Metabolizing Music

Eric Iverson and Roger Hartley
Computing Research Lab Box
30001/3CRL New Mexico State
University Las Cruces, NM
88003-0001

Abstract

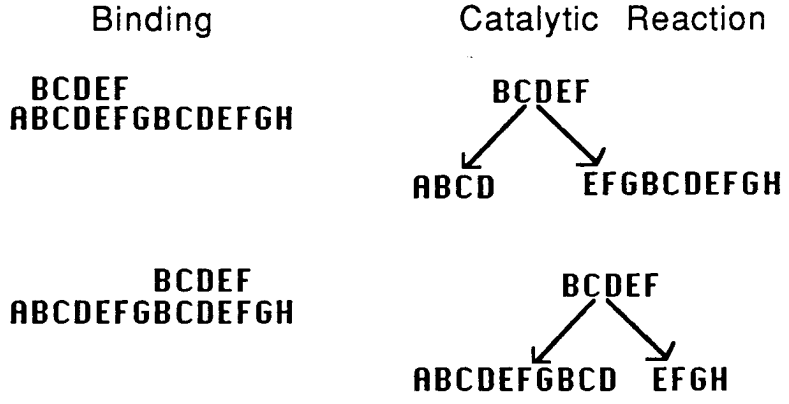
The paper introduces Metamuse, a program that analyzes a piece of music by breaking it into its component parts and then reassembling it into a similar piece. This is done through autocatalytic set theory, which effectively allows portions of a string to break apart other portions, much like a conventional chemical reaction. The end result of this process is a set of substrings that reflect the overall structure in that they are biased towards repetitive patterns present within the piece as a whole. The autocatalytic process can then be run in reverse until a target goal is reached.

Metamuse allows for a certain degree of evolution, since strings are allowed to copy themselves after catalyzing a reaction, with mutations sometimes occurring as a result. In this way a fitness metric is introduced such that strings which catalyze a number of reactions will be deemed more fit than others. Strings which are unable to catalyze reactions are excluded from the autocatalytic set and not allowed to replicate. In this way, self-organizing behavior can be simulated, thereby allowing for variations of a musical piece to be constructed in a non-deterministic manner.

Introduction

In the literature, there are a number of deterministic rule-based systems for the analysis of music, (see, for example [51). However, it is clear that this is not "how it's really done." That is to say, these systems are at best an approximation, in order that the rules can be used and understood by humans, and are at worst inadequate in the face of non-western tonality and various world music traditions. To counter these potential shortcomings, we have chosen to take a self-organizing adaptive systems approach, where a priori knowledge is kept to a minimum. Instead, the user feeds the system pieces of music which it assimilates and then uses to generate a piece representative of the patterns derived from the originals.

To do this, we chose to use Autocatalytic Set Theory. Briefly stated, an Autocatalytic Set is a set of strings where each member is the product of at least one reaction catalyzed by at least one other member. A reaction occurs when a catalyst binds onto a corresponding site on a string, and proceeds to break the string in two. Similarly, the reaction can occur in reverse; where the catalyst joins two strings together to create a larger string. This allows an autocatalytic set of strings to effectively bootstrap itself, making strings of greater and greater length until some goal state or level of stability is reached. Thus, autocatalytic sets can in some way simulate self-organizing behavior.



In the above reaction, BCDEF functions as the catalyst for string ABCDEFGBCDEFGH. It is important to note that there are two sites on the string that the catalyst can bind to. This allows for a certain amount of non-determinism, although only one of the sites is ultimately chosen. It should also be noted that the catalytic reaction can be run in reverse, thereby allowing BCDEF to join two different pairs of strings together with the same ultimate result. The only limit placed upon the catalytic reaction is that the catalyst must bind onto at least two elements of each string involved. This is done in order to limit the number of spurious combinations generated, while at the same time allowing a certain amount of novelty to still occur.

Autocatalysis, as it is being applied here, is essentially similar to using Markov Chains to analyze and then generate a string of elements. However, there are differences. For example, an autocatalytic analysis can be biased toward repetitive structures within a string. This can be done by rewarding a string that successfully catalyzes a reaction by allowing it to replicate. In this way, certain catalytic reactions will become more and more likely to occur as repetitive sites facilitate the replication of their catalysts. At the same time, during replication a certain degree of mutation can be allowed to take place, which allows for potentially productive additions to the autocatalytic set. In addition, unlike Markov Chains, autocatalysis can occur over a variety of different string lengths, thereby allowing larger strings to act as templates for the creation of even larger strings containing the original catalyst as a subset. This allows for the transferal of information at a variety of levels of complexity, thereby allowing for the possibility of repetitive motifs within the resultant string.

