Autonomous Evolution of Complete Piano Pieces and Performances

Palle Dahlstedt

Göteborg University / Chalmers University of Technology
Department of Applied Information Technology, Box 8718, 402 75 Göteborg, Sweden
palle@ituniv.se

Abstract. Evolutionary algorithms are used to evolve musical score material and corresponding performance data, in an autonomous process. In this way complete piano compositions are created and subsequently performed on a computer-controlled grand piano. The efficiency of the creative evolution depends to a large extent on the representation used, which in this case is based on recursively described binary trees. They can represent a wide variety of musical material and corresponding performance data in a compact form, with an inherent potential for musically meaningful variations and archetypal musical gestures. This is combined with a set of automated formalized selection criteria based on experiences from human selection processes in a previous, interactive version of the same system, leading to surprisingly musical output and convincing performances. The system is also capable of rudimentary learning, through recycling of its own musical output, and an accumulated database of human musical input.

Keywords: Musical composition, generative music, evolutionary algorithms, genetic representations, formalized fitness criteria

Introduction

Various formal methods have been used throughout history to create material for art. Music is perhaps especially well suited for such treatment because it is, to a certain extent, quantifiable and abstract. Examples of formal methods include strict musical forms such as canons and fugues, the serialist techniques of permutations and inversions, and various theories of proportions. The digital computer introduced a conceptual change since it provides a platform for almost any kind of generative process, and allows the systematic use of techniques that were previously unthinkable because of prohibitive time scales. Computer science and mathematics provide a large repertoire of algorithms that can be used to generate and process musical material, and various mappings of extra-musical data to musical parameters can easily be tried and evaluated in the search of new musical expressions and novel musical material.

Formal methods help the composer to extend beyond her current sphere of ideas by projecting from known ideas into unknown or unpredictable results. They expose her to unexpected material, and can thus help avoid artistic stagnation. Algorithms can also be used as tools to solve specific design problems, e.g., to design a transition from A to B. By abstraction into higher-level entities, the composer can concentrate on high-level parameters and delegate the details to the algorithm. The musical result is then auditioned, and the algorithm and its parameters are possibly revised, in an iterative process. The computer becomes an advanced creative assistant.

One family of algorithms that has proven their efficiency in creative applications are those inspired by natural evolution (Darwin 1859). Based on the concept of random variation and selection, evolutionary algorithms can efficiently search a large space of candidate solutions. By using a population of potential candidates, they search the space in parallel. In each generation, all objects are evaluated. The most fit survive, and are reproduced or recombined with some random variations to form the next generation. The procedure is repeated until a satisfactory solution is found, or until there is no more improvement. For an introduction to evolutionary algorithms in creative applications, see, e.g., Bentley (1999) and Bentley and Corne (2001).

The objects to be evolved in an evolutionary algorithm are represented in terms of data structures that contain all their inheritable information. This can be a binary string, a set of parameter values for a generative process, or even a computer program (Holland 1975, Koza 1992). Random variation is introduced during reproduction by genetic operators, among which the most common are mutation and

crossover. The first introduces random changes into the data, and the latter combines two genomes by merging parts of two or more parent genomes.

The choice of representation is crucial, since it defines the space of possible outcome, and together with the genetic operators it also determines the topology of the search space, i.e., which results are close to each other, and how the space can be traversed in the evolutionary process.

There are different classes of representations. In the case of musical score material, a basic representation would be a list of notes, with durations and velocities (e.g., Nelson 1993, Horner and Goldberg 1991, Ricanek et al 1993, Unehara and Onisawa 2001). In this form, almost any kind of musical material can be represented, and it is indeed the format of standard MIDI files. But it does not contain any structural information about the music. A mutation that changes a single note could, e.g., destroy internal relationships between thematic material. The next level in sophistication would be to have the individual events in a data structure that mirrors the underlying musical structure, such as a number of motives and their temporal and tonal relationships. Still better would be to generate the data structure through a generative process based on some given material, creating structures of related motives from a single stored motive combined with parameters that control the generative process (see Thywissen, 1999 for an example). Then, it is possible for low-level mutations to affect the musical result on a higher structural level.

Hierarchical data structures have the advantage of being flexible and open-ended, and it has been shown many times that music can be described as a hierarchical structure (e.g. in Schenkerian analysis, for an introduction see Jonas, 1983). Different hierarchical data structures have been used as genetic representations for music by, e.g., Johanson and Poli (1998) and Tokui and Iba (2000).

This paper describes a system that implements recursively described binary trees as genetic representation for the evolution of musical scores, with operators in each branching node and notes or motives in the leaf nodes. During the development process, the tree is evaluated in a number of steps into a flat list of notes with corresponding performance data, with the possibility of melodic, polyphonic and homophonic material. The representation is compact and flexible, and it can represent a wide range of musical structures. In particular, it will be shown that the recursive mechanism with its own set operators provides potential for a number of archetypal musical gestures, and at the same time generates plausible and expressive performances of the music.

This representation was originally created in 2000 for the interactive breeding of score fragments (Berry and Dahlstedt 2003, Dahlstedt 2004), but here we will focus on a later implementation, where it is combined with automated fitness evaluation, making it possible for the system to autonomously compose and perform more or less complete piano miniatures. The first version of this system was exhibited as an installation at Gaudeamus Music Week in Amsterdam, the Netherlands, in 2002. A new, improved version was developed in 2005.

