Generative Art 2002

Can a Genetic Algorithm Think Like a Composer?

Andrew Gartland-Jones
Computational Creativity Research Group, The University of Sussex, Brighton, UK.
e-mail: drew@atgj.org

Abstract

There has now been a substantial body of work utilising Genetic Algorithms (GA) for the purpose of musical composition. A common point of discussion is how far GA's can simulate not just the musical output of human composers, but also the *process* of composing itself.

This paper begins by discussing the suitability of using a GA for composition, and goes on to describe a generative music system (by the author), that utilises a domain specific, knowledge rich GA. The system acts on a *supplied* 2-bar musical phrase (up to 4 parts), and evolves musical fragments towards a *supplied* target. The aim is to provide interim points on the evolutionary path, which represents a 'new' musical ideas audibly based on the supplied fragments.

The paper concludes that the system is able to model at least part of the creative *process* of composition, and is effective at producing musically successful results. (Audio download sources of its output are included to support this conclusion).

The system was used to generate music included in an interactive installation work, exhibited at Brighton Arts Festival 2002, and other applications under developed that use the algorithm are discussed.

1. Introduction and Aims

This paper is a musical and technical report on the development of an algorithmic composition system. The aim was to develop a system that would take in supplied musical fragments and generate *new* musically related fragments in response. An additional requirement was that processes used to achieve this should be modelled as closely as possible on common processes used by human composers.

2. Composition and Genetic Algorithms

Many mechanisms have been employed in the task of algorithmic composition, from the stochastic processes utilised in (possibly) the earliest example of computer generated composition: *The Illiac Suite* 1955 [1], to the more recent application of cellular automata by composers such as Miranda [2].

I decided to focus on the use of a GA model for two reasons: the increased use of GA's in generative artistic systems, and the resonance the simple GA model created with at least part of the processes used in my own (non-generative) compositional practice.

A Genetic Algorithm is a computer search technique, which derives its behaviour from some

of the mechanisms found in biological evolution. This is done by the creation in a computer of a population of individuals represented by chromosomes, in a similar way to the chromosomes that we see in our own DNA. The individuals in the population undergo a process of simulated 'evolution', including mutation, crossover, and selection, in order to evolve a solution to the problem in hand.

GA's were originally developed as a 'blind' search tool. Indeed, a large part of their attraction has been the small amount of application specific knowledge required to achieve a goal. It was clear from the outset however, that in order to utilise evolutionary computing techniques to achieve the desired musical results a simple GA model would have to be specifically adapted to a more musical function.

This section deals with the relationship between the GA model and common human compositional processes. The paper then goes on to describe in more detail the devised algorithm and its applications.

2.1 What Aspects of Musical Creativity are Sympathetic to GA Simulation?

From a process perspective, musical material is often thought to be generated in one of two ways. Either ideas 'appear', or they are developed, which really leads us to the commonly used creative model of inspiration vs. perspiration. In trying to model musical creative activity we should at least consider these two approaches; how do they relate, and what aspects of either may be successfully modelled by an algorithmic system.

An often-used example of musical 'inspiration' is Mozart's ability to conceive entire works in an instant, and a well-known 'perspiration' example is Beethoven, who left evidence of his constant exploration and reworking in his many note books. This is succinctly summed up by Jacob [3]

> "In short, creativity comes in two flavours: genius and hard work."

Attempting to model inspiration, which in this context I mean 'instantaneous production', would be a daunting, and probably impossible task for many reasons, not least of which being the lack of current understanding behind such psychological mechanisms. It is possible that inspiration is a trick of cognition (in the sense that it may turn out to be identifiable as the result of a process of sub-conscious 'perspiration'), or even historical documentation. What maybe commonly seen as a difference between inspiration and perspiration may in fact be a distinction between composing and scoring. Rachmaninov was known to compose entire works in his head before committing them to paper, but because of the accounts we have which describe how he would take long walks in the country, humming and tapping his fingers in musical contemplation, it seems clear that there was still a process happening over time.

So, if creative inspiration is not clearly understood, and in addition the aim is to model compositional processes in some way, it appears sensible to concentrate on how musical ideas develop during identifiable and observable creative activities.

