Further Automatic Breakbeat Cutting Methods

N. M. Collins
Centre for Electronic Arts, Middlesex University, London, UK.
n.collins@mdx.ac.uk
http://www.axp.mdx.ac.uk/~nicholas15/

Abstract

Following the invention of an automatic breakbeat cutting algorithm in the style of early 90s jungle, further experiments are described. These are namely, the use of more advanced techniques to control choices in the original algorithm, and new algorithms for cutting including methods based on campanology and recursion.

Automatic breakbeat cutting can reduce the effort of working by hand in a sequencer with MIDI triggering of a sampler. Furthermore, the parameterisation of the process concedes new techniques that can be awkward to implement with a sequencer, for example, cutting in septuplet demisemiquavers.

To improve the original algorithm, states, whether offsets, cut sizes or repetition counts, can be governed by Charles Ames' method of statistical feedback. Weight distributions can be changed during a phrase to give more control of cut sequence structure. These processes are investigated in the light of the output pacing and variation of cut sequences.

Campanology, or change ringing, is based on a small subset of a permutation group consisting of permutations that can only swap adjacent elements in distinct pairs. When acting upon offsets into the source a fluid series of undulating cuts can be produced.

Recursive cutting is an analytical test of second order cutting. It takes a base sequence of [cut length, offset] pairs and further cuts them up, producing variations on a given cut sequence.

The Warp Cutter is inspired by the thought of constant stutters and rolls. Probabilities control the likelihood of simple blocks, even rolls or geometric accelerating rolls, usually at very fast repetition rates.

All methods have been implemented as SuperCollider patches and classes, and publicly released to accompany this paper in the BBCut Library.

1. Introduction

This paper is like a cookbook of new recipes for automated cutting of target audio. Historically these approaches derive from the author's desire to automate the style of early 90s breakbeat manipulation in what began as hardcore/jungle and became drum and bass. An algorithm for automated breakbeat cutting in that syncopated manner was presented in [5] and is publicly available as SuperCollider code from the author's web site, and as a Csound ugen (Csound 4.14 on). The explicit design of an algorithm has inspired new compositional directions.

Other attempts to apply algorithmic composition techniques to popular music are relatively rare in the literature (but see [1]). Pearce and Wiggins [6] as the practical test of their framework generate drum and bass patterns. Examples are available at [7]. Their perception of machine learning is that humans should not specify rules, but that all rules must be discovered from training examples. This author concedes that the work of this paper proceeds from an opposite viewpoint, that of the 'active style synthesiser' rather than the 'empirical style modeller' to quote their terms. The search for new generative methods extending craft is the domain. New compositions are the ultimate goal, not style synthesis. It is worth noting that dance music is such a fast evolving and broad genre that the ability to reproduce last year's techniques without finding novel viewpoints is of limited practical use.

Experiments with phrase structure of cut sequence generations reference the work of Charles Ames [2,3,4]. This work explores a finer control of probabilistic decisions, with time position dependent weights. The aim is to promote or restrict diversity in automated generations.

Other novel types of cut procedure are then presented, including processes based on campanology and recursion.

Whilst this paper is slanted towards breakbeat science, the algorithms are entirely pertinent to any source audio. For instance, campanology cutting could be a good method of playing background noise segments for long stretches of time without a direct loop becoming obvious.

2. The Phrase, Block, Repeat paradigm; the BBCut Library

The BBCut Library is a collection of automated cutting procedures which will work on any type of source audio stream. It is a set of SuperCollider 2 [10] classes facilitating the separation of synthesis of cuts from the cut procedure. This is not considered in detail here, and we restrict ourselves to the cutting algorithms themselves. The paradigm for general cut procedures used in the library is quickly described since it is assumed throughout the algorithm descriptions.

The code and help files for the library are publicly available from the author's web site and would assist in studying any of the cut method algorithms given later. The names of the procedures given are those in the library. Description for algorithms is informal since the real code is accessible.

