Interactive Evolution of Breakbeat Cut Sequences

Nick Collins
nick@sicklincoln.org

Abstract

This research project explores the potential of interactive evolution in the domain of automatic breakbeat cutting. Specific breakbeat cut sequences are discovered through the evolution of arguments for algorithmic audio cut generators. The implementation of this work is for the Mac based real-time audio programming language SuperCollider, and is made possible by combining extension class libraries by the author supporting interactive genetic algorithms and algorithmic cutting of audio. A human auditor guides a genetic search through argument parameter spaces for cut procedures. Faced with the sheer size of the parameter spaces involved, interactive evolution offers an attractive method of finding successful argument sets to algorithmic composition routines.

Key Words: interactive genetic algorithms, audio cutting, algorithmic composition

Introduction

Where in prior art virtually all composers of dance music have worked extensively by hand in dedicated sequencer packages, sometimes in laborious detail, the methods of breakbeat cutting, that is, resplicing segments of audio in drum loops, have recently been successfully automated (Collins 2001a, Collins 2001b). The automation itself raises problems due to the large input parameter spaces of the algorithms involved. The question arises, is there some way to search for good arguments that promote effective aural results? One method with potential in solving such a remit is interactive evolution.

The space of possible cut sequences, and the subset producible by a given audio cutting algorithm is a massive mathematical territory (Collins 2001b), but can be explored via genetic algorithm in collaboration with a human auditor. This article relates a number of experiments based on the evolution of algorithmic composition routine parameters. (Dahlstedt 2001) showed great potential in the combined optimisation of synthesis and algorithmic composition

parameters (low and high level perceptual entities) and we build upon this foundation. Further academic work of note on algorithmic composition within modern musical styles utilising genetic algorithms research includes (Truong 2002) and (Pearce 2000).

The reader who is not au fait with SuperCollider, or even audio, should still be able to appreciate the application of interactive genetic algorithms to evolving arguments for algorithmic composition routines. One can envisage similar processes, say, for automated video cutting.

This research project combines two previous lines of investigation based on SuperCollider 2 (McCartney 1998) class libraries for breakbeat cutting (Collins 2001b, 2002a) and interactive genetic algorithms (Collins 2002b). The BBCut Library provides support for experiments in audio cutting, facilitating the writing of new cut procedures without rewriting synthesis code. The GAParams Library has base classes for genetic algorithms and user interface tools for interactive evolution.

The parameter space of the original automatic breakbeat cutting algorithm, BBCutProc11, is explored. A new cut procedure, MotifCutProc, which manipulates an intermediate level of hierarchy between block and phrase, is introduced. Motifs for the routine are evolved using the genetic algorithm. The simultaneous evolution of multiple layers of breakbeats is shown to be a powerful method of arriving at polished resultants, as long as a few restrictions are in place to enforce a degree of synchrony between voices. The results are promising and open up both a world of possibilities for developing effective custom audio cutting solutions, and a host of new challenges.

An Initial Critique: Problems of Assessment

The remit is to evolve algorithms, not fixed and rigid solutions. When the evolved arguments for the breakbeat cutting algorithm are applied, the run of the algorithm can still vary from time to time due to its random seeded decision making. We are restricted to judging based on the 'kind of output' that the algorithm favours - the human observer bottleneck in the genetic algorithm is bad enough without this additional requirement for multiple auditioning! A future solution might be in statistical analysis of the cut sequences generated by the algorithm for the given arguments, over many runs, though that is a step towards automating the whole procedure by genetic algorithm and some non interactive fitness function. The Truong thesis

already mentioned allows statistical results to be available to the human decision maker at each generation choice.

A further flaw is that specific example breakbeats are utilised in the evolution. It is entirely possible that a routine that sounds bad with one break may sound better with another! So we may have to audition the same routine many times for many breaks.

