Genomic: Combining Genetic Algorithms and Corpora to Evolve Sound Treatments

Thomas M. Stoll
Buffalo, NY USA

Abstract

Genomic is Python software that evolves sound treatments and produce novel sounds. It offers features that have the potential to serve sound designers and composers, aiding them in their search for new and interesting sounds. This paper lays out the rationale and some design decisions made for Genomic, and proposes several intuitive ways of both using the software and thinking about the techniques that it enables for the modification and design of sound.

Rationale

While there are a good number of examples of systems for the evolution of sound synthesis parameters, there are fewer that consider real audio and its treatment. To the knowledge of the author, there are few systematic evolutionary approaches to modular sound design that would be directly relevant to the interests of a composer using *musique concrete* or other techniques using fixed media. The work that has been done mainly involves spectral representations of sound and operations in that domain (Caetano, Rodet, and others 2010), with some work done on sound design using interactive genetic algorithms (Johnson 1999), which it would seem to lend themselves naturally to use by composers and creators. Many important achievements are summarized in *Evolutionary Computer Music* (Miranda and Al Biles 2007).

This paper aims to describe the beginnings of a system that combines two heretofore unrelated techniques, namely genetic algorithms (GAs) and corpus-based processing (CBP) (Schwarz 2007) of audio. Some of this paper is speculative, in that it lays out possible avenues for exploration of the ways that these two techniques can complement one another and create a system that is defined by their combined usage. The musical (or sonic) goals are clear: the author wishes to evolve sound treatments that modify one source sound to sound more like another target sound, and to be able to track the transformations as they are explored by the GA. The author by no means claims to have conclusively reached either of these goals, but presents an overview of steps taken thus far towards solutions.

Intuitively, the use of GAs is a logical choice for a sound designer or composer: they both take a bottom-up approach to solve problems or optimize solutions. While there might be a perfect solution for a sound transforming seamlessly from source to target, the user may also be interested in intermediate results. Genomic, and corpus-based processing in general, is very much inspired by and perceived by the author as an extension of work by Trevor Wishart. In *Sound Symbols and Landscapes*, Wishart describes seamless sound transformations from a bell to male voice, an utterance of the word "listen" to birdsong, and the analogous nature of the sounds 'book-slam' and 'door-slam' (Wishart 1986). It is expected that higher level functionality will emerge once a lower level system is in place. As the goals become clear and attainable, one would be able to subvert more straightforward procedures and use this software in different ways that would take advantage of a fuller understanding of its inner workings.

Corpus-based system

The system is written in Python using SuperCollider as an audio engine or back-end. SuperCollider was chosen because of its flexibility as a synthesis engine and language and because of the opportunities for eventual use as a real time environment. Sound design is a critical component of the electroacoustic composer's process, and a composer can ideally design sounds in real time within SuperCollider, and then load those synthesis definititons—SynthDefs (see Figure 1), to use the proper nomenclature—into the Python system. Analysis data is stored either as JSON (JavaScript Object Notation, a common, lightly-structured data storage format) or as memory-mapped Numpy arrays. Further storage methods necessary for persistance and scalability are under investigation.

Sound design in Genomic is conceived in a modular way. In the first iterations of its design, there was one synthesis module that sampled a sound, modified it, and analyzed it. In the current version, each step in the processing chain is handled by a different module or node, and the treatment nodes, "EfxNodes", are swapped in as needed. In addition, two topologies have been explored: series and parallel. In a series layout, the result of each step is fed into the next processing step; in parallel, all stages are processed simultaneously and the results are mixed. In the former, the ordering is potentially important, as any subtractive or additive aspects of one node would be processed in different ways depend-

```
SynthDef(\efx_comb_mn, {
    |outbus=21, inbus=20, delay=0.2,
     decay=1.0, gain=1|
    Out.ar(outbus, (CombC.ar(
        In.ar(inbus, 1),
        2.0,
        delay,
        decay)
    ) * gain);
}).load(s);
```

Figure 1: SuperCollider SynthDefs define all nodes, including audio processing algorithms, referred to as "EfxNodes" in Genomic. This allows the user to quickly extend the set of possible transformations.

ing on the ordering. In the future, there are plans to encode the layout of modules as genes so that more complex layouts are possible. The head node could easily be changed to any number of sampling algorithms, and many different configurations of processing networks are possible. It is trivial to change the analysis routine and, as a result, the data that is to be collected.