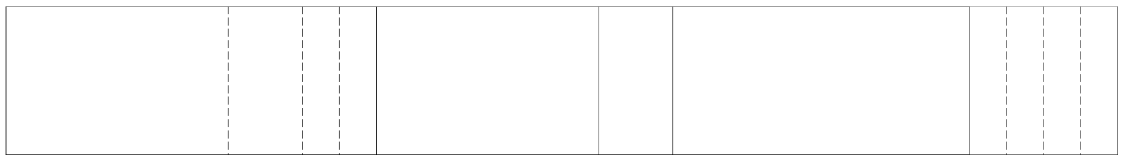

Figure 1 The Cut Hierarchy

Figure 1 shows the hierarchical levels in a general cut procedure. The phrase is a rounded sequence of cuts, perhaps an integer multiple of 4/4 measures. The block uses audio starting from a particular point in the audio source. This might be a given offset into a soundfile or other existing signal buffer (the notion of offset is restricted for a stream where no 'future' audio is accessible). The block is constructed of any number of atomic repeats at the common offset whether of equal size or not. A really fast set of many repeats will often be referred to as a roll or stutter. When the repeat rate is fast enough, what is effectively wavetable playback is occurring which can become of audible pitch rate (greater than 20 Hz).

In the diagram, there is a single phrase composed of five blocks. The first and last are subdivided into more than one atomic cut, or repeat. Those for the final block of the phrase are evenly spaced, a roll or stutter. Those for the first are of irregular length - but each of the four would use audio beginning from the same offset. Note that offsets and repeats are essential if the output is to be anything other than the original source enveloped on blocks. If the audio source to be cut does not have a restricted scope of random access offset data, repeats are still permissible ways of cutting up since they work on past audio. Any past signal component in a stream could also be made accessible to offset cutting as long as it is stored.

The diagram shows a cut sequence without overlaps. In principle, cut sequences could specify [delta time, duration] pairs for the two aspects of atomic cuts (being time to the next cut, and duration of this cut). Overlap is propitious for microlevel cuts (granularisation), though the synthesis engine for the BBCut library has 'duty cycle' parameters to allow such enveloping. It is hard to think up useful examples of duration versus delta time that do not involve a constant or proportion already implementable with a specific BBCutSynth. Delta time equal to duration is assumed for the rest of this paper as this is adopted for all cut procedures discussed.

3. The space of possible cut sequences

Let us assume the cut sequences work on a quantised grid, with a measure evenly divided into 16 possible cut positions. Phrases will be four bars long. Further, let there be eight possible offset points into the source. Then the number of possible non-overlapping cut sequences is given by:

$$(2*\text{offsetstates})^{(\text{cutsperbar}*\text{numbars})} = 16^{64} = 2^{256}$$

and this is a simple example! Spaces of all possible cut sequences will be of intractable size to exhaustive search. Counting partitions is not enough since 3+3+2 is taken as rhythmically distinct from 2+3+3.

Actual cut procedures won't get much more restricted in their productions by imposing a set of states of possible cut sizes. If those states involve the minimal cut size of one unit, unless something stops the aberrant chance of continually choosing that cut length, the procedure would be able to cover the whole space as above. Note that statistical feedback as introduced below can become the mechanism of such curtailment.

Analysing the long term diversity of an algorithm's productions involves assessing the variations of a cut procedure's decisions over many phrases. The algorithm has the potential to cover such a huge space of possible output phrases that it may never repeat itself within a human lifetime of listening. However, this is not to say that the kind of phrase produced does not have certain characteristics. In the following we study attempts to vary and control those characteristics.

For an example of these spaces, an eight bell composition in campanology can take hours to perform before the permutation chain gets back to the beginning, and is potentially maximally diverse in range if it covers the whole set of permutations of the bells. Even then, a given permutation chain might traverse the whole space many times in different orders before the chain restarts in sync with the set of states being acted upon.

In general, though permutation chains are deterministic, the use of probabilistic models in making decisions means that it is a probability space that is examined to give predictions of likelihoods of future events.

Cut sequences might be restricted by holding some memory store of previous sequences that forms the source for future productions. It is likely that subtle variation would still be required.

4. Cut Procedures

I name the procedures as they appear in the BBCut library.

