Toward User-directed Evolution of Sound Synthesis Parameters

James McDermott, Niall J.L. Griffith, and Michael O'Neill

University of Limerick, Ireland
jamesmichaelmcdermott@gmail.com, Niall.Griffith@ul.ie,
Michael.O'Neill@ul.ie

Abstract. Experiments are described which use genetic algorithms operating on the parameter settings of an FM synthesizer, with the aim of mimicking known synthesized sounds. The work is considered as a precursor to the development of synthesis plug-ins using evolution directed by a user. Attention is focussed on the fitness functions used to drive the evolution: the main result is that a composite fitness function – based on a combination of perceptual measures, spectral analysis, and low-level sample-by-sample comparison – drives more successful evolution than fitness functions which use only one of these types of criterion.

1 Introduction

1.1 Motivation

A first-time user of synthesis software is typically overwhelmed with options. Synthesizer plug-ins with 30, 40, or more parameters are not uncommon, and the problem is often compounded by their non-linear interactions. Even an experienced user, while composing with a complex synthesizer, might sometimes prefer to pursue a desired sound through an intuitive process with immediate feedback rather than switching into analytical, "parameter-setting" mode. This is partly because the acoustic and psychoacoustic effects of moving within a synthesizer's parameter space are not well-understood.

EAs are often thought of as good methods for searching poorly-understood or oddly-shaped search spaces. Recalling that an EA can be driven by a user (as in, for example, [13]), rather than by a computer-calculated fitness function, we can say that EAs have the potential to control synthesis parameters "on behalf of" a user.

1.2 Previous work

Several authors (see [11] for an overview) have applied the techniques of Evolutionary Algorithms (EAs) to musical problems, including composition, musicology, and of most relevance here, synthesis.

Johnson [9] created a stand-alone graphical interface to the CSound FOF synthesis algorithm, which allows the user to direct the evolution of a population

of 9 sounds. He reports good results, including that the system allows easy and intuitive exploration of the FOF algorithm's possibilities.

Horner, Beauchamp and Haken [5] used Genetic Algorithms (GAs) in attempting to emulate the spectra of real instruments using FM synthesis. The GA was used to determine the best carrier-to-modulator frequency ratios and (time-invariant) modulation indices. They achieved good results, especially when using several carriers.

Horner, Cheung and Beauchamp [6] used GAs to find additive synthesis envelope breakpoints, and found that a GA performed better than a greedy algorithm, a "sensible" equal-spacing algorithm, and a simple random search algorithm in all cases.

Takala et al. [13] used LISP-like tree expressions ("Timbre Trees") to generate sounds. Evolving the sounds was then an application of Genetic Programming (GP), where evolution is directed not by an explicit fitness function but by user choices, made with reference to accompanying animations.

Blackwell and Young [1] used a particle swarm algorithm to control granular synthesis: their system is also capable of "interpretation", so that live musicians can improvise with the system.

Miranda [12] used cellular automata to control granular synthesis parameters in the Chaosynth program.

1.3 Method

In this paper we use unconstrained synthesizer representations which are closer to those seen by the typical end-user: filters, envelopes, amplitudes, etc, are controlled by continuously-variable parameters, in contrast to the free-form tree representation used by Takala et al. Thus, our approach is more general than [5] or [6], but more applicable to real synthesizer design than [13]. [1] and [12] are not directly comparable, since they do not use GA's; and finally, [9] is a user-directed system, which therefore does not use automatic fitness functions as developed here.

We see the development of synthesis software controlled by a user-directed EA as a three-step process. The first step, to be reported here, is simply to mimic known sounds using EA techniques. Several fitness functions, which measure the success of candidate sounds with respect to the target, are implemented and compared. The functions can be considered as moving from low-level, detail-oriented ones towards higher-level and perceptually-oriented ones, so leading towards our second step: attempting to generate sounds with a purely human fitness function (i.e. under user control). Here, the user will be attempting to generate sounds "to order" - whether the target is a sound with a particular metaphorical or verbally-described quality, or just "what sounds good". Success in this will be have to be measured in terms of both human satisfaction with the results, and psychoacoustically-based measures of timbre.

Finally, we hope to use the software in the design of experiments investigating timbral invariance: the phenomenon that psychoacoustic attributes such as

centroid and *roughness* constitute a many-to-one mapping from sounds to real values.

2 Experimental setup

2.1 Synthesizers

We choose to work with three different synthesizers (*simple additive*, *granular*, and *FM*), partly to ensure that any idiosyncracies of a single one don't have too large an effect on the results; however in this paper, for clarity, we report results for the FM synthesizer only, partly because FM is a more familiar and intuitive method of synthesis than granular. We use a single-carrier, single-modulator FM synthesizer with a peaking EQ filter: including envelopes and LFOs, it has 24 continuously-variable parameters.

