Exploring sound-space with interactive genetic algorithms.

Colin G. Johnson.

Contact address:

Colin G. Johnson.

Computing Laboratory.

University of Kent at Canterbury.

Canterbury, Kent, CT2 7NX, England.

Telephone: +44/0 1227 827562

Fax: +44/0 1227 762811

Email: `C.G.Johnson@ukc.ac.uk`

Exploring sound-space with interactive genetic algorithms.

Colin G. Johnson.

Colin G. Johnson (computer scientist, composer).

Computing Laboratory.

University of Kent at Canterbury.

Canterbury, Kent, CT2 7NX, England.

Email: `C.G.Johnson@ukc.ac.uk`

Abstract

This article describes a system which uses evolutionary computation to provide an interface to a complex sound synthesis algorithm. The article then considers a number of general issues which need to be considered when evolutionary computation is applied in artistic domains, and the differences between interactive and non-interactive genetic algorithms.

1 Introduction.

Algorithms for the production of sound on computers are usually very complex, requiring the specification of a large number of parameters to generate each sound. Users who understand the detail of a synthesis algorithm can construct sounds by working out how the desired sound can be created from the components of the algorithm. Other users will learn to program a synthesizer in a more empirical way, by studying programs written by others, listening to their sound, and constructing their own sounds by modifying and combining existing code, and by tweaking parameters in existing programs to see what they do. Both of these are valuable ways of programming such systems, but they both require a large investment of time. In this paper we describe a system for exploring the sound-space of a synthesis algorithm, which aims to give users a middle way between the complexity of programming and the simplicity of using preset sounds.

This core of this paper is divided into three sections. In the first section we briefly describe the concept of interactive genetic algorithms upon which our system is built. In the second section we describe our system. In the final section we discuss interactive genetic algorithms more generally, and consider what we might learn from their application to systems such as this.

2 Interactive genetic algorithms.

The idea of using evolution with a human in the loop first occurs in the work of Dawkins [1, 2] and Todd and Latham [3]. They both created graphical systems which present the user with a number of pictures, generated by some process from a list of parameters. The user then gives a rating to each of the pictures, and the parameters used in creating those systems are then crossed over and mutated to present the user with a new set of pictures. By iterating this process through a number of generations, the user can create images which emphasize the characteristics of those images which have been selected during the run of the algorithm. This can be used both to search the space of possible pictures in an exploratory way, and to search that space for a particular image. These ideas have since been applied to music, computer-aided design [4] and searching large data sets for patterns [5]. A comprehensive survey is given in the introductory article in this special section of *Leonardo*.

3 An IGA for controlling sound synthesis.

In this section we give a brief outline of our system. More complete details of the implementation are given in [6]. The system is designed to act as a front end to the CSound synthesis system [7, 8]. In particular the current version of the software acts as an interface to the FOF algorithm [9], which is a granular synthesis technique. Granular synthesis is a technique which builds up a complex sound by constructing a "cloud" of tiny grains of simple sounds, e.g. very short sine-wave bursts. To create a sound in a granular synthesis system requires the specification of a large number of parameters, for example the shape of the overall cloud, the fundamental frequency, the way in which the individual grains are generated, the structure of the individual grains used, et cetera. Therefore understanding this sound-space is complicated, so we have chosen it as a case study for our methods.

The evolutionary interface to this works in a way similar to many other interactive genetic algorithms. Members of the population consist of a list of (binary encoded) parameters, which can be put into a synthesis algorithm to create a sound. The user is presented with an interface (figure 1) which contains a number of buttons, each representing a current member of the population. Upon pressing a button, a sound is produced, by processing the parameters from the corresponding member of the population. The user can listen to the sounds in any order, and they can rate the sounds by moving a slider. When they are happy with their ratings they press an "evolve" button, which carries out one round of the genetic algorithm on the population and resets the window.

FIGURE 1 AROUND HERE.

So far no formal user testing has been carried out using this system, but informal usage suggests that it is easy to adjust the sound at various levels of detail. Firstly it takes only a small number of generations to concentrate the population into one of the main regions of sound-space; the main interesting regions of the space are a short percussive sound and a sustained strident sound. Within these regions it is possible to steer the population towards a particular characteristic, for example speeding up or slowing down the speed of repetition in the percussive sound-region, or raising or lowering the pitch in the sustained region. More detailed user testing of this kind of system will be an important next direction for this project, and some ideas for issues that could be investigated are suggested below.