A commonly used compositional process may be described as taking an existing musical idea, and changing it in some way. Musicians in various styles and genres may follow the process

differently, some through improvisation, others through pencil and paper, but what is most often practiced is taking an existing idea and *mutating* it in to provide a new idea. In fact mutation is closely related to notions of development, which lie at the heart of western musical concepts of form and structure. It may even be possible to see development as *directed* mutation.

In this model of [idea $\Rightarrow$ mutation $\Rightarrow$ new idea] we see three aspects to the creative act:

1. Creating/selecting the initial starting idea
2. The process of mutation itself
3. Assessment of the results of mutation.

With the core elements of GA's being mutation (including cross-over) and fitness assessment there appears to be a strong correlation with at least some aspects of the human *process* of generating musical material.

Do we escape the difficulties of ill understood inspiration mechanisms by using this model? There are still inspirational aspects at play when human composers follow variations on this process, primarily in the nature of applied mutations (the mutations are likely to be highly directed), and the act of selection or assessment. GA's are therefore no better or worse than any other algorithmic solution with regard to fundamental issues of artificial creativity, but because their process has some correlation with human compositional processes, they may make *integration* with human composers, in the form of creative support tools, more successful.

2.2 Composition as Exploring a Search Space

GA's were developed primarily as an *optimised* search tool. Even if there is a correlation between the mechanisms of a GA, and the (human) compositional process of [idea $\Rightarrow$ mutation $\Rightarrow$ new idea], can the aims of composers, who as a group are not usually happy to accept the most efficient solution over the most interesting, be met by mechanisms primarily devised for optimisation? If interesting pathways are to be found, any traversal through a search space should be guided in some way.

"...creativity is not a random walk in a space of interesting possibilities, but...directed".
Harold Cohen [4].

How then, is the search to be directed? The direction taken by a GA at any given time is determined primarily by the nature of the mutation and crossover operators, and the mechanisms for selection. Due to the clear importance placed on selection I will discuss this aspect in more detail later.

In a standard GA the mutation and crossover operators are random and take place on a simple bit array genotype. The bit array obviously has a task specific representation in some part of the system, but the actual manipulations take place on the *abstraction*, causing them to be more useful in a generic problem space but less useful in specifically *musical* manipulations. In order to make the GA more 'musical', two important changes are made to the algorithm with regards to mutation and crossover. Firstly, the phenotype was made less abstract, and more closely related to the problem space (the actual phenotype used in the system is discussed below), and secondly the mutation/crossover operators are designed with musical

processes in mind. The algorithm makes *musical* mutations to *musical* genotypes.

2.3 The Fitness Function Problem

Much of the work previously undertaken in the area of using GA's for composition focuses on the problem of assessing fitness. Often determining what is 'good' about a particular genotype is equated to deciding what is 'good' music in a general sense.

There are two common approaches to assessing fitness:

1. Use a human critic to make the selection - an Interactive Genetic Algorithm (IGA). This requires an individual to use all the real world knowledge they possess to make decisions as to which population members should be promoted into the next generation.

2. Automatic Fitness Assessment. Which means encoding sufficient knowledge into the system to make fitness assessment automatic after each mutation and crossover process.

The problem with using an IGA, is the time taken to make assessments, described by Werner and Todd [5] as the "fitness bottleneck". This describes the difficulty in trying to assess many possible musical solutions due to the temporal nature of the medium. The benefit however, is that we are not required to develop a formal rules base. With Automatic Fitness Assessment we escape the fitness bottleneck, but face the daunting task of encoding the essence of the music we want to encourage.

In fact what we often see in examples of GA music are two distinct goals being suggested, if not explicitly articulated:

1. Creating 'original' music, where the aims are more closely aligned with the subjective process of conventional composition. When a composer sits down in front of a blank sheet of paper they don't normally feel the need to explicitly define 'what music is' before notes are placed on the page, even though what they write may in fact contribute to, or even extend, such a definition.