Algorithmic Composition

We take it that the goals of algorithmic composition are to produce "musical" compositions with the minimum amount of human interaction. The spectrum of such techniques runs from full-blown composition tools with automatic generation of fixed musical patterns (repetitive rhythms especially) to the rule+random systems where a musical skeleton is produced by a set of rules, and then variations are generated by randomizing the parameters. Examples are described in [6]. However, none of these techniques start from a given composition as the provider of musical ideas. A set of rules for say, Bach chorales, or Top-40 tunes may be induced by lengthy analysis of many examples, but this is not the same as taking the ideas in one composition and working from that basis. Metamuse does just that. Our techniques are independent of any particular compositional theory or personal biases. Autocatalysis simply finds whatever patterns exist in the composition, and builds new compositions from these patterns taking their mutual compatibility into account.

Applying Autocatalysis to Music

When applying autocatalysis to music, the process can be roughly divided into assimilation and regeneration. Assimilation is used to break a string into lengths of 4 to 7 elements, thereby creating a base set from which to begin. During the process of assimilation, it is likely that there will be points at which no further progress can be made without the introduction of a new catalyst. New catalysts are generated by finding a random site on an existing string and allowing a binding to occur; creating a new catalyst out of unassociated single elements in the process. These elements are not specifically listed as strings in the program's inventory, but are assumed to always be present in sufficient enough quantities to generate new strings as they are needed. This not only allows assimilation to continue, it also further defines the autocatalytic set being generated. After assimilation has been completed, regeneration takes place by taking the existing inventory of strings and using them to create longer and longer strings until some goal is met.

In order to assimilate a piece of music, one must first isolate it into different series of elements. These elements can correspond to aspects such as rhythm, pitch, interval, or any other salient feature the user wishes to isolate. This is similar to work done by Conklin and Cleary (1988) regarding the generation of music using multiple viewpoints. The important thing to remember is that the individual strings should consist of elements drawn from a relatively small set, in order that repetitive patterns will be more likely to occur. Also, it is important that the different series do not overly depend on each other (i.e. pitch and interval) so that conflict can be minimized during recombination. In the future we intend to implement a method so that conflicting viewpoints can enable or inhibit each other in order to create a final aggregate representation of the piece. In the examples below, we have minimized conflict, and maximized repetition by using strings drawn from the set {A,B}.

fig. 2

```
(06)BBABA -> (07)BABBA*
(01)AABBA||BABAA(08)
```

Here BBABA is acting as a catalyst by breaking AABBABABAA into AABBA and BABAA. It does this by binding onto a site **on AABBABABAA** where a one-to-one correspondence with its elements exists. At the same time, BBABA is rewarded for catalyzing a reaction by being allowed to copy itself. This copy is mutated (using a 5% probability of error) into BABBA.

When a piece has been completely assimilated, we are left with an inventory of strings from which to create a new piece. The probability aspect of Markov Chains is here simulated as populations of each string type relative to the others (i.e. the number of members of a type of string is equivalent to its probability of occurrence.) To create larger strings, we merely reverse the previous process:

fig. 3

```
(05)BAAAB -> (09)BAAAB*
(10)BABBAAABBA*
(07)BABBA||AABBA(01)
```

Here BAAAB is creating a new string BABBAAABBA out of the existing strings BABBA and AABBA. It also copies itself as BAAAB. It is important to remember that when a string is copied or created, both it and the template string(s) still exist. This allows us to build a growing inventory of strings which can act both as templates to create new strings, as well as catalysts for reactions.