Implementation

First, a few words on the general architecture of the system. The system in question was designed for an interactive musical installation consisting of a MIDI controlled grand piano and a hidden computer. A number of musical scores, represented by data structures, are evolved from an initial population over a number of generations until a required fitness rating is attained, or until a maximum number of generations is reached (due to time constraints). The best score from the last generation is performed on the grand piano and saved to disk, both as a MIDI file and as a genome file in a custom file format. About every three minutes a new piece is evolved and performed, each piece being between one and one-and-a-half minute long. When a visitor/user plays something on the piano keyboard, this music is immediately translated into a genome data structure and used as initial population for a new piece, which is immediately performed. In other cases, the initial population is based either on random generation of genomes, on recombinations of previously evolved scores, on recombinations of material given as a collection of MIDI files, or on the accumulated human input.

In the following, details of the implementation of the system are given, describing the data representation, the genetic operators, the evolutionary design choices and the formalized fitness criteria.

Recursively described trees as representation of musical structure

We have developed a representation based on recursively described binary trees. Using only two operators, a large variety of musical material can be represented. In addition, a recursive expansion mechanism allows the representation of very complex structures in compact form.

A genome tree consists of branching nodes and leaf nodes. Each leaf node contains a note or a musical motive in the form of a list of notes. In the branching nodes, operators merge or concatenate the notes into larger musical structures, represented as lists of notes. For each note, the following information is stored:

Onset time Measured from the beginning of the list.
Pitch Represented as a MIDI note number.
Amplitude The amplitude of a note.
Note duration The musical duration, i.e., the distance in time to the following note.
Articulated duration The actual duration of the played note. As an example, staccato notes are represented by articulated durations much shorter than the note duration.

A list can contain any number of notes, with a total duration measured to the end of the last note.

Most musical structures can be described using only two operators, *S* and *U*, which represent two basic ways of combining notes or sequences. The *S* operator (short for *splice*) concatenates two lists of notes *x* and *y*:

$$xSy \equiv x \bigcup y' \qquad (1)$$

where *y′* is a copy of *y* time-shifted by the duration of *x*. The total duration is the sum of the durations of *x* and *y*. The *U* operator (short for *union*) merges two lists of notes, so that they will be played simultaneously:

$$xUy \equiv x \bigcup y \qquad (2)$$

During the development phase, the tree is flattened into a list of notes, by evaluating each node according to its operator, from left to right. Rhythms are formed implicitly by the concatenation of durations. An example of the evaluation of a tree is shown in fig. 1.

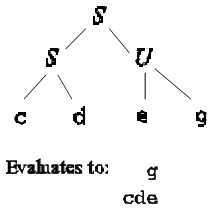

Fig. 1. A genome tree. Note durations and amplitudes are not shown. Below the graph is shown a simple notation of the playback result of the tree, to be read from left to right. *c*, *d* and a chord of *e* and *g* are played in sequence.

Branches of the tree can have recursive properties. A recursive branch is expanded to a more complex subtree during the development process. In this way, elaborate structures can be represented in a compact way.

The tree expansion mechanism works as follows. The tree is evaluated through a number of iterations, corresponding to the maximum number of recursive iterations requested by any branch in the tree. In each iteration, the whole tree is evaluated. A leaf node can contain a pointer to a branching node a number of steps higher up in the hierarchy, together with a recursion depth, i.e., the number of recursive iterations. In each iteration of the development process, the leaf node is replaced by a copy of the subtree starting at the pointed branching node. This subtree will also contain the original leaf node, in the end of the expanded branch. The process is illustrated in fig. 2, where the development of a small tree with one recursive branch is shown.

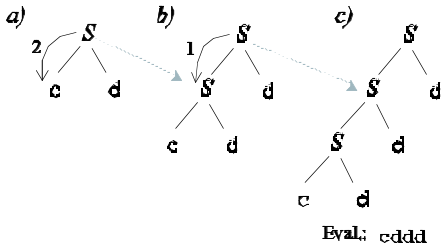

Fig. 2. A recursive genome tree. The recursion is indicated by an arrow and a number. In a) the original genome structure is shown. During the development phase (b-c), the left branch is recursively expanded by repeatedly inserting a copy of the tree from the previous time step (indicated by the gray arrow). The final result is obtained by evaluating tree (c). For simplicity only note names are shown in the diagrams. In the full representation, duration and velocity are also included.

In the tree expansion, the copied subtrees can be assigned one or more of three different operators that process the note list at that node:

A(x,α) multiplies all amplitudes in x by α, α>0.
D(x,δ) multiplies all durations in x by δ, δ>0.
T(x,τ) transpose all pitches in x by τ steps.

Since these operators are assigned repeatedly to subtrees, they will create gradual changes in performance parameters, such as exponential tempo and dynamics changes. The operator *A* applied repeatedly to a growing list generates an exponential diminuendo or crescendo. See fig. 3 for an example of this process. The development of a tree with multiple recursions is shown in fig. 4. Such nested recursion can create very intricate structures, e.g., transposed imitation with gradual variation.