2.2 Target sounds

The target sound for preliminary experiments was a 3-second sine wave at 440Hz, with an amplitude of 0.305 (where digital full scale is 1.0). For the main body of experiments, we used targets with 1, 4, 8, 16, and finally 50 partials (the upper limit for a 440Hz fundamental and a sampling rate of 44.1kHz), where the amplitude of partial n is given by $1/n$. The 50-partial target is therefore a bandlimited sawtooth wave.

We shorten the targets to 0.5 seconds for efficiency: this is justified since we treat length as a user-specified input parameter (see section 2.4).

As a preliminary test, we confirmed by hand that the synthesizer was capable of closely matching the simplest, 1-partial target sound.

2.3 Genotypic representation

Individual genomes were represented in GALib [14] as arrays of floating-point numbers, where the length and ordering of the array is fixed for the synthesizer. The genotype-to-phenotype mapping is performed by the synthesizer as it parses the genome into its own parameter format, and generates the corresponding sound.

2.4 Fitness functions

A fitness function, in this context, is a measure of similarity between the individual sound and a target. Several fitness functions were implemented and tested, each returning results of the form $1/(1+d)$ ($\in [1/2, 1]$), where d ($\in [0, 1]$) is the *distance* between the two sounds and is calculated in a different way for each function.

Uniform metric Digital audio signals can be thought of as discrete versions of real-valued functions of time. The *uniform metric* on L_1, the space of such functions, is defined as

$$d_U(x, y) = \frac{\sup_{t=0}^{T} |x(t) - y(t)|}{2} \tag{1}$$

for two functions or sequences x and y; the same expression can be used whether t varies discretely or continuously. We divide by a factor of 2 since the audio signal varies in $[-1, 1]$. The Uniform metric is commonly used in analytical mathematics but is too "severe" for this application, and is not used in this study.

Pointwise metric The most obvious definition for d, and the simplest generalisation of the Uniform metric, might be

$$d_P(x, y) = \frac{\sum_{t=0}^{T} |x(t) - y(t)|}{2T} \tag{2}$$

which is of course the discrete equivalent to integrating the difference between the two functions. We divide by a factor of 2 for the same reason as in the Uniform metric, and by T in order to keep $d_P(x, y) \in [0, 1]$. We'll call this the *pointwise metric*.

Discrete Fourier Transform metrics We define DFT metrics as follows:

$$d_{\mathrm{D}_L}(x, y) = \frac{\sum_{j=0}^{N} \left(\sum_{i=0}^{L/2} |X_j(i) - Y_j(i)| \right)}{C_L} \tag{3}$$

where L is the transform length, X_j and Y_j are the normalised outputs from the jth transforms of the input signals x and y, and N, the number of transforms for each sound, is determined on the basis of 2×-overlapping Hann windows. C_L, a normalisation factor, is determined by experiment.

Initial experiments showed that the transform of length $L = 256$ gave the best results.

Perceptual metric Research including [3], [8], and [10] shows that timbral attributes such as *centroid*, *harmonicity*, *attack time* and so on can be defined, measured, and used to measure the degree of human-perceived similarity between pairs of sounds. We define a simple *perceptual metric* (named d_{H_1} to indicate that we intend to define further human-perception oriented measures later) as follows:

$$d_{H_1}(x, y) = (1/3)(d_A(x, y) + d_C(x, y) + d_M(x, y)) \tag{4}$$

with the *attack time*, *centroid* and *mean amplitude* metrics defined as follows:

$$d_A(x, y) = \frac{|\mathrm{attack}(x) - \mathrm{attack}(y)|}{C_A} \tag{5}$$

$$d_C(x,y) = \frac{\sum_{j=0}^{N} |\text{centroid}(X_j) - \text{centroid}(Y_j)|}{C_C} \tag{6}$$

$$d_M(x,y) = |\overline{\text{amp}}(x) - \overline{\text{amp}}(y)| \tag{7}$$

where the normalisation constants C_A and C_C are determined by experiment. Finally, we define

$$\text{attack}(x) = \min\{t \ : \ x(t) = \sup_{s=0}^{T} |x(s)|\} \tag{8}$$

$$\text{centroid}(X) = \frac{\sum_{i=0}^{L/2} f(i)X(i)}{\sum_{i=0}^{L/2} X(i)} \tag{9}$$

$$\overline{\text{amp}}(x) = \frac{\sum_{t=0}^{T} |x(t)|}{N} \tag{10}$$

where $f(i)$ is the centre frequency of the ith frequency bin in the DFT X.

