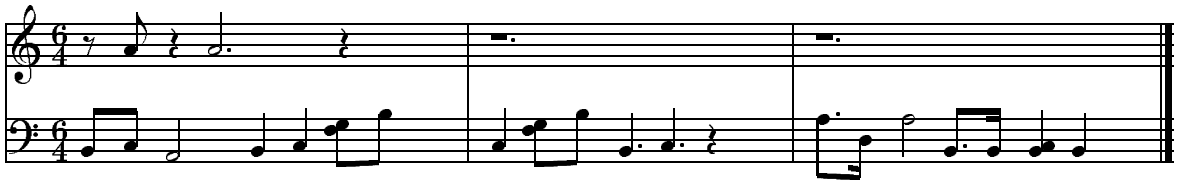
Example 10.



Example 11.



Example 12.



These examples represent an infinitesimal portion of the space created by juxtaposing variations of just one theme against each other. The space gets exponentially larger as more themes are introduced. The computer is far better suited to performing routine functions repeatedly than a human is, so it is appropriate to assign the task of generating all possible variations to the computer. We have redefined our problem: now, the 'intelligent' part of music composition is no longer to define music composition; it is to be able to distinguish between 'good' and 'bad' material. While it is no trivial task, it is nonetheless a step in the right direction.

5. Conclusions

Algorithmic composition is a methodology for allowing a human composer to work more quickly. Its success depends very heavily upon two things:

1. A close match between a composer's creative methodology and the implemented algorithm.
2. An accurate mechanism for quickly determining the viability of a specific phrase.

The problem of computer-aided music composition is thus reduced from answering the nebulous *how do I create music* to answering two slightly more concrete questions: *what*

steps do I use to compose music and *what types of harmonic movement do I like.*

This is neither a trivialization of the compositional process nor a vague specification for an algorithmic composition system. These are the questions one must successfully answer before one can use algorithmic composition in a manner that escapes mudslinging. Each composer has a unique compositional process and therefore one algorithmic tool cannot hope to satisfy the demands of many different composers. Algorithmic composition is perhaps a little bit like religion or politics; one must find one's own path. The more closely one can match an algorithm to one's own creative process, the more the computer becomes a simple compositional tool, and the less it appears to be a compositional crutch.

References

Cope, D. 1987. An expert system for computer-assisted composition. *Computer Music Journal*, 11(4):30–46.

Cope, D. 1992. Computer modeling of musical intelligence in EMI. *Computer Music Journal*, 16(2):69–83.

Englert, G. 1989. Automated composition and composed automation. In Roads, C., editor, *The Music Machine*. The MIT Press, Cambridge MA.

Friberg, A. 1991. Generative rules for music performance: A formal description of a rule system. *Computer Music Journal*, 15(2):56–71.

Hiller, L. 1981. Composing with computers: A progress report. *Computer Music Journal*, 5(4):7–21.

Hofstadter, D. 1995. *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. BasicBooks (A Division of HarperCollins), New York NY.

Jacob, B. 1995. Composing with genetic algorithms. In *Proceedings of the 1995 International Computer Music Conference*, Banff, Alberta.

Jones, K. 1981. Compositional applications of stochastic processes. *Computer Music Journal*, 5(2):45–61.

Laske, O. 1981. Composition theory in Koenig's Project One and Project Two. *Computer Music Journal*, 5(4):54–65.

Lorrain, D. 1980. A panoply of stochastic 'cannons'. *Computer Music Journal*, 4(1):53–81.

McNabb, M. 1981. Dreamsong: The composition. *Computer Music Journal*, 5(4).

Risset, J.-C. 1985. Computer music experiments 1964-... *Computer Music Journal*, 9(1).

Roads, C. 1989. An interview with Max Mathews. In Roads, C., editor, *The Music Machine*. The MIT Press, Cambridge MA.

Rowe, R. 1993. *Interactive Music Systems*. The MIT Press, Cambridge MA.

Truax, B. 1982. Timbral construction in Arras as a stochastic process. *Computer Music Journal*, 6(3).

Vaggione, A. 1982. The making of Octuor. *Computer Music Journal*, 8(2).

Walker, W. 1994. *A Conversation-Based Framework for Musical Improvisation*. PhD thesis, University of Illinois at Urbana-Champaign.

Widmer, G. 1992. Qualitative perception modeling and intelligent musical learning. *Computer Music Journal*, 16(2):51–68.

Widmer, G. 1994. The synergy of music theory and AI: Learning multi-level expressive interpretation. Technical Report OEFAI-94-06, Austrian Research Institute for Artificial Intelligence.

Zicarelli, D. 1987. M and Jam Factory. *Computer Music Journal*, 11(4):13–29.