There is some redemption: the great potential of this work may be exhibited to be in evolving cut routines for *specific* situations, not generic good solutions, so with that in mind, the problem is reduced to showing the facility of the method to sound exploration. In fact, in the cases outlined below, we evolve synthesis values for free simultaneous with cutter arguments, accepting the context sensitivity of the judgements.

Interactive Evolution of Parameters for the BBCutProc11 Algorithm

The original breakbeat cutting algorithm (Collins 2001a) was intended to simulate the style of early jungle/drum and bass breakbeat splicing. BBCutProc11 is the current version of this algorithm in the BBCut Library, and its input parameter space for these experiments was mapped out as follows:

Input Parameter	Range of Permissible Variation	Notes
subdivision	6, 8, 16, 32	weighted indexing using [0.1, 0.5, 0.35, 0.05]
bar length	1, 2, 4	weighted indexing using [0.05, 0.15, 0.8]
phrase length	1 – 8	round 0.5
repeats	1 – 5	round 1.0 (integer)
stutter chance	1.0 - 2.0	exponential, then mapped to 0.0 to 1.0
stutter speed	1 – 5	round 1.0 (integer)
stutter area	0.1 - 1.0	exponential
random offset	1.0 - 2.0	exp, map to 0.0 - 1.0
amplitude function	0 – 2	round 1.0
pan function	0 – 3	round 1.0
pan function parameter 1	0.0 - 1.0	
pan function parameter 2	0.0 - 1.0	

Table 1 Parameter set form for evolution of arguments to the BBCutProc11

The first seven are the arguments of the cutting algorithm, the last five are independent parameters for the synthesis of cuts evolved for free at the same time. The table indicates the possible values for the input parameter space, and notes wherever the form of the mapping is anything other than linear. .

The GAParams Library only supports continuous float values for genetic algorithm variation, but there is facility for automatic rounding, and it is also helpful to use weighted indexing amongst a small set of states (the evolved parameter is in the range 0.0 to 1.0), as noted in the table above. Variation in the parameter range may be linear (additive) or exponential (multiplicative).

A very rough calculation of the magnitude of the sensation space (assuming that all parameters are independent excepting the pan function parameters and that continuous params have about ten distinct audible states) gives about 500,000 states for the algorithm and 600 million states including synthesis parameters. Because human observers may listen with varying coarseness of judgement, the sensation space is quickly limited in practise to rougher categorisation.

Working with the interactive evolution system and the BBCutProc11 routine was a pleasure, with many new facets of the routine's capabilities made accessible.

Interactive Evolution of Motifs

The MotifCutProc is a new cut procedure for the BBCut Library which handles an intermediate level of hierarchy between the standard block and phrase. Motifs are short phrases, small chunks of cut sequence, composed of a number of blocks. They are concatenated in an order given by an indexing function to form the larger phrases. The phrase length may mean that the last motif of the phrase must be cut short. The MotifCutProc holds an arbitrary number of Motifs, the indexing function is a user parameter, and UI facilities have been built to allow the specification of Motifs during run time.

In evolving arguments for the MotifCutProc, the motifs themselves are evolved and the size of the parameter space is necessarily extended to cover the maximum number of motifs that could possibly be required. The table of parameters for this evolution for a single cutter was as follows:

Parameter	Range	Notes	Number of parameters of this type
cut size	0.125, 0.25, 0.5, 0.75, 1.0, 1.5, 2.0	windex using [0.1, 0.15, 0.35, 0.1, 0.15, 0.1, 0.05]	50
repeats	1, 2, 3, 4, 5	windex using [0.4, 0.2, 0.15, 0.15, 0.1]	50
motif size (number of blocks)	1 – 5	round 1.0 (integer)	10
number of motifs	1 – 10	round 1.0	1
phrase length	1 - 8	round 0.5	1
offset chance	1.0 - 2.0	exp, map to 0.0 - 1.0	1
amplitude function	0 – 2	round 1.0	1
amplitude	1.0, 2.0	exp, map to 0.0 - 1.0	1
pan function	0, 3	round 1.0	1
pan parameter 1	0.0 - 1.0		1
pan parameter 2	0.0 - 1.0		1

Table 2 Parameter set form for evolution of arguments to the MotifCutProc

The first five parameters are for the cut procedure, the last six are again for cut synthesis.