GA system

The author has built a simple but effective genetic algorithm framework in Python suited for the present purpose. In its earliest versions, it was not dependent on or integrated within the corpus-based framework, but was used to fill a corpus with units/sounds generated by the evolved parameters. Currently, the system employs a corpus and much of the accompanying functionality. This allows the programmer to rely on the database to store information, and frees one to deal with sound material and analysis material as abstraction, only tracking index numbers or tags for sounds. In most applications, there are some "client-side" or "user-side" data structures that maintain the current state.

Representation of genes

The genes in Genomic map directly and indirectly to parameters for sound treatment. Each gene encodes one parameter or acts as an activator. The mapping is based on mapping from 8-bit integer representations for genetic operations to floating point, continuous valued parameters for audio realization. There are potential drawbacks to using such a method. Imagine tuning a notch filter—or any filter with a narrow effective range—with only 256 possible values for the center frequency. One would have a hard time precisely matching this filter to a sound in a mix to be attenuated, a real concern if these techniques are to be applied to real world applications. Possible solutions for this include increasing the size in bits of the representation for each gene or assigning two 8-bit genes to cover one 16-bit parameter. The latter brings to mind the "coarse" and "fine" controls on many synthesizers. Other solutions involve exploring floating point representations of genes (Budin, Golub, and Budin 2010).

Another aspect of Genomic that is loosely based on natural systems is the availability of activator genes. In much the same way that some genes might influence or "turn on" other genes or groups of genes, there are activator genes in Genomic. These activators perform a very simple task, but one that affects the resulting sound designs in profound ways. Imagine a scenario without activation genes. In this case, the system allows for a series of parameters that are set randomly that then evolve through mutation and crossover operations—the more or less standard genetic algorithm. Recall that genes are mapped to sound transformation parameters and that each parameter will contribute at least slightly to the output sound. One solution would incorporate a gain or scaling factor in each sound production phase, but even this would mask the potential problem of too many processes contributing to the mixture. Genomic can be set to use a set number of activator genes with a relatively larger pool of available sound treatment modules for a more selective and simple system. These activation genes need not be mere on-off switches; they could be encoded in different ways to influence the combinatoriality of modules. Table 1 shows a genotype with activator genes, parameter genes, and their raw and mapped values.

Fitness and similarity

While there is ample room for further variation and nuance, the main paradigm for the fitness function is some measure of similarity of a modified source sound to a target sound. Since there is no guarantee that a sound is modifiable such that some processed variant sounds exactly like a given target sound, the goal of similarity is one likely approached but not actually reached. Genomic offers the user flexibility to define different low level analysis routines, different methods of summarizing and segmenting data across audio files or sequences, and other high level control over goals.

The most simplistic—and ineffective—fitness functions would rely on comparing the raw audio data of two sounds. For example, consider the comparison of a sound and its attenuated variant. Significantly more effective, but still crude, are comparisons utilizing windowed analysis data, either Short-term Fast Fourier (FFT) coefficients, Mel-frequency Cepstrum Coefficients (MFCCs, a good general measure of timbre), or even Chromagram data, gathered from Constant-Q Frequency Transform (CQFT) terms collected into sets of pitch classes. The author has used raw MFCCs extracted over the length of each sound file (25 analysis points per second) and concatenated into a one-dimensional array for comparison with marginal results. The current system makes available to the user MFCC, Chromagraph, and amplitude envelope data averaged over temporal segments. For the purposes of this paper, analysis data is averaged over the entire duration of each (short) sound file.

