USER-GUIDED EVOLUTION OF GRANULAR SYNTHESIS

Corrado Coia, Brian J. Ross

Brock University
Dept. of Computer Science, St. Catharines, ON, Canada
corrado.coia@gmail.com, bross@brocku.ca

ABSTRACT

An innovative user interface for an audio plugin is presented. The system features an interactive genetic algorithm, for user exploration of granular synthesis parameter settings. The interface design is motivated to allow the user to intuitively and efficiently audition and evaluate parameter sets. This is particularly important with granular delay effects, since many parameter combinations will be undesirable to most listeners. The interface uses a minimalist 3-value evaluation scheme, which lets the user protect, use, or delete candidate parameter combinations. This permits parameters to be auditioned in a matter of a few seconds or less. The interface also lets the user directly edit parameters, for further use in evolutionary exploration, or for final music production. This style of evolutionary interface should be easily adapted to other musical applications in the future, for example, generalized synthesis engines.

1. INTRODUCTION

Genetic algorithms have a proven track record for finding acceptable solutions to complex problems [7]. The application of genetic algorithms in music composition is also well established [5]. They have been used towards creating musical scores, as well as processing or synthesizing audio signals. For example, evolution can be used to produce a synthesizer or sound source that generates output matching characteristics of an example sound sample [8] [11] [12]. These techniques use automatic fitness evaluation, to determine the closeness of match between candidate and target sound data.

An alternative approach is to use interactive genetic algorithms (IGA), in which the user interacts with the genetic algorithm to explore variations of sounds. IGAs are useful when there is no known computable evaluation function, for example, when subjective musical taste is of concern, and the criteria are too complex to formalize [2]. In such cases, the user auditions a candidate solution, and assigns it a score manually, according to its suitability. Johnson uses a mutation-based IGA to explore different granular synthesis settings [9]. Mandelis describes an IGA for the exploration of synthesis parameters [10]. The system makes use of various crossover operators, and a real-time interface controlled by a data glove. Dahlstedt uses an IGA to explore synthesis parameters [6].

Unfortunately, user evaluation of populations is the bottleneck in interactive evolution, due to user fatigue when having to manually evaluate hundreds of possibly unimpressive individuals. The problem is compounded with music and audio applications, because the user must invest time to hear an audio signal. This contrasts to visual art applications, in which the time invested is the significantly shorter duration it takes to glance at and evaluate an image.

EGDE (Evolutionary Granular Delay Environment) is a granular delay system[1]. Granular delay is a type of granular synthesis technique [15]. Sampled grains of an input signal are fed back into the output stream with some temporal delay. As is typical with granular synthesis, many parameters can be used to control the process, for example, duration of grains, frequency of creation, delay time, play-back pitch, forward or reverse direction, and synchronization intervals. These parameters interact in complex, often unexpected ways. This makes granular effects somewhat challenging to set up, especially when many parameter combinations results in extreme cacophony. This suggests that an IGA is worth considering.

EGDE's uses interactive evolution to let a user explore different parameter sets, in order to discover new and interesting effects from the wide variety possible. To help address the problem of user-exhaustion when using the IGA, EGDE's interface is designed to promote fast and effective auditioning of candidate effects. We have tried to make EGDE's interface nearly as efficient as IGA interfaces for graphical applications. To do this, EGDE uses an innovative ternary evaluation scheme, in which parameter sets are evaluated as one of *delete, use*, or *hold*. Knowing that many granular delay effects will not be useful to a user during early generations, our interface lets users immediately hear an effect, and just as quickly ignore it. When an effect is of more interest, rather than assign a score (which takes a certain amount of effort to determine, and is probably an arbitrary value anyway), the user merely flags it for further

[1]Download at http://www.cosc.brocku.ca/~bross/EGDE/.

reproduction. If the effect is even more preferable, it can be protected. And finally, the best effects can be saved, and of course, used in the main host application as desired.

The implementation of the granular delay engine is described in Section 2. The design of the IGA interface is discussed in Section 3. A walk-through of using EGDE is in Section 4. Conclusions, comparisons with related work, and future directions are given in Section 5.

2. GRANULAR DELAY ENGINE

Granular synthesis is a well-known audio synthesis technique in which audio is comprised of grains having durations between 5 to 100 milliseconds [14] [15]. The power of granular synthesis stems from its ability to combine and manipulate grains in countless creative ways. For example, CSound's *granule* operator has 22 parameters for controlling grain generation [4].