Meta-Selection Techniques

While autocatalysis is a powerful tool, it cannot in and of itself create strings of the length needed in an average musical piece. One reason for this is that for every string of length N, at least two strings of average length N/2 are needed to create it. In addition, since shorter strings take fewer reactions to create than longer strings, we end up with a distribution heavily weighted towards shorter strings, when in fact it is the longer strings that we are interested in. Therefore, we can infer that strings over a certain length will become impractical if not impossible to generate. This can be seen in the distribution of strings created by Metamuse in fig. 4 as well as in fig. 5 which shows a similar distribution for the autocatalytic generation of polypeptides done by Farmer et al. (1986).

fig. 4

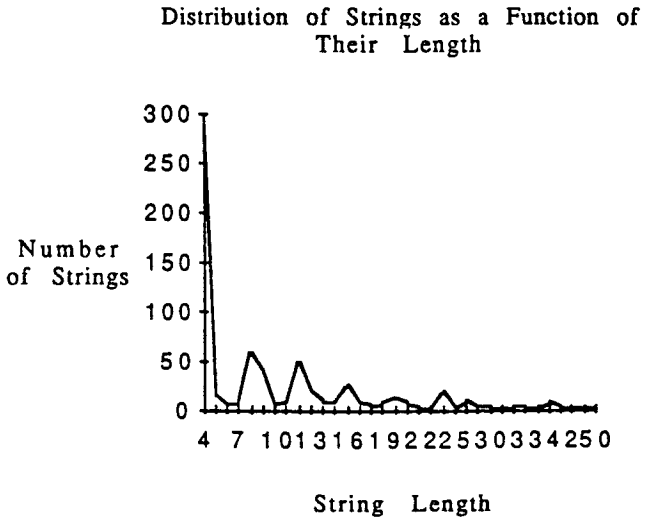
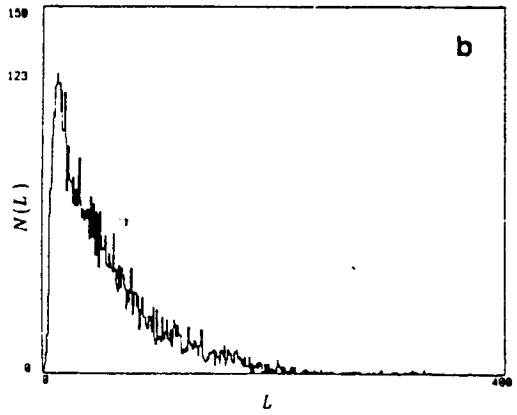


fig. 5

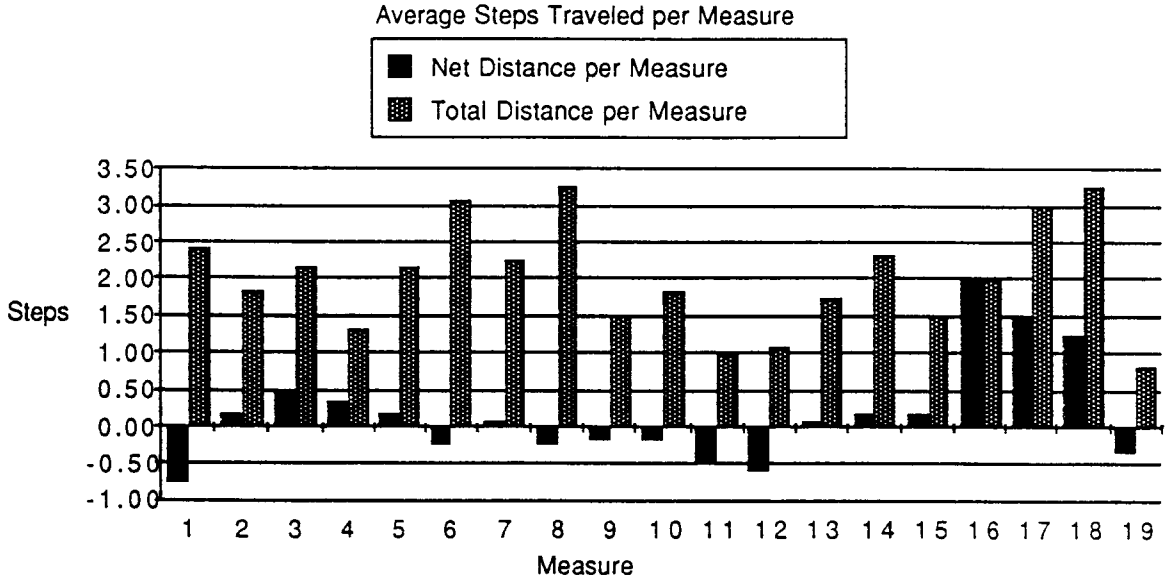


Distribution of Metamuse strings (fig. 4) and polypeptides (fig. 5) as a function of their length.