There are a number of ways in which this software will be developed in the future. At present the system allows the exploration of a small range of sounds, i.e. those which can be produced by plugging single, constant parameters into one instance of a synthesis algorithm. Most programs used on synthesizers are more complex than this. They often use multiple, layered sound sources, the sounds are filtered and processed in various ways, and the parameters vary through the duration of the sound. To produce a system which is capable of exploring this richer sound-space, we intend to extend the system by manipulating networks of sound generating, parameter generating, and sound filtering processes. Ways to manipulate complex graph-like structures such as these have been described in Koza as part of *genetic programming* [10], a heuristic for the evolution of computer programs. Similar ideas could be applied to other computer music systems; in particular it would be easy to fit the evolution of *Max* patches [11] into this framework.

Another application of these ideas which could be incorporated into a future version of our system draws on the ideas of Yuan and Horner [12]. Their work is concerned not with exploring a sound-space, but with finding the parameters in a synthesis algorithm which produce a sound which matches a real sound. It is difficult to find a set of synthesis parameters which produce a sound which is like a real-world sound; however this is important because storing a set of synthesis parameters for a sound offers a much more compressed way of storing that sound than storing a sound-file of that sound. Their program analyses the sound produced by the recording and the synthesizer, and uses similarity between the output from these analyses as a way of assigning a fitness to the synthesized results.

4 Interactive genetic algorithms in general.

One of the aims of the workshop on which this special section of the journal was based was to consider some general questions surrounding the use of genetic algorithms in visual art and music. Thus participants were asked to reflect on the work that they had described, and raise some general questions to provoke discussion and future study. In this section I consider some such questions, in particular thinking about interactive genetic algorithms and how they differ from the typical non-interactive GA.

One characteristic of IGAs is that the fitness given to a particular item need not be the same if it reappears in future generations. This effect has been studied by May [13], who developed a system which takes musical phrases from the user and learns, via an IGA, which kinds of modifications to those phrases make the most interesting musical results. One possible "modification" that the system can make is not to change the phrase at all. It was discovered that there was a decline in the fitness assigned to a phrase when it had occurred several times.

We can consider a couple of possible explanations for this. The first is that users become bored of hearing the same phrase, and give higher scores to more novel phrases. Another explanation is that the quality of the phrases is improving, and that the relative quality of the original phrase is therefore lower compared to the rest of the population. Other experiments in May's thesis [13] suggest that the average fitness of the population increases with time, however the interaction between these two phenomena is not explored. In general we can ask the question **do users of IGAs rate the solutions relative to the current population or relative to individuals from across the whole time-span of the run?**

If users to adjust their scoring mechanism as they go, then we may end up with an implicit form of *fitness scaling.* Fitness scaling [14, 15, 16] is a technique where the fitness function is changed so that differences between higher values of (real) fitness are scaled so that they are larger than differences between lower values of (real) fitness. This is because individuals with lower fitness will be largely lost from the population in the early rounds, whereas the majority of comparisons will be between individuals with comparatively high fitness. Therefore the scaling increases the power of these comparisons at the expense of reducing the subtlety of comparisons between the less-used low fitness individuals. When we are using a traditional GA, we have to explicitly include this scaling. However we can **conjecture that in an IGA this scaling happens in an implicit fashion**. As an example consider the system above. During the early rounds users will typically assign a reasonably high fitness value to any sound which is vaguely relevant to the kind of sound that they are searching for. However, once all of the sounds in the population are reasonable, the user will adjust their idea of fitness so that they assign high fitness only within the context of reasonable examples, e.g. if they want to raise the tessitura of the sound they will only give a high fitness to those sounds which contain higher-pitched components. Investigating the details of this would be an interesting future project.

Another interesting question concerns the structure of the population in an interactive genetic algorithm. One idea for extending the system above is as follows. We would like to be able to "put aside" certain sounds which have a certain characteristic, and then reintroduce those sounds into another population during a future run of the algorithm in order to "steer" the new population towards having that characteristic. As an example we might take samples of melancholy sounds that we come across, and then introduce those sounds into a population at some stage in order to add a melancholy feel to those sounds. It is an interesting question to consider whether this idea might work. **Can we put aside elements from an IGA population which**

have a common characteristic, and use them in future populations to steer that population towards that characteristic?