One central feature of corpus-based (or unit-based) summarization of sound is that the definition of a unit within a corpus allows for the summarization of audio data over a temporal segment, or tile. This tile can be the entire sound file or sonic event, but it can also be a salient segment within a changing sound. The definition and implementation of a segmenting scheme is left up to the user, and

Gene	Raw, calculated values.	Mapping
Activator	256 / 23 = 0	Activate gene 0.
Activator	256 / 144 = 2	Activate gene 2.
Activator	256 / 151 = 2	Activate gene 2.
Gene 1	101 / 256 = 0.394	Gain stage gene - active.
Gene 2	212 / 256 = 0.828	Comb filter gene, delay time - not active.
	65 / 256 = 0.254	Comb filter gene, decay time.
	92 / 256 = 0.359	Comb filter gene, gain.
Gene 3	21 / 256 = 0.082	Filter gene, center frequency - active.
	102 / 256 = 0.398	Filter gene, gain.

Table 1: One implementation of activation genes. Activators are read to determine which genes are active, duplicates are ignored. The genes themselves are used as needed for active genes. Mutation and crossover still take place, even for unexpressed genes.

may be adapted to different types of material, just as basic analysis is. There is a tradeoff, but there can be real savings in memory and complexity by reducing data from 25 analysis points per second to 1 or 2, especially if there are hundreds of sounds or huge populations of potential treatments. The use of averaged data masks potential complexity and nuance—especially that which takes place on timescales smaller than an analysis period or segment length—and violates an assumption that one can capture all the salient features of an audio recording. The authors of FeatSynth (Hoffman and Cook 2007) address some shortcomings of using features in controlling a synthesizer parameterized over frames, and these concerns apply to Genomic's analysis over frames of audio data. An implicit premise is that as the features detected in analysis are necessary approximations of more complex data; the parameterization of Genomic's inputs allow for temporal changes that are equally difficult to pinpoint or directly control. Further work needs to be done in order to explore the properties of parameter encoding, analysis space, the mappings between them, and their time-based changes.

Finally, there are supplements to the fitness function. In addition to measuring similarity to some target, the author has explored measurements of diversity, which included measuring the distance of each member's data to an average over the entire population. Intuitively, this is expected to separate the individuals as they explore using parameter combinations. Further work is necessary to show how this strategy may be applied more effectively. In addition to diversity, there is the possibility to measure vitality of any individual. One may easily measure vitality by comparing the amplitude envelope of a prospective population member to that of the target's. This calculation may also be used in deciding whether to admit an individual to a population, even in the absence of other more intensive analyses.

Integrating Corpora and GAs

The most basic combined use of a GA system and a corpus uses the database of sounds as the repository for analytical representations of phenotypes—audio files are referred to indirectly during evaluation by fitness assessments based on comparisons of analysis data. This use case also uses a corpus as the tracking mechanism of a population's membership, past and present. Discarded members are removed, and the final state of the population is the entire corpus. In early versions of Genomic, there was no need for a corpus, as only the current population members were maintained in an array.

The use of a structured database enables the tracking of all members created throughout the history of a population. It allows, at least in theory, for a full history of said population if the user is able to record changes of population state. Individuals can also be assigned tags within the corpus, although the tagging mechanism is limited to one tag per individual. The possibility of recovering past "dead end" individuals at a later time does suggest techniques where past sounds are revived and reentered into the active population. In order to maintain groups of genomes and their resulting phenotypes-as-analysis-data, more than one corpus may be used. This concept might even be expanded to include realizations of multi-population and niched evolutionary schemes that use corpora themselves as (sub-)populations, with the ability to freely pass genomes among different corpora and quickly recalculate relevant fitness and statistical information.

The basic visual representation of a corpus presented here is a two-dimensional array with columns containing individuals. Figure 2 shows a newly initialized population with the seed/source sound in column 0, the target sound in column 1, and the members of the population in the remaining columns. As a population evolves, it is trivial to produce these views, and it is quite easy to animate the changing population over time. In Figure 2, one can clearly see some individuals that are incomplete; showing up in the graphic as black columns. These individuals were produced by a combination of parameters that resulted in silence or some other pathological state.