4.1 ChooseCutProc

The investigation begins with a pertinent simplified version of the early algorithm ([5] - current version in BBCut library named BBCutProc11). Whilst BBCutProc11 automatically chooses the sizes of possible cuts based on the subdivision parameter (with odd numbers and syncopation as the goal), this cut procedure simply allows one to specify a set of allowed cutsizes, and a set of permissible repeats. Each block is created as cutsize*numrepeats. A phrase may constrict the number of repeats or even the cutsize for the final block so as to meet the phrase length stipulation. In terms of possible rolls, this algorithm is very similar to BBCutProc11, but for the purposes of the following, assume that the chance of rolls has been set to zero.

4.2 StatBalProc

The simplification of the ChooseCutProc makes explicit the number of possible states for cutsizes and repeats. The simple way to choose from the set of states is the Lehmer random number generator [3]. There are other ways of selecting states. In this procedure, Charles Ames' method of statistical balance/feedback [2] keeps track of selections in comparison to the bare use of the Lehmer process.

The heterogeneity parameter of statistical feedback gives some control over characteristics of dispersion and probable speed of realisation of distributions [4]. With low heterogeneity, dispersion approaches the deterministic, and rare states cannot occur close together, unlike the Lehmer process' effectively independent trials. Low heterogeneity means faster realisations (within a threshold of confidence). This makes statistical feedback very effective for fast changing weights (see section 4.3 below).

In practical application, the size of phrases has a direct interference on the effect. There is a dependence of how many decisions are taken in a phrase on what those decisions are since larger cutsizes fill the phrase more quickly. Further, at the ends of phrases certain decisions are changed, say if there is not room to fit a new block into the phrase length.

Even on sets of offsets, the use of statistical feedback is not especially obvious. Its effect is easily seen examining the statistics of output cut sequences, but the audible impression is limited. Dispersion character can be perverted when repeats and cutsize work independently. A rare cut choice of 0.3 beats might team up with a 4 repeats state to give a block [0.3, 0.3, 0.3, 0.3] which hardly promotes the rarity of the 0.3 size cut! This problem can be reduced by imposing temporary constraints; repeats of cut sizes force multiple selection of that cut size. Yet, gross statistical imbalance may result. The nature of breakbeats can also work against us. Sources may already include repeats, perhaps a similar motif of kick to snare. Effects are more pronounced with a more disparate source.

4.3 StatBalProc2

This procedure experiments with changing weightings for state decisions within phrases. This is to achieve the aim expressed in [5] of giving more control over phrase structure. We might desire large cuts, high numbers of repeats and early offsets towards the front of phrases and the converse towards the end. StatBalProc2 uses statistical feedback, theoretically quicker for low heterogeneity at realising distributions. With changing weightings that may go to zero, we must use the normalisation step of the feedback process, so that zero weighted states' statistics do not fall unrealistically behind whilst they are unavailable. The statistical feedback process is perfectly suited to tracking changes to weightings of states without any imposition of probability frames.

It is quite hard to hear the benefits of StatBalProc, but StatBalProc2 demonstrates far more shaping of phrases. Whilst again apparent from statistics of output, the use of statistical feedback remains only weakly appreciable compared to the same process of changing weightings realised with the Lehmer generator. A Cutter might form one stream in a multilayered audio, we may never listen that closely to it. Lehmer was already enough to convince the ear of variation. Statistical feedback may be suitable for some specific compositional concerns over realisation and dispersion of distribution, particularly across long phrases. When phrases are too short, neither sequence generator has time to realise weighted distributions, though the dispersion of rare states across many phrases will be restricted for Ames. The principle of changing weightings over phrases is good for phrase structure control.

As a future improvement, it might be wise to investigate the balanced bit generator as driver. This is the continuous rather than discrete version of statistical feedback.

4.4 CampCutProc - Campanology Cutting

Campanology is the craft of bell ringing, change ringing is the name given to the special permutation system where only adjacent bells may swap position in the peel. See [8] for a good web resource on the theory, with a practical tool to try out compositions.