2. The 'objective' validation of a given rules-set, as well as other aspects of the algorithm, by providing compositional output as a test. In effect asking, 'how good are my rules'. Examples of this work include Wiggins [6], where the output is designed to be assessable 1st year undergraduate harmony exercises.

The primary difference of course is that the first goal is subjective and creative, whilst the second aims to be objective and may be seen as principally a musicological, or music-analytical task. I am not suggesting that the two goals are mutually exclusive, as most music students know, merely that much previous work tends to emphasise one or the other.

Baring in mind my previously stated creative goals, and my desire as a composer to map out a search space, I decided to supply the algorithm with a starting musical fragment to evolve from, and a target musical fragment to evolve towards. See fig. 1 below.

This has the effect of both directing the search and making fitness assessment significantly

easier. The fitness function can then be described as the *similarity* of a specific genotype to the provided target music. The nature of the similarity test is described below.

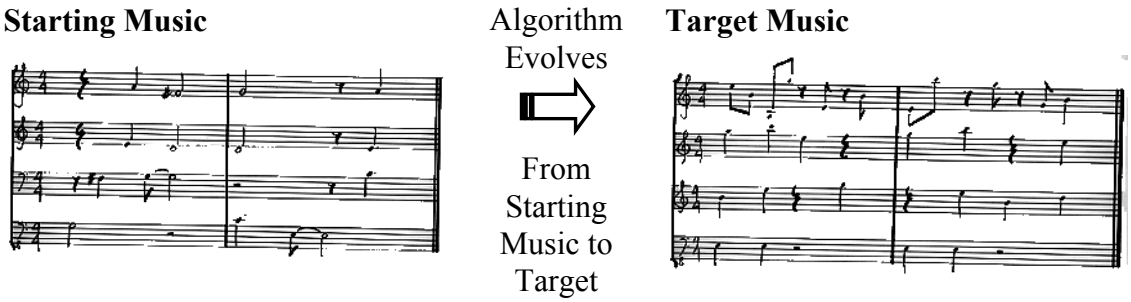


Fig.1 - Evolving from supplied starting musical section, to supplied target musical section.

3. The Algorithm

In this section I outline the steps of the developed algorithm, paying particular attention to describing the implementation details of the phenotype, fitness assessment and mutation operators.

3.1 Steps of The Algorithm

The individual steps of the algorithm are detailed below:

1. **Create initial population**

   A supplied 2-bar, 4-part MIDI[1] file provides the input to the algorithm. This is used to create a population of identical genotypes, which comprise the initial population.

2. **Select a population member (in turn) and perform mutation operators and crossover**

   As described above, rather than 'blind' bit flipping, the mutations have been modelled on processes used in musical composition, from simple addition and deletion of notes, to transpositions, inversions and reversals. Moreover, the application of any operators that require the selection of new pitches is based on a harmonic evaluation of the musical target.

3. **Assess the fitness of the mutated population member**

   Fitness is defined as how similar the genotype is to a supplied target. The details of this are discussed below.

4. **Accept into the population or discard**

   If the fitness value for the mutated population member is higher than the lowest fitness found in the population, it replaces the low fitness population member, and is stored as a

[1] Musical Instrument Digital Interface

musical point on the evolutionary path to the target, otherwise it is discarded.

5. Repeat until the target is reached

All the musical sections produced, representing the evolutionary path, are then available as MIDI files. In effect these are an audible journey from the starting music to target music.

3.2 The Phenotype

Rather than represent musical attributes as a bit array, a simple musical object model was chosen. The Phenotype also contains meta-data attributes, which hold fitness values for each genotype, down to the granularity of a single note. A simplified object model for the phenotype is described below:

Section - The Section object is the outer object representing a fragment of music. It has attributes for target match fitness, and collection of notes (a table holding individual Note objects – its dimensions are 4 by 32, representing 4 parts to a resolution of a semi quaver).

Note – The note object represents the values for a single note, and has attributes for *velocity*, *pitch*, *duration*, and *target fitness* (0/1).

3.3 Fitness Assesment

Fitness of a genotype (musical section), is assessed on how *similar* it is the target.