Clearly a simple form of this would work. Consider a list of parameters for a synthesis algorithm where one of the parameters is amplitude. We would be able to put aside the loudest sounds, and then reintroduce can introduce those into a population with the aim of making it louder. The loudness parameter would get integrated into the current population by crossover, and (assuming that the effects of "hitchhiking" by other parameters was not excessive) we could steer the current population towards a louder region of sound-space. We can imagine the technique working in a more subtle way (especially with uniform crossover) for combining characteristics which are contingent on more than one allele. However it is not clear whether such a technique would apply to a more complex situation, for a number of reasons. We can imagine that some characteristics are contingent on the same alleles which are giving the sounds in the current population some of their other characteristics, and so it might not be possible to find a sound which combines the two characteristics. This is perhaps unsurprising, we are unlikely to be able to make a population of bright sounds sound melancholy, for example. A more subtle potential complication comes from the structure of the sounds, in that the characteristics which need to be changed to make some sounds e.g. melancholy might be different from those needed to effect a similar change on other sounds. Nonetheless it would be interesting to see whether this idea can be usefully applied, and to investigate connections with other kinds of "design by analogy". The idea of design by analogy has been surveyed by Goel [17], and Miranda [18] has discussed these ideas in a more specifically musical context.

Another interesting question about the structure of GAs which impacts upon their use in artistic domains is **whether the structure of individuals used in GAs which are "exploratory" should be of a different kind to the structure of those which are used for optimization**. It has been pointed out by DeJong [19] and others [20] that the view that GAs are just optimizers is needlessly limited—GAs are a way of robustly exploring a space, and optimization is only one such application of such explorations.

An example of how GAs might need to change is given by our work above. We found that this GA worked a lot better with a high mutation rate, 0.1, when compared to mutation rates for traditional optimization GAs which have a rate of 0.001–0.0001. This high mutation rate made it easier to search the space effectively, and we are currently looking into experiments to measure effect this more accurately. This kind of question would seem to be important for the development of "exploratory" type GAs in general.

So far we have considered the idea that the user will provide feedback directly, by giving a specific rating to a particular individual. Another avenue of exploration would be to make this feedback implicit and contextual. **Can we create a more contextual way of assigning fitness in an artistic GA?** A number of systems have attempted to place the musical or artistic GA in context. An example is Al Biles's *GenJam* system (see paper elsewhere in this issue) which contextualizes the GA by having the human and computer alternate improvisations on a theme. Another approach is given in the paper by Rowland and Biocca (also in this issue) which places evolutionary sculpture in the context of a virtual "sculpture park".

If we are putting our systems in such contexts, can we make our fitness allocations equally contextual? One approach has been tried by Biles [21], who experimented with audience-mediated interaction, where the audience hold up red or green paddles to indicate their like or dislike of what the computer is playing. This is then fed into the computer via a human mentor ascertaining the proportions of red to green in the audience. We could imagine extending this idea further by taking this measure directly from audience applause, for example.

Taking this idea further we might explore ways in which to implicitly measure the fitness of an individual without asking the user for explicit feedback. One example has been discussed by May [13], who was interested in creating "catchy" tunes. A catchy tune is one which is easy to remember, which "catches" the mind (this idea leads onto Dawkins's idea of *memes* [22]). May has suggested using the hum-back of the tune as the feed-back into the system. The system would measure how accurate the hum-back of the tune was (using e.g. a pitch-to-MIDI converter) and then use this as a fitness measure. We can think of using similar ideas in a visual art context, e.g. tracking visual attention given to the various items in a virtual sculpture park *a la* Rowland and Biocca, and evolving new sculptures from those that most attention was payed to. A basis for future developments of this kind could be based around the recent work by Picard and others [23] on *affective computing*. This is computing (particularly human-computer interaction) which takes into account emotions, e.g. measuring the emotional state of the user and feeding that into a program. Making use of such feedback in an artistic domain would seem to have great potential.

References

[1] Richard Dawkins. The evolution of evolvability. In Christopher G. Langton, editor, *Artificial Life*, pages 201–220. Addison-Wesley, 1989.

[2] Richard Dawkins. *The Blind Watchmaker*. Penguin, 1990.

[3] Stephen Todd and William Latham. *Evolutionary Art and Computers*. Springer, 1992.

[4] Peter J. Bentley, editor. *Evolutionary design by computers*. Academic Press, 1999.

[5] G. Venturini, M. Slimane, F. Morin, and J.-P. Asselin de Beauville. On using interactive genetic algorithms for knowledge discovery in databases. In T. Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 696–703. Morgan Kaufmann, 1997.

[6] Colin G. Johnson. Exploring the sound-space of synthesis algorithms using interactive genetic algorithms. In Geraint A. Wiggins, editor, *Proceedings of the AISB Workshop on Artificial Intelligence and Musical Creativity, Edinburgh*, 1999.

[7] Richard Boulanger, editor. *The CSound Book*. MIT Press, 2000.

[8] Barry Vercoe. CSound manual. Manual distributed freely with CSound software, 1992.

[9] J.M. Clarke. An FOF synthesis tutorial. Appendix 4 of the CSound manual, 1992.