The CampCutProc will work with any appropriately presented change ringing composition, that is, permutation recipe. The number of bells involved is critical; the mnemonic is one block per bell and one bell per offset state. With eight bells and a phrase length of 8.0 beats, then a phrase consists of eight (undivided) blocks each of length 1.0. The procedure maintains the current permutation status according to the change ringing pattern, updating to the next state with each new phrase. Each block can then be given the offset into the source corresponding to the current bell positions

Figure 2 - first few steps of a change ringing composition

```
Gainsborough Little Bob Major
x.18.x.18.x.16.x.18.x.18.x.12

abcdefgh  under x = (12)(34)(56)(78)

badcfehg under 18 = (23)(45)(67)

bdafcheg under x …
```

Because offsets are being handled, the source cannot be shuffled and reordered by the cut procedure if it is, for example, the current audio in stream. Therefore only sources that respond to offset position data will be audibly effected (apart from block enveloping) by the cutting.

When using the procedure it is relatively hard to distinguish different change ringing compositions, say a Gainsborough Little Bob Major from an Ashtead Surprise Major, unless one concentrates very intensely. The same might be said of hearing real bell ringing, though the different pitches of bells aid their aural discrimination. The effect of differentiation is obviously improved if all the parts of the source that will be presented are well known and discernible. For breakbeats, where a similar kick sound might be presented later in the sample, the position states are not so discrete. Because permutations may not take account say of the difference between different positions in a measure (strong, weak pulses) output breakbeat patterns can be very distorted in their feel. However, the sequence of permutations as applied to breakbeats gives a good sense of continuous variation, covering a far share of the permutation space, without sounding like total randomness; there is method here. This author has had good success particularly when the CampCutProc is used as a background process to a second voice with a more rhythmically sharp cut procedure.

Phrase length does not have to correspond to the source length. If block sizes play more than the bell's share of the source, then linked bell patterns emerge, where A cannot be played without part of B or beyond following.

If the source is a sample of the bells evenly spaced, the campanology cutter just does change ringing. Use a set of samples and choose an index based on the offset if the overlaps are to be successful.

4.5 RecCutProc - Recursive Cutting

To do recursive cutting in a straight forward manner, record the output of a cut process and pass it as the source for a second round of cutting, to whatever desired level of recursion. The trend is quickly towards increasingly restricted amounts of the original source in ever smaller repeat patterns, since a cut procedure can only return as much audio as the original source at best, and usually involves the discarding of some information. Cutups continue to subdivide already subdivided parts, so that the cut level heads to the smallest quantisation level possible within the cutsizes.

As an analytical demonstration of this, a first order recursion cut process was explicitly worked out.

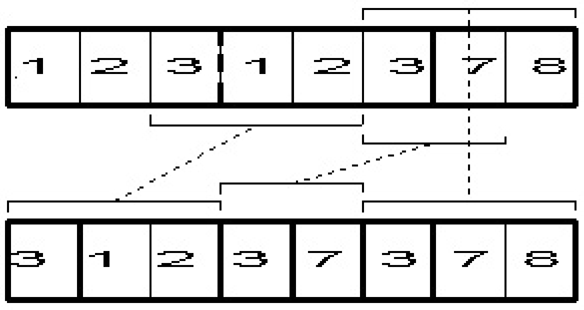

Figure 3 a recursive cut

Figure 3 shows an example of the procedure in action. A source 3+3R+2 pattern is the higher in the diagram. There are two blocks, the first with two repetitions, the second a single repeat. The offset positions into a source are listed. The lower pattern is produced from the higher by taking cuts of size 3, 2 and 3 to form a new phrase. These cuts can be from any point in the source cut sequence. The diagram shows the derivation of these cuts and the blocks created in the output sequence.

The benefit may be in controlled variations of existing cut sequence motives. The recursive cutter cannot explore any new region of the source than its own source cut sequence, but may distort and vary existing rhythmic figuration.

Non-analytical examination of nth order recursive cutting supports this result. The tendency is very quickly to stutter on small parts of the source. Informally, it seems appropriate to begin at the top level with a long phrase and source with large cuts. Successive recursive cutting quickly breaks this up, the details being dependent on the type of cut procedures applied.