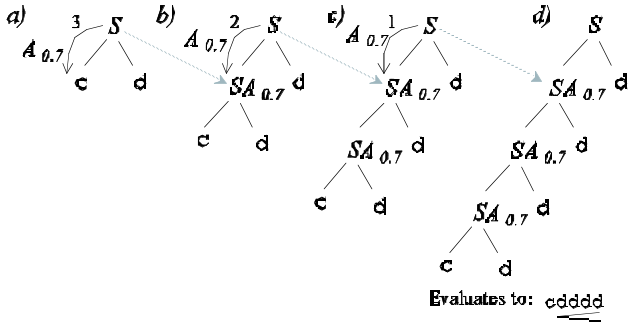

Fig. 3. A genome tree with recursion. The operator A multiplies all amplitudes in a note list by a constant. The recursive application of A results in an exponential crescendo.

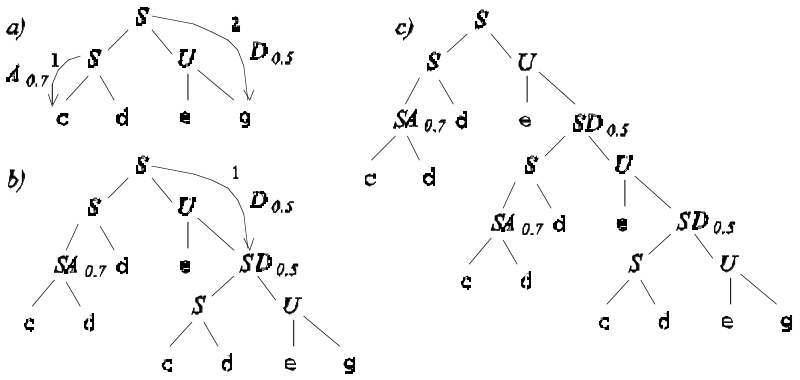

Fig. 4. A genome tree with multiple recursions. During development, subtrees are copied following the arrows, and the recursive level is decreased by one. The final result is obtained by flattening tree (c).

Genetic operators

As genetic operators, three kinds of mutations are used, combined with crossover implemented as a merge of branches from two parent trees, cut at random points in the tree structure.

The first kind of mutation is the random modification of an existing leaf node, where a note can be changed with respect to pitch, duration and amplitude, and recursion parameters can be altered. Second and third, branching nodes in the tree structure can be inserted or deleted. The probabilities of these mutations can be independently adjusted. The default setting is biased towards a slow increase in genome size, leading to gradually more complex musical structures. A single mutation can alter the whole structure of the music, introduce repetitions, processings or a single new note. In all cases, there will be a strong resemblance to the parent score fragment.

Initial population

The search space is very large, and the initial population will necessarily be small compared to the size of the space. In the interactive version, the population used was very small, due to the slow audition process. In an autonomous system this restriction is lifted, but there are still time constraints, since a new piece is expected to be performed every three minutes. Also, when human input is provided from the keyboard, a new piece should be performed virtually immediately. So exceedingly large populations are not possible. Usually, populations of between 25 and 50 objects have been used. For the same reasons, the number of generations must be limited. In the case of human input, not more than ten generations are computed, because a certain degree of aural similarity between input and the end result is conceptually important. In other cases, the computation is run for between 50 and 100 generations. More is not possible on a fast computer of today.

Under these circumstances, the initial population is very important, since it determines the starting points in the search space. In many cases a randomly generated initial population is appropriate, since it provides maximal variation and coverage of the search space. In other cases, for example when one wants to explore a certain neighborhood in the search space, previously evolved objects or human input may be more appropriate. In those cases, the initial population provides a means to indirectly affect the musical results. As the number of generations increase, this effect diminishes.

The system provides three mechanisms for computing initial populations: random generation, recombination from a collection of "good examples", and human input. They can be used separately or together.

Random generation Genome trees are generated based on a number of parameters. They control the allowed range of notes, the range of note durations, the distribution of durations (linear or exponential). Quantizing can be on or off, i.e., with quantizing on, only even divisions of the beats are used. Otherwise, all durations in the allowed range are possible. There is also a mechanism for using note and duration distributions from stored genomes. Another parameter controls the distribution between the *S* and the *U* operators, which in effect controls the proportion between vertical and horizontal structures in the music. Probabilities and ranges for the recursive mechanism affect the probability and character of the more intricate structures formed in the generative process (probability for recursion to appear, maximum recursive depth, maximum length of upwards recursive pointer). There are also probabilities for the recursive operators, *A*, *D* and *T*. With these parameters, more abstract properties of the music, such as repetitivity and plasticity, can be adjusted. The generative parameters that control the generation of new genomes also control the mutation of nodes, e.g., when new nodes and leaves are inserted into the structure, or when leaf node parameters are altered by mutation.

Recombination from stored genomes The initial population can also be created from stored genomes or MIDI files in a special folder that is searched every time a new population is created. This is also used to feed back successful genomes and human input into the evolutionary process, by accumulating the results. Each new genome is created by crossover of two stored genomes or MIDI files. In the case of MIDI files, they are first converted into genome trees by a straightforward translation process, which does not involve any high-level structural analysis other than differing between sequential and chordal structures.