It was stated above that each Note object in the genotype has metadata holding its fitness value. If this note has the same pitch as a note in the same position in the target musical section, it has a fitness value of 1, otherwise the value is 0.

Overall genotype fitness is defined as:

$$\textbf{Musical Section fitness} = \textbf{nSection} / \textbf{nTarget}$$ where,

nSection = number of Notes in Musical Section with a Fitness of 1
nTarget = number of Notes in the Target

It is therefore, simply the proportion of notes in the target section that have a matching pitch in the genotype.

3.4 The Mutation Operators

As discussed above, the system is required to evolve musical fragments towards a target, *and* generate musically interesting results on the way. These are potentially conflicting, multi-fitness functions, and the mechanism chosen for resolving this is to not assess musical interest directly, but include processes through the mutation operators that are more likely to provide a musically successful output.

The mutation operators used (presented in the order applied) include:

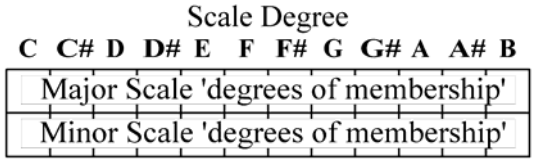
1) Add a note
2) Swap two adjacent notes
3) Transpose a note pitch by a random interval
4) Transpose a note pitch by an octave
5) Mutate the velocity (volume) of a note
6) Move a note to a different position
7) Reverse a group of notes within a randomly selected start and end point
8) Invert notes within a randomly selected start and end point. (Inversion around an axis of the starting pitch).
9) Mutate the Duration of a note
10) Delete a note

Each of the mutations listed is applied according to a user specified probability, and only to individual, or groups of notes, if the application of the mutation will not in any way alter notes that already have a targetFitness value of 1.

The fitness of each genotype is recalculated after *each* mutation. Therefore if a mutation has resulted in a note having a fitness of 1, it will not be mutated further. This optimisation of mutations enables the evolution process to be efficient enough to withstand interesting musical manipulations and still move towards the target.

3.5 Selecting New Pitches

When the system is started a harmonic analysis of the target musical section is made. A table is constructed (see fig. 2 below) in which each index location represents the target's 'degree of membership' of that scale type (Major and Minor), for that scale degree (C to B).



degree of membership = number of pitches present in scale/number of pitches

Fig. 2. This shows the table representing the 'degree of membership' of the target analysed. The table has values for only major and minor, but could be extended to include other scale type.

These analysis values, in conjunction with user-supplied weightings, are used to determine the *probability* as to which scale (eg. C# major) a new pitch should be drawn from when transposing a pitch within an octave, or creating a new note.

Once a scale is selected, the final pitch within that scale is chosen using a similar process. Now however, instead of the target music being analysed, the table is populated from an analysis of the other notes in the musical section being mutated. This is done for the *part* that the note is present in (so tending to create *melodies* from the same scale type), and the *chord* that this pitch will become part of (tending to create *harmonies* from the same scale type).

The degree to which this pitch selection process is applied maybe controlled by the user, giving the system a type of harmonic 'dial' that can be turned up to give tight constraints, or down to produce 'free' harmonies.

This rough estimation of harmonic analysis allows new pitches to be selected that are more likely to be present in the target music, but with enough fluidity that new harmonies will often occur. Of course, a more conventional harmonic analysis could be implemented, but this solution offers a more flexible harmonic model than methods associated with historical, *functional* (motion directed) harmony.

3.6 Crossover

After all mutations have taken place a random crossover is applied to the population, whereby note patterns are exchanged between population members.

3.7 The System is Interactive

Many aspects of the algorithm can be altered during running, including setting the probability of the application of each mutation operator (and changing parameters used to control them), use of crossover, and the degree to with harmonic analysis is applied.

4. Comments/Conclusions and Next Steps

The system certainly uses processes more closely related to those of a human composer than many other examples of GA composers. And it does, at least to me (and many of the installation visitors), appear to produce 'musical' output, which is clearly related to the supplied fragments.

It is interesting to note, although fairly predictable, that if all the harmonic restrictions described here are applied, the music produced is musical, but safe and lacking in novel ideas. As harmonic restrictions are released the music becomes more interesting, but with a greater tendency towards harmonic and melodic 'noise'.