In order to allow a variable number of motifs (up to 10), and blocks per motif (up to 5), all the parameters that could possibly be needed for the maximal case are always evolved, and we use only those that the number of motifs parameter indicates. The values 10 and 5 are easily changed in the code to try out other possibilities. In creating motifs, each motif is the concatenation of a number of blocks, a block being the cutsize reiterated repeats times.

The indexing function for the MotifCutProc was left as the default, random selection from the bank of available Motifs.

The evolution of specific cut sequences as within the remit of the Motif concept can be further explored by limiting the number of Motifs to one, and forcing phrase length and motif size to be large.

Again, a great sense of access to the far corners of the potential of this cut procedure was engendered.

Simultaneous Evolution of Multiple Layers of Cutters

All the genetic algorithm work on BBCutProc11 and the MotifCutProc made automatic provision for simultaneously evolving N cutters. For such an application there were a few further restrictions on the group to keep them in some sort of sync with one another. Such limitations could be dropped for general audio cutting textures, but for breakbeat cutting the excitement is usually most intensified for quantised timing with virtuosic cutting and synchronous ensemble. With the evolution of specific groups of cutters working as a composite, the real power of this paper's techniques were certainly heard.

Conclusions

Results of this study have been very encouraging, and show the power of interactive evolution as a practical method of gaining a handle on the large parameter spaces involved in algorithmic composition. The future work is open ended, for similar application could be extended to any other cut procedure or combination of procedures in the BBCut Library. The composer's imagination is the limit. Further, aside from automatic audio cutting, interactive genetic algorithms are expected to be a wonderful composer's tool in general algorithmic composition work, and indeed, anywhere that a complex parameter space rears its ugly head. The proviso from earlier on , that specific rather than general solutions are to be evolved, is the likely set up.

So many freedoms are available to the designer of the parameter space for interactive evolution that it can be very hard to see how combined changes are effecting the output. The GAParams Library has facility to keep certain parameters fixed while evolving others, to help in isolating behaviour. There are always difficulties in assessing how effectively you are controlling the evolution. We must pay tribute though to the challenges raised by the interactive evolution of arguments to algorithmic composition routines. Evolving algorithms that themselves can change in output each time they are run, it is hard to assess fine differences. It may be helpful to build a statistical analyser to assess many different runs of the algorithm with the evolved arguments and report on trends. Further, this work has shown that the GAPArams Library may be improved with more user control over the scaling of changes at different generations of the evolution.

References :-

Collins, N. (2001a) Algorithmic Composition Methods for Breakbeat Science. *Proceedings of Music Without Walls*, *De Montfort University, June 21-23, 2001.*

Collins, N. (2001b) Further Automatic Breakbeat Cutting Methods, *Proceedings of Generative Art*, *Milan Politecnico, Dec 12-14, 2001.*

Collins, N. (2002a) The BBCut Library. *Proceedings of the International Computer Music Conference*, *Gothenburg, Sweden, 2002.*

Collins, N. (2002b) Experiments With a New Customisable Interactive Evolution Framework. *forthcoming in Organised Sound Vol 7 No 3, December 2002.*

Dahlstedt, P. (2001) Creating and Exploring Huge Parameter Spaces: Interactive Evolution as a Tool for Sound Generation, *Proceedings of the International Computer Music Conference*, *Habana, Cuba, 2001.*

McCartney, J. (1998) Continued Evolution of the SuperCollider Real Time Synthesis Environment. *Proceedings of the International Computer Music Conference*, *Ann Arbor, Michigan, 1998.*

Pearce, M. (2000) *Generating Rhythmic Patterns: a Combined Neural and Evolutionary Approach*. Master's thesis, University of Edinburgh, UK. Pearce's thesis and online audio examples can be accessed at http://www.soi.city.ac.uk/~ek735/msc/