Human input When music is performed on the piano keyboard, it is immediately translated into a tree genome structure, which is reproduced through mutations to form the initial population. A simplified evolutionary process is started, with only a few generations, not to end up too far away from the initial material.

Formalized fitness criteria

This representation is designed for the generation of contemporary musical material, and does not contain any information about keys, harmony, or traditional forms. Instead, it has the potential to generate complex atonal, rhythmic gestures based on any kind of pitch material. The recursive mechanism in the genome can generate a large number of musically relevant gestures, such as gradual tempo changes, musical dynamics, imitations and variations. Because of this, a large fraction even of randomly generated musical fragments are musically interesting from the start, and most of the outcome is gesturally and rhythmically consistent.

It is very difficult, if not impossible, to formally define aesthetic selection criteria, for several reasons. First, being *good* or *aesthetically pleasing* are very relative properties, depending on who is giving the evaluation. Second, the criteria may change over time, for the same person, depending on what one is looking for at the moment, and on the musical context. Sometimes smooth, beautiful music is appreciated; another time harsh and violent tunes fit our mood. Third, the target may not even be known, and the criteria may change when we are exposed to new unexpected material.

However, based on a couple of years experience from real-world use of the predecessor of this system, a number of conclusions could be drawn from the interactive selection process. The rate of musically useful material was very high, thanks to a data representation very well suited for contemporary musical structures. Hence, the selection procedure consisted mostly of ruling out the failures from weaknesses in the representation and to extreme, unusable material. The un-selected candidates mostly fell into the following categories: wrong duration (most often too long due to excessive or nested recursion), too high, too low, or too uniform note density, or too long sections with very little substance due to excessive recursion. As an experiment, a number of simple statistical measures were implemented based on these observations. Other similar measures could have been included, but the following collection was chosen to give reasonable control over the duration, density, character and dynamics of the music:

Density Average number of notes per second over the whole sequence. Too sparse or too dense music is generally not desirable. Our perception seems to be tuned to a certain range of information density, although this also depends on the relationships between the notes, which are not accounted for in this measure. The note density is also related to the concept of performability.

Minimum and maximum density The minimum and maximum local density appearing in the sequence. By choosing appropriate target ranges for these measures, dynamic change in density within a sequence is encouraged in the evolutionary process. Also, sequences with short bursts of very high density can be avoided. Similar results could be attained with a measure based on standard deviation of the density.

Repetitivity A measure of the number of repeated notes in the sequence. It is calculated as the total number of occurrences of similar notes (same note number) with a maximum list index distance of 5. This may or may not be a desired property in music, and gives rudimentary stylistic control.

Duplicity The number of notes in the sequence divided by the number of leaves in the tree (which equals the number of source notes that are used in the generative process). This gives an indication of excessive or nested recursion, often leading to very long passages of repeated notes. On the other end of the spectrum, it indicates too little recursion taking place, leading to sequences with little generated structure, essentially a flattened tree. In a sense, it is a crude measure of information content or randomness (disregarding the potentially meaningful musical relationships between the notes in the leaf nodes, which could be regarded as coincidental), which is possible since we have access both to the generative description and the actual result.

Silence The proportional amount of silence in the sequence, where silence is defined as where the density of notes per second is below a certain threshold, typically 0.5 notes/sec. This is to avoid sequences with too long passages where nothing happens.

Pitch variation The standard deviation of pitches over the whole sequence.

Duration The duration of the sequence in seconds.

For each measure, a rating is calculated depending on where the result is compared to a desired target range. Within the target range a rating of 1.0 is assigned. Outside, the rating decays by the square root of the distance to the target range. A weighted sum of the ratings for all components is the fitness value for the sequence. See fig. 5 for a graph of the fitness rating as a function of the measured value.

This is essentially a kind of negative selection, ruling out the failures and relying on the expressive power of the representation. If one of the measures is well outside the target range, the large negative value will almost disqualify it from selection.

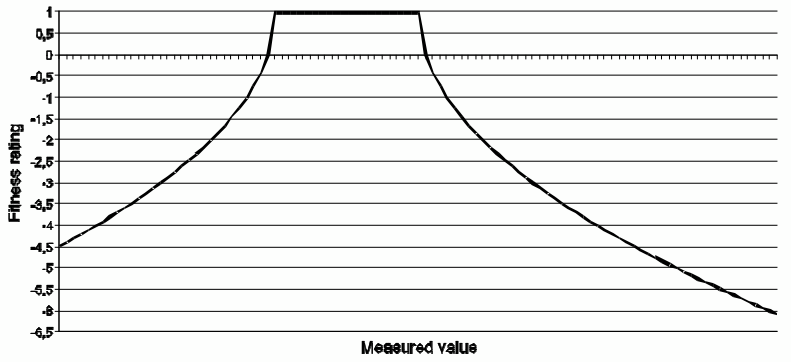

Fig. 5. The fitness rating for each statistical measure as a function of the measured value and the target range. Within the desired range, the value of 1.0 is returned. Outside the range, it decays by the square root of the distance from the range.