Experiments in Evolving Sound Treatments

Several experiments were run with Genomic to test its promise as a compositional tool. Although the tool is in need of refinement, it has shown encouraging results. In particular, the author investigated several variations on the fitness function and several topologies for the network of sound treating modules.The basic outline of the task is out-

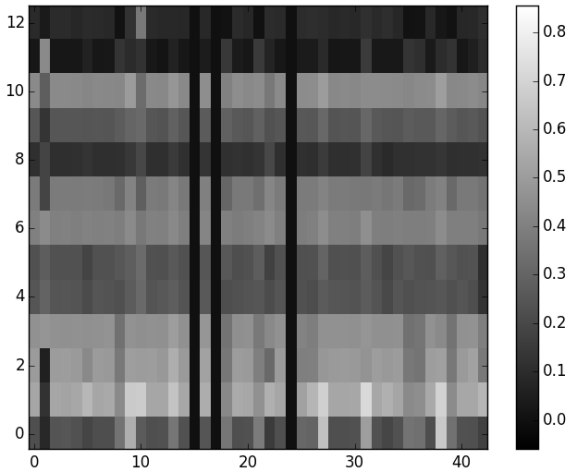

Figure 2: Initial members of a population with randomly seeded genomes. Each row represents a normalized MFCC bin.

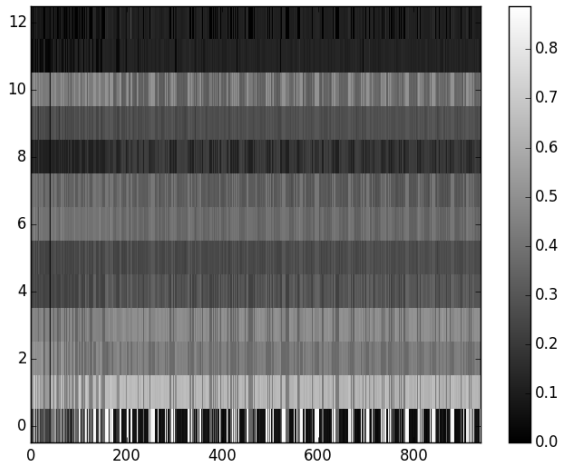

Figure 3: Oscillatory pattern in evolving population using equally-weighted fitness function.

lined in the following list:

1. Choose two sounds. One is the source and one is the target.
2. Analyze and add to a new corpus. Tag each with an integer: -1 and 0 respectively.
3. Generate random parameters for genomes for each population member.
4. Analyze and add each to the corpus. Tag each with an integer corresponding to the generation number.
5. For every age increment, mutate genomes according to some probability; add to the corpus and analyze newly mutated population members.
6. At multiples of the age counter, rank all the population members according to the fitness criteria using units extracted from the corpus.
7. Mate the fittest individuals (using crossover) to generate new individuals. New individuals replace weakest individuals.
8. Repeat steps 5-7 until some stopping condition has been met.

As discussed above, there are many nuances and variations on this general scheme that will have to be compared and tested. Preliminary tests have been more informal with the aim of demonstrating basic feasibility of this system. By looking at how the populations developed and evolved over generations, a few trends have become apparent. The first attempt at a fitness function used similarity between MFCC data as averaged over the entire sound file (a unit covering the entire duration). If the fitness function is realized as a similarity measure between population members and the target individual, there is a tendency to find a single solution rather quickly that dominates. It appears that the GA has difficulty exploring the solution space.

Overall, the reliance on timbre alone as a metric led to some development of novelty, but more often results in sound results that are inconsistent.

A more sophisticated approach leads to slightly better results. Instead of relying solely on similarity to a target, the author tried an experiment with a three-aspect fitness measurement. In addition to measuring similarity in timbre (MFCCs), the fitness function took into account similarity in amplitude envelope data and dissimilarity to the average of the entire active population's timbre measurements. This last criterion is a crude measurement of diversity within the population. Moreover, the three aspects were weighted, first equally then unequally. Equal weighting resulted in sounds that oscillated between general similarity to the source sound and general dissimilarity—see Figure 3.