Granular delay is a variant of granular synthesis, in which grains are created from a finite-duration buffer of a (real-time) audio signal. These grains are mixed in with the current audio signal. The result combines characteristics of traditional "tape loop" delay, with the unique qualities of granular synthesis. An example of granular delay in a commercial product is Ableton Live [1]. Since Ableton is designed towards rhythmic music and real-time playback, its granular delay can be synchronized to the source audio's tempo.

Our granular delay effect is inspired by that in Ableton Live [1]. Along with many of the parameters used by Ableton's effect, our engine additionally permits tempo-synchronized parameter changes, which are described below.

EGDE's granular delay engine uses the following parameters, most of which are common with Ableton's granular delay: *grain density*, *pitch*, *delay*, *feedback*, *granular spray*, and *reversal probability*. Most of these settings include an *offset* parameter, which determines the range of values to be selected from. Similar to Ableton Live, EDGE can synchronize grain delays with a tempo. Parameter changes can occur at three levels of synchronization:

1. *Cloud*: A cloud is a duration of time in which all the parameters are locked together. If cloud duration is activated, then all parameter changes occur together within the cloud duration. They will only change in unison at the beginning of the next cloud.

2. *Beat synchronization*: If the parameter is set to synchronize on beats, it will do so with respect to k quarter-notes. Various parameters can synchronize at different note rates, which can result in complex rhythmic effects.

3. *Unsynchronized*: A parameter change can occur asynchronously, irrespective of the host clock.

With the above, activated cloud synchronization pre-empts beat synchronization and unsynchronized changes. Likewise, if clouds are not used, then beat synchronization pre-empts unsynchronized changes.

EGDE is implemented as a VST plug-in [16]. It relies on a host application from which to read and write audio signals, clock tempo information, and other information.

3. INTERACTIVE GENETIC ALGORITHM INTERFACE

Figure 1 shows EGDE's user interface. The population is on the left, the granular delay parameters on the top-right, and evolution controls on the bottom-right.

The population consists of 16 individuals. Each individual contains rating controls, an age indicator, and a "sample" audition button. The following tri-state rating scheme is used:

1. *Hold* (highest score): Clone this parameter set into the next generation, and also use it for reproduction.

2. *Use*: Use this set for reproduction, but do not clone it.

3. *Delete* (lowest score): Do not use this set at all.

Clicking a sample button will activate that individual. When sample is activated, the parameter set is loaded into the host, as well as the top-right parameter control section. It will then immediately process the audio, which will be heard by the user.

The *crossover/mutation balance* control determines the probability that crossover is selected for reproduction, verses mutation. If the user would rather use mutation exclusively, then this control should be set to 100. One-point crossover tends to preserve recognizable groupings of parameters, while N-point results in more varied permutations of parameter settings. The *Mutate % of the effects* control determines the probability that each parameter will undergo a mutation, should mutation have been chosen to perform reproduction. Mutation occurs on a gene by perturbing its current value, by the range specified by the *% mutation variance* control. Setting the variance low will result in a small tweak of a value, while a high value may result in a drastic alteration. *Next generation* will create a new population of parameter sets, using the current population ratings.

The parameter control section shows the parameter values (described in Section 2). They permit direct user editing of parameter values within the active chromosome, and result in immediate audio feedback of edited changes. Parameters can be *frozen*, and no longer subject to mutation. Freezing parameters stops the introduction of new values into designated parameter fields.