An example

The musical aspects of the evolutionary process, as well as the nature of the genetic operators, are better understood by example. In sound example 1*, the most fit sequence from 39 generations are played in succession. Only mutations were used as genetic operators, for clarity, and the duration of each piece was limited to about 15 seconds. The starting population was a single sequence of repeated quarter note middle C:s. Already in the first generation, a subtle accelerando and crescendo has been superimposed on parts of the note sequence, and some notes have been altered. In the third generation, scale structures and other, faster note repetitions have appeared, as a result of recursion. At several points, major transitions can be noted (e.g., at sequence 6, 8 and 12). These are most likely not the direct results of mutations, but instead illustrate that a different part of the population, which has developed in slightly different direction, takes over the lead. Sometimes, a transition is followed by another one that restores the previous strand (e.g., 13-*14*-15, 19-*20*-21, 31-*32*...-36). Remember that only the highest rated sequence out of 50 is shown. Taken as a whole, the progression from a simple repeated note towards rather complex musical material is obvious, and the latter part of the succession contains some rather complete music. In a normal session, the conditions are different. The initial population is more diverse, with more complex material; imported, previously stored or randomly generated. The allowed piece duration is much longer, and crossover is used to introduce further variation. And only the best result of the last generation is ever performed. Still, this example gives an impression of the inner workings of Ossia.

* The sound examples accompanying this paper are available at the author's homepage: http://www.ituniv.se/~palle

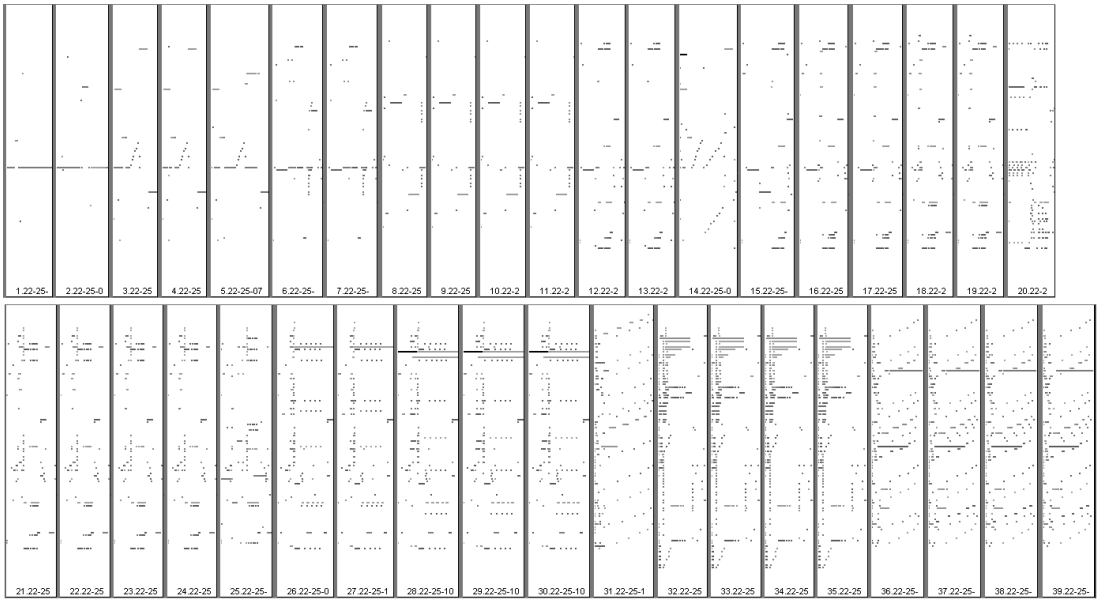

Fig. 6. Piano roll notations of the most fit sequence from each generation in a session of 39 generations, as heard in sound example 1. Each piece is about 15 seconds long, and the starting population was a simple sequence of repeated notes. The progression towards more complex material is clearly visible, and both gradual changes and major transitions can be seen. See the text for a more in-depth discussion.

Mechanisms to create variation between pieces

An example of a typical final output can be heard in sound example 2, with a rough transcription given in fig. 7. However, a single example says very little about the quality of a generative system of this kind. In the variation over a larger set of examples one can see if a system has potential to surprise the listener over larger time-span, or if the system quickly "dries out". Since this system was designed for a continuous installation, long-term variation was essential. The genetic representation has a vast scope in itself, but the end result is to a large degree influenced by the generative parameters that control the initial population and mutated nodes, and by the target ranges for the different measures of the fitness function. To maximize the variation, and push the evolutionary process in different directions, a simple mechanism was implemented.

Four complete sets of different generative parameters were created, with different rhythmic and tonal potential. Also, a number of sets of target ranges for the statistical measures for the fitness function were created, with different expected durations, densities and stylistic properties. For each new evolutionary process, one generative parameter set and one target range set are selected at random. Combined with the flexible mechanism for creating initial populations from previous results and human input, this gives the system a large potential for variation. The results can be anything from impressionistic meditations to aggressive, rhythmically pregnant statements (listen to sound examples 3, 4, 5, 6, 7, 8, 9, and 10 for further examples).