However, there are ways to extend the length of strings that can be practically generated. One way to do this is through a variant of simulated annealing. Here we would only allow strings over a certain length to be generated, while setting a quota for the number of strings to be generated exceeding that length. By gradually restricting the quota, while simultaneously raising the cutoff length, we will eventually generate a single string whose length matches our goal. This is analogous to a selection process in that we start out with a large number of candidate strings, and proceed to eliminate them until only one remains. While this constrains the number of smaller strings that are generated, its main benefit is that it allows us to proceed toward a goal length at a faster rate than would normally occur. This allows us to practically generate strings of longer length within a reasonable amount of time.

In addition, other meta-selection constraints can be applied to candidate strings before allowing them to be added to an existing set. Among these constraints are: presence of notes not in the original piece, notes failing outside of the range of the piece, and incompatibility with the statistical distribution of some aspect of the piece. To explain what we mean by statistical distribution, it is best to show an example:

fig. 6



Here we can see that some parts of the piece involve greater intervalic distance than others. This can either be expressed as net distance traveled by adding all positive and negative intervals, or as total distance by adding the absolute values of the intervals. Thus we can take a candidate string and compare it to an existing profile to see if it is "too busy" or lacks adequate contrast throughout. This approach could be applied to other aspects of the piece such as note duration. However, since the piece we analyzed consisted mainly of eighth notes, that has not been done here.

In order to use statistical distribution to increase the length of a piece we could average the values over a number of measures, thereby creating higher order sites that substrings could bind onto. This would be done by stipulating that the substring have a similar average parameter value as its parent site. In this way a rudimentary hierarchical structure could be generated; giving the piece greater length, as well as greater internal structure.

So far we have been able to generate pieces using simulated annealing, but have yet to implement meta-selection constraints. The pieces in the appendix show examples generated from the notes of the analyzed piece, as well as examples generated from its intervals. We feel that autocatalytic sets are a potentially powerful method for generating pieces of music in a non-deterministic, yet internally consistent manner.

References

[I] Conklin, D., and Cleary, J. G. "Modelling and Generating Music using Multiple Viewpoints," Proceedings of the First Workshop on Artificial Intelligence and Music, AAAI-88, 125-137, 1988.

[21 Farmer, J. D., Kauffman, S. A., and Packard, N. H. "Autocatalytic Replication of Polymers," Physica D, 22: 50-67, 1986.

[31 Kauffman, S. A. "Autocatalytic Sets of Proteins," Journal of Theoretical Biology, 119: 1-24, 1986.

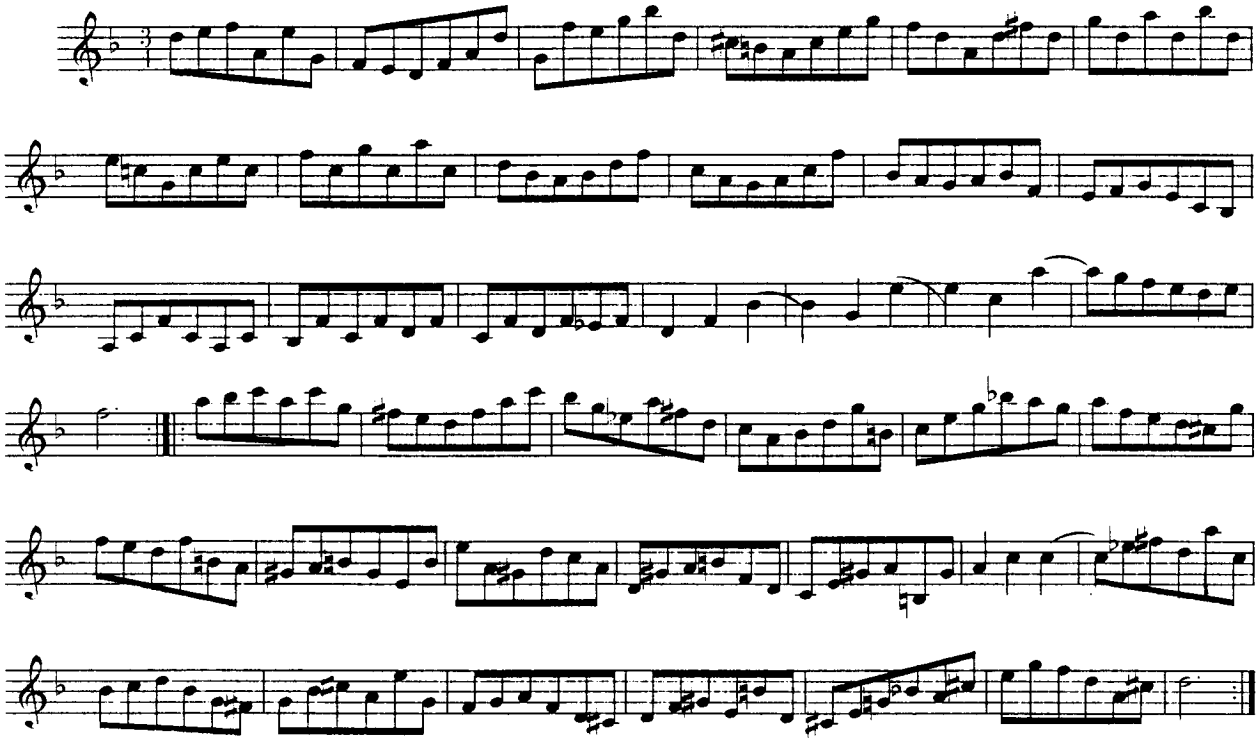
[4] Schuster, P. "Dynamics of Molecular Evolution," Physica D, 22: 100-119, 1986.