4.6 WarpCutProc1

Warp cutting is named after the infamous Warp Records, and might be found in the work of Aphex Twin, SquarePusher, μ-ziq and others. Their use of rolls often uses the technique of very small repeated cuts even up to audible pitches of frequency. For the algorithm inspired by this, different types of rolls can happen throughout a phrase (as opposed to only at the end for the original BBCutProc11). The algorithm works a block at a time, filling the current phrase as it goes, subdividing any given block into some set of varying sized but same offset cuts. This is in contrast to BBCutProc11 or the ChooseCutProc derived procedures which start from the basic cut size and try to fit in a given number of repeats. There are three possible ways that the procedure may treat a block (selected by user provided probabilities):

1. Do not subdivide the block.
2. Subdivide the block evenly.
3. Subdivide the block using a geometric progression on the cutsize, with a user provided probability of choosing fast to slow or vice versa.

The user provides the blocksizes and number of subdivisions to allow. More properly, the user can specify functions that return the necessary values, a trick allowed by the SuperCollider language - these complexities are not discussed in detail in this paper.

A version of the WarpCutProc respecting phrase, like the StatBalProc2 above, is an obvious next step. The probabilities can change during the phrase. Normalised statistical balance could be used to keep track of shifting weights and react to smoothly varying distributions quickly.

5. Further work and conclusions

Hopefully this paper has given some idea of the scope of automated breakbeat cutting! A number of further automatic breakbeat cutting methods are presented, all available within the BBCut Library.

Some future paths seems more promising than others. The properties of statistical feedback as compared to Lehmer random number generation are not of great consequence in audible terms in the context of breakbeats. The use of changing weightings within phrases is audible and gives structure. An area for future investigation is the use of a memory store of fragments of cut sequences, particular motifs that can be reused. An algorithm might only permit a certain set of motifs at any one time, combining them to form phrases. This methodology would give a better controllable restriction on diversity, and would add an additional level of hierarchy between block and phrase - a motif. Versions of all procedures with changing weightings, or alternative sequence generations are plausible. For long phrases, the use of statistical feedback may yet be justified.

This author would not dismiss the ultimate power of working out cut sequences manually (compare the superbly crafted Mad Cat track by Roni Size [9]). Yet, automation does not restrict creativity but rather inspires new compositional directions.

Acknowledgements

Thanks to Rob Godman for challenging me to generate offset sequences from change ringing compositions. Thanks to John Eacott for making me think about algorithmic dance music more than I might have otherwise thought possible.

References

[1] Ames, Charles. and Domino, Michael. (1992). "Cybernetic Composer: an Overview". *Understanding Music with AI: Perspectives on Music Cognition*. ed M.Balaban, K.Ebcioglu, O.Laske. The AAAI Press/ MIT Press. pp. 186-205.

[2] Ames, Charles. (1990). "Statistics and Compositional Balance". *Perspectives of New Music* 28:1.

[3] Ames, Charles. (1992). "A Catalogue of Sequence Generators". *Leonardo Music Journal* 2:1. pp. 55-72.

[4] Ames, Charles. (1995). "Thresholds of Confidence: An analysis of statistical methods of composition; Part I -- Theory". *Leonardo Music Journal* 5:1.

[5] Collins, Nick. (2001). "Algorithmic Composition Methods for Breakbeat Science". *Proceedings of Music Without Walls*, De Montfort University, June 21-23, 2001, ISBN 1857213319. CD-ROM Proceedings.

[6] Pearce, M. T. and Wiggins, G. A. (2001). "Towards a Framework for the Evaluation of Machine Compositions". *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, Brighton, UK: SSAISB. pp. 22-32.

[7] training example d&b patterns - http://www.soi.city.ac.uk/~ek735/msc/ (accessed Nov 2001)

[8] van den Doel, Kees. http://www.cs.ubc.ca/spider/kvdoel/bells/bells.html (accessed Nov 2001)

[9] Roni Size and Reprazent. (1997). "New Forms". Audio CD. Mercury Records.

[10] McCartney, James. SuperCollider program available from http://www.audiosynth.com.