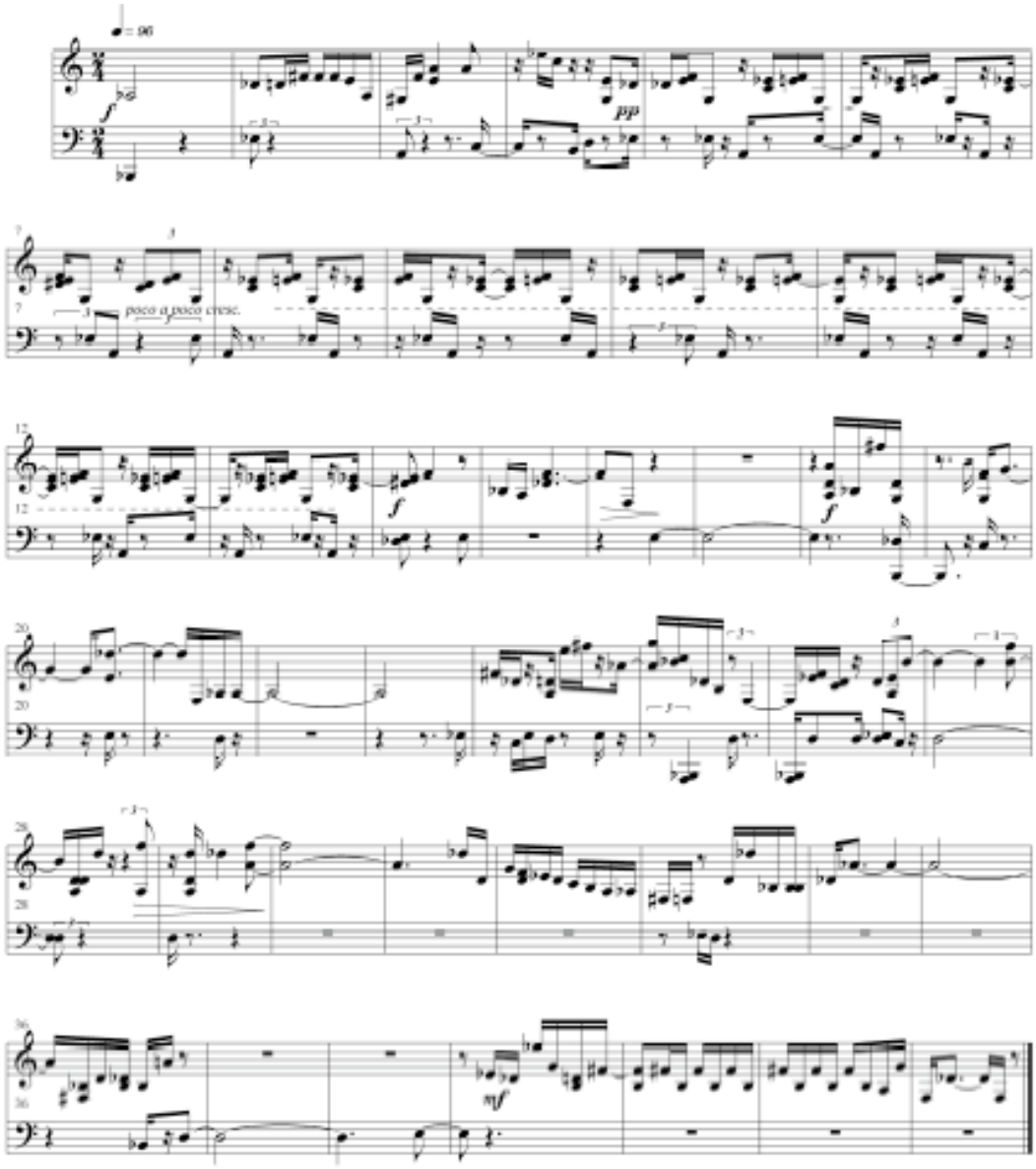

Fig. 7. A rough transcription of a piano miniature evolved from a random initial population. The rhythms are only approximate. The actual output from the system contains detailed dynamics and articulations, as well as subtle tempo fluctuations and rubato, which can be heard in sound example 2.

Discussion

Musical results

Music is difficult to describe in words, but the sound examples given hopefully show that the Ossia system has the potential to generate and perform piano pieces that could be taken for human contemporary compositions. They vary greatly in style and expression, and through carefully selected source material that form the building blocks for the initial populations, this variation can be pushed even further.

There are several reasons for the musical qualities of the output. The inherent potential of the recursive representation for the production of archetypal gestures, such as exponential diminuendo, crescendo, accelerando and ritardando explains why the music feels familiar and authentic. Even if the pitch material is sometimes arbitrary, these gestural qualities form structures that are perceptionally comprehensible.

The recursive mechanism also leads to clear thematic structures, where each section of the generated material is melodically and rhythmically consistent and homogenous. Each exposed idea or motive is exploited not too much, and not too little. This is explained by the limited tonal material given in the tree leaves, which is exploited and reused by the recursive processes on different levels, which creates themes or small-scale structures containing short-term repetition, or larger sections of recycled but varied material. Overlapping recursion in neighbor branches provides further interesting intricacies.

The different sets of generative parameters and fitness target ranges create an interesting friction. They concern different aspects of the music, on different conceptual levels. Together they affect the process from two directions, symbolizing the composer and the listening critic, respectively. One tries to fulfill the others expectations, by means and devices that are very uncorrelated. This discrepancy leads to interesting results.