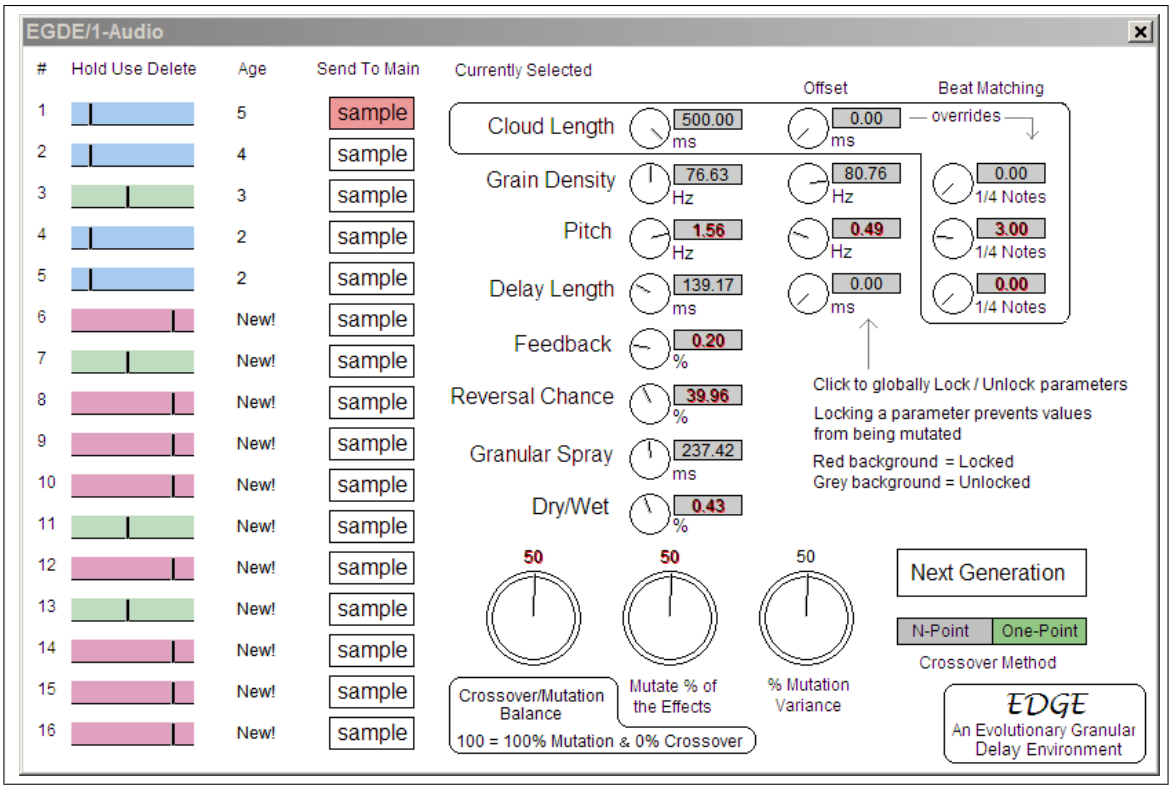

Figure 1. Granular delay interface.

4. USING THE SYSTEM

Many granular delay parameter combinations create drastic distortions of the audio, and most will likely be undesirable. A main goal of EGDE is to let the user quickly and effectively evaluate the population. First, a looped audio clip should be played by the host, which is immediately processed through EGDE's granular delay. The currently activated population member (parameter set) will be the one that processes the audio signal. The user merely clicks each sample button to immediately hear that parameter set. Depending on the duration of the loop being processed, an effect may be auditioned within seconds (or less, if the effect is clearly horrible), and hence the entire population in perhaps 20 to 30 seconds.

Evaluating individuals is performed quickly. The user decides if an individual should be held (protected), used, or deleted. To promote efficiency, individuals set to *hold* are kept on a hold state for subsequent generations. Otherwise, the automatically pre-set rating for non-held individuals is *delete*. Therefore, users will often only hit the *sample* button to hear an effect, and move on to the next individual, knowing that that effect will be deleted automatically. When an effect is considered interesting, then its rating may be changed to *use* or *hold*. This tri-state evaluation is a natural and intuitive fitness scale for interactive applications such as this, in which the user must make a series of reactive aesthetic judgements. The alternative, more conventional approach of evaluating each individual on a scale of (say) 1 through 10, is arguably less useful, since numeric evaluations have less intuitive meaning to the user. Numeric scores are not useful to the genetic algorithm anyway, given the small population being processed.

The interface is flexible with regards to the style of interactive evolution used. During the start of a session, the user is recommended to use a roughly equal combination of crossover and mutation. The mutation perturbation rate should be set high as well. As the user samples different individuals, many initial results will be deemed unacceptable, and will be pruned from the population. Interesting effects will be retained. As generations progress, a particular effect may be especially pleasing to the user. This parameter set may be set on *hold*; the crossover rate can be reduced, as can the mutation rate. Then evolution will be used to tweak the effect, until a final desired version is found. At any time, the user is free to directly edit granular parameters. This will directly alter chromosomes, and user changes will be available for use during subsequent evolution.