By adjusting the weights of the fitness function, it was possible to evolve a population that gradually changes over many generations. The choice of weights is a matter of hand tuning, but Figure 4 does seem to suggest a gradual change towards the target sound. Again, this data is preliminary, but the sounds that emerge as a result of these processes do sound novel and do include combinations of effects that might not otherwise be arrived at by random or purposeful adjustments.

Conclusions and Future Work

The author has demonstrated a functional system that evolves sound treatments with a source sound, a pool of sound treatment algorithms, and a defined target sound. While the system is crude and in need of much refinement, it demonstrates potential as a tool for the composer or sound designer exploring his or her options for deriving novel sounds. The most pressing issues are the weighting of each component of the fitness function, the expansion of genes to use 16 bits per parameter, and the use of *segmented* sounds in the corpus-based analysis. Especially promising is the use of multi-objective fitness functions (Konak, Coit,

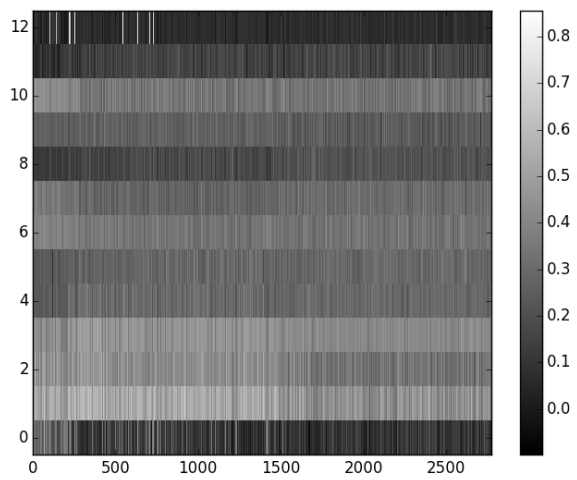

Figure 4: After 3000 generations with unequal weights, there is some evolution towards a goal.

and Smith 2006).

As easy as it is to speculate about possibilities for any system before it has been fully implemented, it is useful to consider some possible applications, as they can guide and influence development of Genomic. Major unexplored ideas involve expanding upon some preliminary attempts to generalize the system: there can obviously be more modules for sound design and more possible topologies for connecting them to one another. More complex sounds are possible with the inclusion of sound treatment algorithms such as convolution or delay-with-feedback effects within the system. Eventually, the author would like to generalize the software enough to incorporate direct sound synthesis and the use of multiple source sounds.

There is an opportunity to explore more complicated, but ultimately more rich, analysis methods. In addition to taking into account more of the temporal domain, one might mask or weight the analysis data itself. As a very simple example consider weighting the similarity measurement to take in to account the predominance of higher frequency ranges in the attack portion of a sound versus the decaying segment. More generally, Genomic should fully incorporate segmented sound analysis.

References

Budin, L.; Golub, M.; and Budin, A. 2010. Traditional techniques of genetic algorithms applied to floating-point chromosome representations. *sign* 1(11):52.

Caetano, M. F.; Rodet, X.; et al. 2010. Independent manipulation of high-level spectral envelope shape features for sound morphing by means of evolutionary computation. In *Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10)*, 11–21.

Hoffman, M., and Cook, P. 2007. The featsynth framework for feature-based synthesis: Design and applications. In *Proc. International Computer Music Conference (ICMC-07)*, 184–187.

Johnson, C. G. 1999. Exploring the sound-space of synthesis algorithms using interactive genetic algorithms. In *Proceedings of the AISB'99 Symposium on Musical Creativity*, 20–27. Society for the Study of Artificial Intelligence and Simulation of Behaviour.

Konak, A.; Coit, D. W.; and Smith, A. E. 2006. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety* 91(9):992–1007.

Miranda, E. R., and Al Biles, J. 2007. *Evolutionary computer music*. Springer.

Schwarz, D. 2007. Corpus-based concatenative synthesis. *Signal Processing Magazine, IEEE* 24(2):92–104.

Wishart, T. 1986. Sound symbols and landscapes. In Emmerson, S., ed., *The Language of Electroacoustic Music*, 41–60. Basingstoke: MacMillan Press.