The expressional power of this binary tree representation can be better understood if the formal class of possible outputs is defined. The output that can be described by this tree representation can also be described by a version of formal grammars called indexed context-free grammars, first mentioned by Aho (1968), which is a class of grammars with more expressive power than the ordinary context-free grammars. A recursive tree can be substituted by a set of production rules and a number of indexed variables. We have developed a translation scheme from the binary trees to grammar rules, and believe that this is a more logical representation, providing interesting possibilities for generalizing and extending the representation. These results will be published elsewhere.

Limitations

There are many limitations to the system, and it can certainly be much improved. For example, it does not incorporate much, if any, musical theory, in the form of theory of harmony or musical forms. This is intentional. To many rules will invariably limit the result, and here the musical qualities emerge from simple systems in a bottom-up process, without explicit knowledge. But this also detaches the result from a strong tradition. The music is not evolved in relation to any outside context, except the material that is used to form the initial population. The musical result has gestural and structural qualities in common with much contemporary music, but this resemblance is in a sense superficial. Computer generated music cannot be created (by the *program*) as a part of the general artistic discourse. The programmer/composer can and should take part in this discourse, but the internal process of the program is detached from the real world, and instead has its own struggle between composing algorithms and fitness evaluating critics. The program itself is part of the real-world discourse, but then this presupposes that the program is the artwork, and not the output, which is questionable.

There is also a lack of correlation between superimposed structures. The tree operators allow totally unconnected material to be superimposed without any regard to their contrapuntal, harmonic or rhythmic relationships. In practice this has not been a problem. The layered structures are often interesting, either because the layers blend very well, or the opposite – because they are clearly distinguishable and complement each other. Human perception has a remarkable capacity to create relationships.

One clear limitation in the music is the arbitrariness of the large-scale forms. The tree structure allows many different forms to emerge, such as gradual processes, repetition or imitation with variation, and juxtaposition of different material. The general concept of recapitulation is not possible, as in the common ABA form. This does not matter so much in short compositions, but may be limiting when evolving longer material. A more flexible type of recursive pointers could allow for greater flexibility in form, e.g., by pointing to other branches, and thus allow for the same material to appear in different parts of the composition. This would loosen the correlation between the layout of the tree structure and the temporal form of the result.

Performance qualities

Since performance parameters are integrated into the genetic representation, and are processed and generated by the same mechanisms that generates the note structures, the performance is inseparable from the composition. When for example a repetition is created by the recursive mechanism, e.g., by repeating a single note in a scale, this very same note is also subject to dynamical and temporal operators, leading to tight integration between compositional structures and performative expression.

This inseparable nature of compositional and performative aspects of the output creates direct and convincing “interpretations" of the pieces, even though there is no interpretative phase in the algorithm.

But this also makes the musical result difficult to transcribe into traditional notation, since the temporal subtleties are difficult to squeeze into the rigid system of rhythmic notation, and the fine details of dynamics and articulation get lost when transcribed. So the results are better heard as performed by the system itself.

Variation and learning

Thanks to the flexible representation and the different generative parameter sets and target ranges, the musical output is very varied. It can also vary over time, since the system provides means for rudimentary learning, in the form of recycling of its own output, and accumulation of human input. External parallel processes, such as web-based search scripts, have also been used at runtime to collect fresh material for the database, providing further stylistic variation over time.

However, the statistical measures that form the basis for the fitness evaluation can certainly be developed much further. In their present form they touch upon important properties from information theory, such as complexity, information content and randomness, but this connection is mostly in intent, and real implementations of such measures should be implemented in future versions. Also, the fitness criteria, although variable through different parameter sets, are static. They could, in principle, be evolved in a separate process, or coevolved with the music (for an effort in that direction, see, e.g., Todd and Werner, 1999). However, the author strongly believes that a system that is intended to produce music for human ears has to retain a connection to human aesthetic criteria. If "compositions" and "critics" were evolved in a closed system, they would most certainly find a way to coexist in that system, but what works for them there is not certain to be interesting music for us. Any such system would still need an ad hoc mechanism to steer the evolution towards musically interesting material, and we are back where we started.

Is it creative?

If this work is to be evaluated according to McCormack's "open problems" of evolutionary music and art (McCormack 2005), it is surely one more of those ad-hoc systems that rely heavily on the creativity of its creator/programmer. At the same time, it uses an open-ended genetic representation, which in essence is a generative description. It poses limitations on the output, but these limitations are not as severe as in the case of, e.g., the evolution of parameters for a fixed generative process. And it can be claimed that the musical results in a certain way satisfies McCormack's open problem #3, i.e., how to create systems that produce art recognized by humans for its artistic qualities. The musical results of this system, both compositional and performance qualities, have received very positive response from both general audiences and from audiences of "paper-based" composers and traditional musicians (who are both generally quite hostile to generative music). It has been performed without any explanations of its provenience, and gained acceptance of the music on artistic qualities alone. However, as a Bach fugue may be beautiful at first hearing by an uneducated ear, deeper knowledge may induce another level of appreciation.