It does of course, have serious limitations that restrict its usefulness. Chief among these is the small size (2-bar, 4-part) of the individual musical fragments that comprise its output. Although many of these are produced on each evolution run (usually ranging from 50-200), and it may be possible to argue that played in sequence they could represent a valid piece in the style of ‘process’ based musical forms associated with minimalism, I find simply playing the output stream in turn musically unsatisfying. It is the first stage in a 'bottom-up' compositional model, and was designed to work in conjunction with another system, see ‘Music Blox’ below.

Other possible next steps for further development include:

- Assessing fitness duration and velocity as well as pitch.
- Fitness evaluation could be extended beyond single yes/no note match, and could include an analysis of note patterns, which could be graded as closer or further away

from the target. For example if an inversion of a melodic pattern found in the target is created this may result in an increase in fitness for that genotype, exact matches of a melodic pattern would score higher etc.

- I have asserted that the mutation operators are based on some of the processes I find myself using when developing musical ideas, but an obvious next step would be to research the compositional processes of others, indeed to further explore existing research into general creative processes. Mutation operators could then be made both more suitable for an individual composer, and more complex as a combination of smaller elements.

5. Use of the Algorithm

5.1 nGen.1 – Interactive Public Art Installation

The system was used to produce the music for an interactive installation 'nGen.1' exhibited as part of Brighton Festival 2002, UK, and funded by South East Arts. The algorithm was run four times with the same starting and target MIDI files. Different parameters for the evolution were applied on each run, including the position of the harmonic 'dial'.

The resulting output MIDI files were taken at the 20%, 40%, 60% and 80% points (towards the target music). Therefore the four runs of the algorithm each produced four musical sections.

The sixteen MIDI files were octave transposed in order for them to be playable on the guitar, and each was recorded. Recording the output on a 'real' instrument enabled the perceived musicality of the fragments to be brought out, and provides additional musical dimensions.

For the installation, ultrasound sensors were placed on the wall and used to map out a virtual space for the sixteen recordings to be placed within, as shown in fig.3 below.

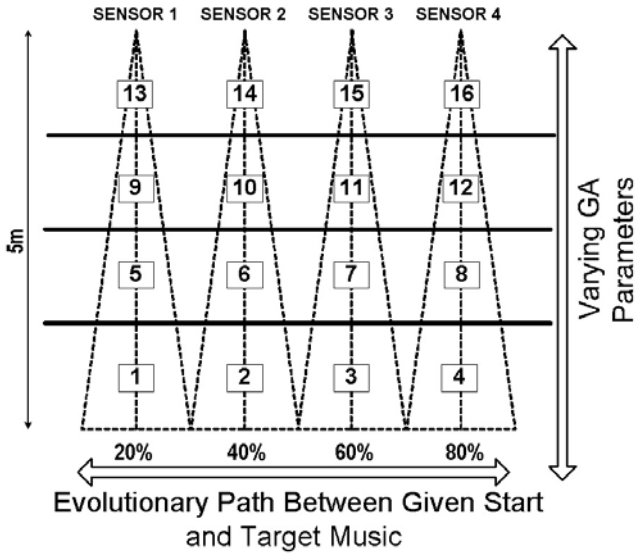


Fig. 3 – top view of area mapped out by ultrasound sensors

As people moved in front of the sensors the recording of an evolved musical section was heard.

The sounds in zones 2, 6, 10 and 14 are therefore all 40% evolved towards the target, meaning they have 40% of the pitches of the target. The evolution further away from the sensors uses stronger harmonic constraints, becoming more freeform towards the sensors. This creates a kind of musical logic to the ordering in the space, based on variation. Along the evolutionary path the variation is from starting to target music, towards the sensors it is variation of evolutionary parameters.

A simulation of walking along an evolutionary path (one with harmonic constraints, one without) may be downloaded at http://www.atgj.org/drew/, following the link to ‘Musical Examples’, ‘Example of OriGen Output’.

5.2 Live GA Accompanist

A version of the algorithm has been implemented as a live accompanist.