151 Ebcioglu, Kemal. "An Expert System for Harmonizing Four-part Chorales," Computer Music Journal 12(3): 43-51, 1988.

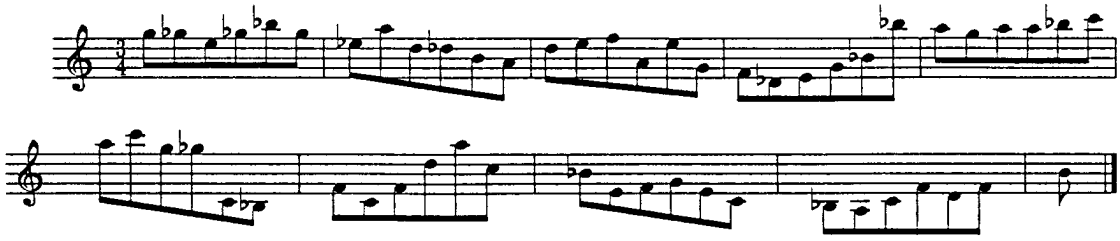
[6] Zicarelli, David. "M and Jam Factory," Computer Music Journal, 11(4): 13-29, 1987.

Prelude in C minor (transcribed for B flat Clarinet)

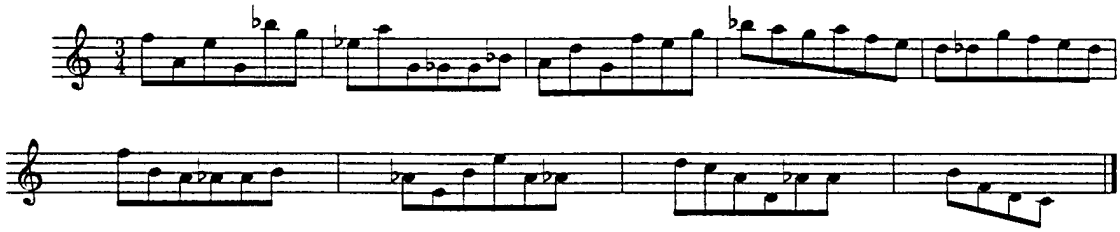
Johann Sebastian Bach



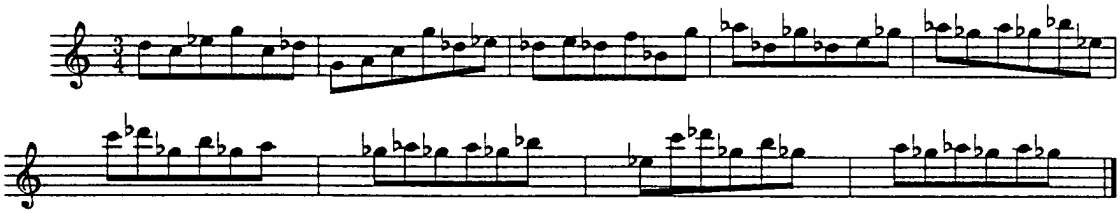
Note-Based Annealed Example 1



Note-Based Annealed Example 2



Interval-Based Annealed Example I



Interval-Based Annealed Example 2