In the actual application, the recursive representation is implemented not as a literal tree expansion, but as iterated recirculation of temporary results from branch nodes into leaf nodes, which is equivalent. This recirculation of already processed material into the generative mechanism points to an important property that real-world creative systems exhibit: to allow for coincidences to be picked up and propagate into the creative process. The generative process must in some way take into account the actual outcome of its own process. This is also discussed by McCormack, although in different terminology. Since in this system the generative tree structure is evaluated and flattened into a list in every node in each stage of the development phase, small motives created not explicitly by the representation but generated implicitly by the merging of neighboring branches are picked up and processed by next recursive iteration. In essence, this represents a collapse of hierarchies, and leads to complex and interesting results. This feature may be further improved, but it is believed that it even in its relatively crude implementation here contributes to the structural qualities of the music.

Conclusion

This paper has described a system that autonomously evolves and performs complete piano pieces. The musical output is structurally complex and musically convincing, and the performances sound "authentic", vivid and musical. This is due to the combination of a powerful and adequate

representation with selection criteria that rule out the failures. The evolutionary process can hardly be regarded as an optimization process, because of the limited population size and the small number of generations. Based on observations of the evolutionary progress in a number of sessions, the process is better described as starting at a number of random points in the search space, extracting those that prove fruitful, and refine them into a decent piece of music in a relatively low number of steps. The system is not suited for the generation of specific kinds or styles of music, but it generates very interesting novel material and exposes a wide range of variation. It will find interesting music, but it is difficult to control exactly what kind of music.

On a number of points the system is primitive and crude, but the results are nevertheless valuable, and it deserves to be further developed. The insights are currently applied to the implementation of a new, generalized version of this representation, which will hopefully be applicable to artistic material of many different kinds, both visual and musical.

To conclude, the combination of a generative recursive representation and autonomous negative selection has proven a fruitful one. Very seldom is the musical output a complete failure, and often it is very interesting, and could often be taken for a contemporary composition – which it is.

References

A. V. Aho. Indexed Grammars --- An Extension of Context-Free Grammars. Journal of the ACM, 15, 647--671, 1968.

P. J. Bentley (ed.). Evolutionary design by computers. San Fransisco, CA: Morgan Kauffman, 1999.

P. J. Bentley and D. W. Corne (eds.). Creative Evolutionary Systems. San Fransisco, CA: Morgan Kauffman, 2001.

R. Berry and P. Dahlstedt. Artificial Life: Why Should Musicians Bother?, in Contemporary Music Review, 22:3, 57-67, 2003

P. Dahlstedt. Sounds Unheard of: Evolutionary algorithms as creative tools for the contemporary composer, doctoral diss., Chalmers University of Technology, 2004.

C. Darwin. On the Origin of Species by Means of Selection, or the Preservation of Favoured Races in the Struggle for Life. London: John Murray, 1859.

B. Degazio. The Evolution of Musical Organisms. Leonardo Music Journal, 7, 27--33, 1997.

J. H. Holland. Adaptation in Natural and Artificial Systems. Ann Arbor, MI: The University of Michigan Press, 1975.

A. Horner and D. E. Goldberg. Genetic Algorithms and Computer-Assisted Music Composition. In Proceedings of the 1991 International Computer Music Conference (ICMC91), pp.479--482, 1991.

D. Horowitz. Generating Rhythms with Genetic Algorithms. In Proceedings of International Computer Music Conference 1994 (ICMC94), pp.142--143, Aarhus, Denmark, 1994.

B. Johanson and R. Poli. GP-Music: An Interactive Genetic Programming System for Music Generation with Automated Fitness Raters. In Genetic Programming 1998: Proceedings of the Third Annual Conference, pp.181--186. San Fransisco, CA: Morgan Kaufman, 1998.

O. Jonas. Introduction to the theory of Heinrich Schenker : the nature of the musical work of art. Translated by John Rothgeb. New York and London: Longman, 1982.

J. Koza. Genetic Programming. Cambridge, MA: MIT Press, 1992.

G. L. Nelson. Sonomorphs: An application of genetic algorithms to the growth and development of musical organisms. In Proceedings of 4th Biennial Art & Technology Symposium, pp.155--169, Connecticut College, CT, 1993.

J. McCormack. Open Problems in Evolutionary Music and Art, in F. Rothlauf et al. (Eds.): EvoWorkshops 2005, LNCS 3449, pp. 428–436, 2005.

K. Ricanek II, A. Homaifar, and G. Lebby. Genetic algorithm composes music. In Proceedings of The Twenty-Fifth Southeastern Symposium on System Theory (SSST'93), pp.223--227. Los Alamitos, CA: IEEE Computer Society Press, 1993.

K. Thywissen. GeNotator: an environment for exploring the application of evolutionary techniques in computer-assisted composition. Organised Sound, 4, 127--133, 1999.

P. Todd and G.M. Werner, Frankensteinian methods for evolutionary music composition in Griffith, N. and P.M. Todd, Musical networks : parallel distributed perception and performance. 1999, Cambridge, Mass.: MIT Press

N. Tokui and H. Iba. Music Composition with Interactive Evolutionary Computation. In Proceedings of 3rd International Conference on Generative Art (GA2000), Milan, Italy, 2000.

M. Unehara and T. Onisawa. Composition of Music Using Human Evaluation. In Proceedings of the 10th IEEE International Conference on Fuzzy Systems, vol. 3, pp.1203--1206. Melbourne: IEEE Press, 2001.