[10] John R. Koza. *Genetic Programming : On the Programming of Computers by means of Natural Selection*. Series in Complex Adaptive Systems. MIT Press, 1992.

[11] Todd Winkler. *Composing Interactive Music: Techniques and Ideas using Max*. MIT Press, 1998.

[12] Jennifer Yuen and Andrew Horner. Hybrid sampling-wavetable synthesis with genetic algorithms. *Journal of the Audio Engineering Society*, 45(5):316–330, 1997.

[13] Tony D. May. Music and computers: The design and implementation of a musical genetic algorithm. Master's thesis, University of Kent, 2000.

[14] Stephanie Forrest. Documentation for PRISONERS DILEMMA and NORMS programs that use the genetic algorithm. Unpublished software documentation, 1985.

[15] A.M. Gillies. *Machine Leaning Procedures for Generating Image Domain Feature Detectors*. PhD thesis, University of Michigan, Ann Arbor, 1985.

[16] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[17] Ashok K. Goel. Design, analogy and creativity. *IEEE Expert*, pages 62–70, May/June 1997.

[18] Eduardo Reck Miranda. An artificial intelligence approach to sound design. *Computer Music Journal*, 19(2):59–75, 1995.

[19] Kenneth De Jong. Genetic algorithms are NOT function optimizers. In L.D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 5–17. Morgan Kauffmann, 1993.

[20] Inman Harvey. Cognition is not computation: Evolution is not optimisation. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Proceedings of the Seventh International Conference on Artificial Neural Networks*, Lecture Notes in Computer Science 1327, pages 685–690. Springer-Verlag, 1997.

[21] John A. Biles. GenJam Populi: Training an IGA via audience-mediated performance. In *Proccedings of the 1995 International Computer Music Conference*, 1995.

[22] Richard Dawkins. *The Selfish Gene.* Oxford University Press, 1989. Second edition. Original edition 1976.

[23] Rosalind Picard. *Affective Computing.* MIT Press, 1997.

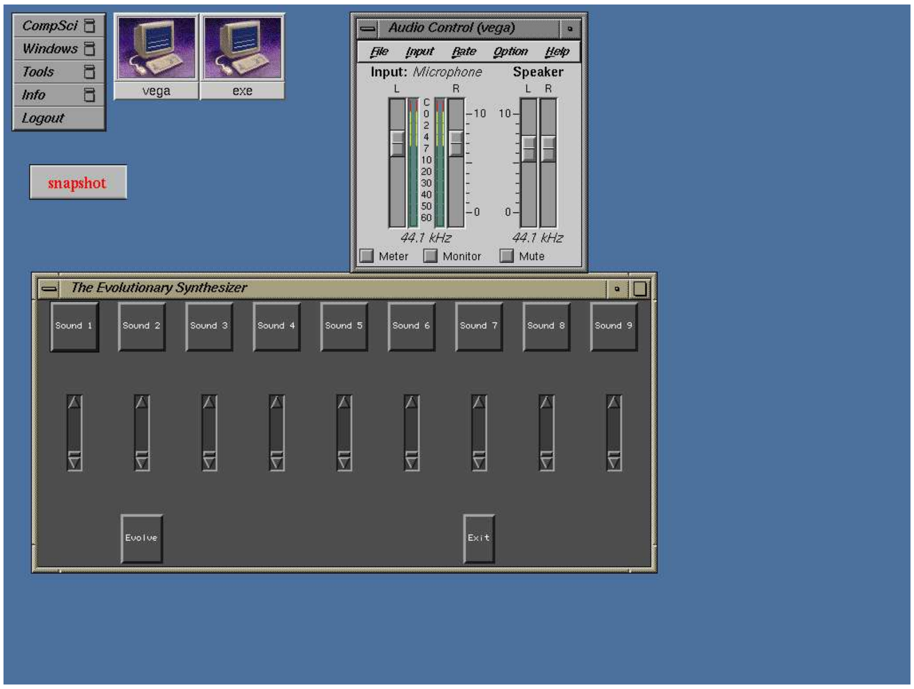
CompSci
Windows
Tools
Info
Logout
vega
exe
snapshot
Audio Control (vega)
File
Input
Rate
Option
Help
Input: Microphone
Speaker
44.1 kHz
44.1 kHz
Meter
Monitor
Mute
The Evolutionary Synthesizer
Sound 1
Sound 2
Sound 3
Sound 4
Sound 5
Sound 6
Sound 7
Sound 8
Sound 9
Evolve
Exit

Figure 1

Figure captions.

Figure 1. The interface to the evolutionary synthesizer.