Exclusively mutation-directed evolution is possible as well, by turning the crossover rate to zero. As usual, the mutation perturbation rate can be fine tuned as generations progress. Mutation-based evolution can be used on multi-

ple individuals, or (more sensibly) just one individual set to *hold*. Of course, in the case of multiple individuals being mutated, the lack of crossover prevent shared characteristics from appearing in offspring.

The VST interface permits the user to save a selected parameter set as a parameter file. This can be loaded and used by the granular delay engine at any time. It can also be read and used outside the interactive genetic algorithm interface, since the granular delay engine can be used independently of the genetic algorithm interface. The user will therefore save favourite effects in a library, for future reference.

5. CONCLUSIONS

Our use of an IGA differs from the approach of Johnson [9]. His system uses mutation exclusively, applies numeric evaluation of individuals, and keeps the granular synthesis parameters hidden from user inspection and editing. The use of IGA's with general synthesis systems in [6] and [10] is comparable to our approach, although our IGA interface design is motivated for fast sample evaluation. The EGDE approach should be readily adaptable to other synthesis and effects technologies. Although work using other techniques towards granular synthesis, such as swarms [3] and cellular automata [13], is technically different than genetic algorithms, all share the goal of helping the user navigate the complex parameter space of granular synthesis.

Future revisions of EGDE are being considered. The granular synthesis engine could be enhanced with additional parameters, for example, parameter acceleration, inter-grain spacing, envelope control, panning, amplitude, and others. We are also considering the idea of having some sort of visualization for IGA population members, to give the user a visual representation of a parameter set. This would allow the user to see how generated parameters relate to their parents, which might be of use during auditioning and evaluation. Finally, more population editing controls for sharing parameters amongst the population members, and perhaps removing parameters from evolution entirely, are being considered.

Acknowledgements: The authors thank Michael Winter and Beatrice Ombuki-Berman for helpful comments. Research partially supported by NSERC Operating Grant 138467.

6. REFERENCES

[1] Ableton, "Ableton live," last accessed Mar 15, 2010. [Online]. Available: http://www.ableton.com/

[2] P. Bentley and D. Corne, *Creative Evolutionary Systems*. Morgan Kaufmann, 2002.

[3] T. Blackwell, "Swarm Granulation," in *The Art of Artificial Evolution*, J. Romero and P. Machado, Eds. Springer, 2008, pp. 103–122.

[4] R. Boulanger, *The CSound Book*. MIT Press, 2000.

[5] A. Burton and T. Vladimirova, "Generation of musical sequences with genetic techniques," *Computer Music Journal*, vol. 23, no. 4, pp. 59–73, Winter 1999.

[6] P. Dahlstedt, "Creating and Exploring the Huge Space Called Sound: Interactive Evolution as a Composition Tool," in *ICMC 2001*, 2001, pp. 235–242.

[7] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, 1989.

[8] A. Horner, J. Beauchamp, and L. Haken, "Machine tongues XVI: Genetic algorithms and their application to FM matching synthesis," *Computer Music Journal*, vol. 17, no. 4, pp. 17–29, 1993.

[9] C. Johnson, "Exploring Sound-Space with Interactive Genetic Algorithms," *Leonardo*, vol. 36, no. 1, pp. 51–54, 2003.

[10] J. Mandelis, "Genophone: An Evolutionary Approach to Sound Synthesis and Performance," in *EvoWorkShops*, 2003, pp. 535–546.

[11] J. Manzolli, A. M. Jr., J. Fomari, and F. Damiani, "The Evolutionary Sound Synthesis Method," in *MM'01*, 2001, pp. 585–587.

[12] J. McDermott, N. Griffith, and M. O'Neill, "Evolutionary Computation Applied to Sound Synthesis," in *The Art of Artificial Evolution*, J. Romero and P. Machado, Eds. Springer, 2008, pp. 81–101.

[13] E. Miranda, "On the Origins and Evolution of Music in Virtual Worlds," in *Creative Evolutionary Systems*, P. Bentley and D. Corne, Eds. Morgan Kaufmann, 2002, pp. 189–203.

[14] C. Roads, *The Computer Music Tutorial*. MIT Press, 1996.

[15] ——, *Microsound*. MIT Press, 2001.

[16] Steinberg, "Cubase," last accessed Mar 15, 2010. [Online]. Available: http://www.steinberg.net/