In this application the performer produces MIDI messages, either directly from a MIDI instrument, or as a result of an audio to MIDI conversion package. The system then records 2 bars of the performers MIDI output, and uses the *last-but-one*, and *last* 2 bar sections recorded as the starting and target sections supplied to the algorithm. When the required fitness has been reached (default is 50%) the output is either played back directly, or sent to a scoring package for another live performer to read. The overall effect is therefore a recomposed delay of performer one played by the machine or performer two.

5.3 Music Blox

Music Blox is the primary reason for developing the algorithm, but is still in development at the time of writing.

The MusicBlox project uses blocks, similar in size and shape to children's wooden building blocks, which are combined together in a similar way to make physical structures. However, each block has the ability to play and compose music, so building a physical structure results in creating a piece of music.

Imagine you have a single block. You give it a small musical fragment (its 'home' music), put it on the table and start it off playing. The box then starts repeating its musical phrase, either sporadically or continuously. Make another box, this time with a different piece of 'home' music. Place it next to the first box and they start playing together, and in synch.

Now press a button on block 1 labelled 'send music'. This causes block 1 to send its music to block 2. Block 2 then performs a composition activity, based on its own 'home' music, *and* the music it has just been passed by block 1. The compositional aim for the block is to produce a *new* musical section that has relationships to both its home music and the music it has just been passed. Block 2 then starts playing its new music.

Now image a chain or group of several blocks in any 3D structure. If a block is passed some music it recomposes itself, then passes its new music on to *all* of its neighbours.

Block 1 sends its music to Block 2 – Block 2 re-composes itself.
Block 2 sends its new music to Block 3 – Block 3 re-composes itself.
Block 3 sends its new music to Block 4 – Block 4 re-composes itself. Etc….

All blocks have a 'send music' button, so the start of the chain does not have to be block 1.

By sending out music from a starting point all other blocks within a specified range recompose, and the collective music of the structure is transformed. The development of the overall piece of music will therefore be determined by how the blocks are built into structures, how these structures are changed over time, and how the user sends music around the structure. It is important to clarify that each block has its own 'home' music that it holds onto throughout. This enables any music composed by a block to remain related to the initial music supplied to it, despite the constant process of re-composition each block undertakes. In this way the composer of the home music for all blocks maintains a compositional thumbprint on the evolving musical structure.

In effect the listener/performer is able to shape the overall music by choosing to send musical fragments from boxes they like to *influence* other boxes. One part of the structure may have composed some ideas the user likes. A block from that group could then be placed in another part of the structure to see what effect it has.

Although the aim is to have physical blocks to build with, the present prototype exists only as a 3D graphical model.

6. Acknowledgements

I would like to thank: Prof. Phil Husbands and Dr. Adrian Thompson for ongoing information, advice and general support, Theresa Gartland-Jones for a successful collaboration, and South East Arts for funding the installation.

References

1. Hiller, L, & Issacson, L. Musical Composition with a High-Speed Digital Computer, 1958, *Journal of the Audio Engineering Society*.
2. Miranda, E,R. On the Origins and Evolution of Music in Virtual Worlds, in *Creative Evolutionary Systems*, ed. Bentley, P, Corne, D, Academic Press (2002).
3. Jacob, B. L. Algorithmic Composition as a Model of Creativity, in *Organised Sound, vol. 1, no. 3, pp. 157-165*. Cambridge University Press. December 1996.
4. Cohen, H. A self-Defining Game for One Player: On the Nature of Creativity and the Possibility of Creative Computer Programs. *Leonardo*, Vol. 35, No1, pp. 59-64, (2002).
5. Werner, G.M., & Todd, P.M. (1998) Frankensteinian Methods for Evolutionary Music Composition, *Musical Networks: Parallel distributed perception and performance.* Cambridge, MA: MIT Press/Bradford Books.

6. Wiggins, G. Papadopoulos, G. Phon-Amnuaisuk, S, Tuson, A. Evolutionary Methods for Musical Composition. *Proceedings of the CASYS98 Workshop on Anticipation, Music & Cognition, Li`ege*, (1998).