Truong, Brian. (2002) *Trancendence: An artificial life approach to the synthesis of music*. Master's thesis. School of Cognitive and Computing Sciences, University of Sussex, UK. Available from www.cogs.susx.ac.uk/lab/adapt/MSc2002/bt20.pdf

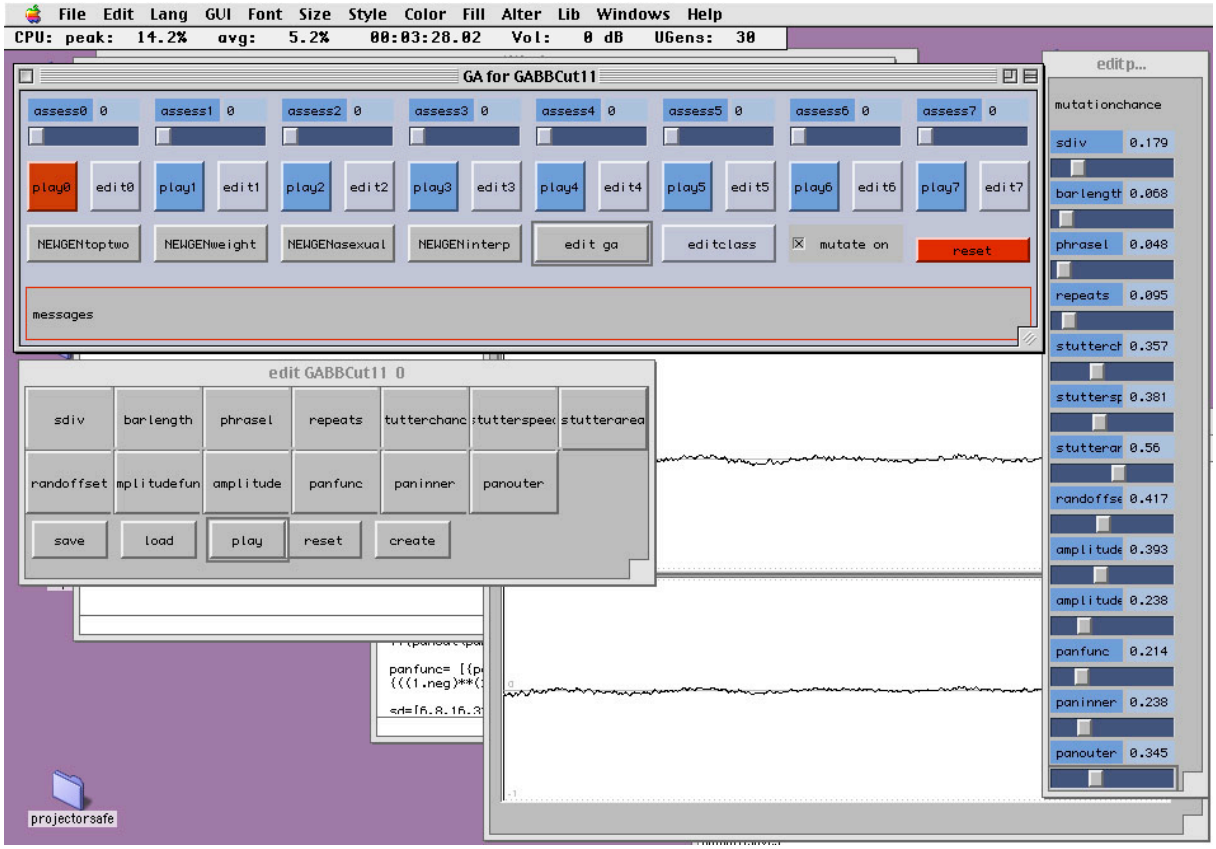

Figure 1 Screenshot of the GAParams Library, evolving BBCutProc11 arguments