Composite metric We define a composite metric by summing the weighted results of several simpler measures:

$$\begin{aligned} d_{\text{Comp}}(x,y) = (1/9)d_{D_{256}}(x,y) + (1/9)d_{D_{1024}}(x,y) + (1/9)d_{D_{4096}}(x,y) \\ + (1/3)d_{H_1}(x,y) + (1/3)d_P(x,y) \end{aligned} \tag{11}$$

2.5 Pitch and length parameters

Early experiments showed that a fitness function based on the pointwise metric led fairly consistently to the evolution of silence. This happens because unless the candidate individual and the target are very close in frequency, they will go in and out of phase over the length of the sounds. Using a target sine wave 3 seconds long with a frequency of 440Hz, we found that candidate sine waves have fitness values as shown in fig. 1.

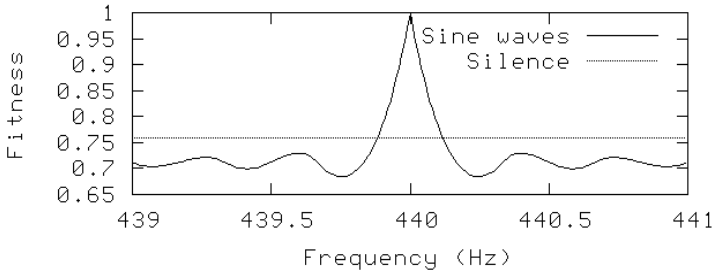

Fig. 1. Fitness values for silence and for sine waves, using a pointwise fitness function and a 440Hz sine wave target

Clearly, there is only a very small area of the entire frequency axis on which the GA can follow an upward gradient towards the target frequency. Evolution becomes a random search, and since silence scores higher than any wave outside this area, populations converge on silence from various directions.

We solve this problem by deeming pitch to be a fixed parameter of the synthesis. That is, we fix it at a certain value (here, 440Hz) and do not place it under the control of the GA.

A similar problem occurs in the case of the length of the sounds. It is not clear how to compare two sounds of different lengths - the obvious approaches (zero-padding the shorter is one; truncating the longer is another) both constitute "loopholes" that a GA can exploit in producing individuals which score highly but have obvious defects, such as trivial length. So, we deem length to be a fixed parameter also.

These decisions are justified since both pitch and note-length are typically controlled by the player of an instrument (eg by choosing a key to strike, and by holding it for a certain length of time), rather than by the sound designer. Even when these two jobs are performed by the same person, they're usually performed at different times and in different contexts.

2.6 GA parameters

We use a steady-state GA with 100 generations, population size 300, and replacement probability 0.5. The crossover probability is 0.5, while the Gaussian mutation operator is applied with a per-gene probability of 0.1. Selection is by a roulette wheel scheme.

2.7 Running the GA

For each target waveform, and for each of the 4 fitness functions *pointwise*, *perceptual*, *DFT 256*, and *composite*, we run the evolution 30 times. Each evolution is driven by a single fitness function: at the end of each run we have a single best individual, and its fitness according to the fitness function driving that run. For the purpose of comparison, we also evaluate the fitness of the best individual under each of the other 3 fitness functions.

3 Results

3.1 Results using 4 fitness functions

These graphs show the averaged best results over 30 runs using the FM synthesizer and targets consisting of 1, 4, 8, 16, and 50 partials. Each graph shows evolution *driven* by each of the four fitness functions *evaluated* by the function indicated in the caption. Error bars indicate the standard deviation for each data set.

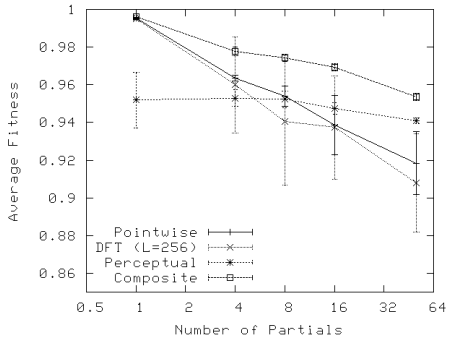

Fig. 2. Composite evaluation

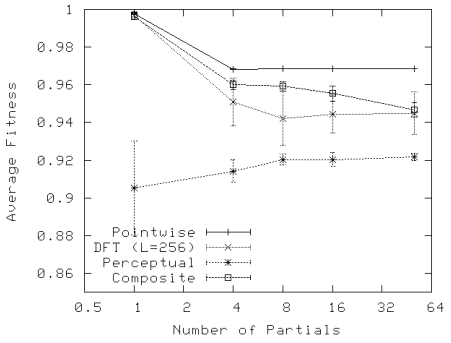

Fig. 3. Pointwise evaluation

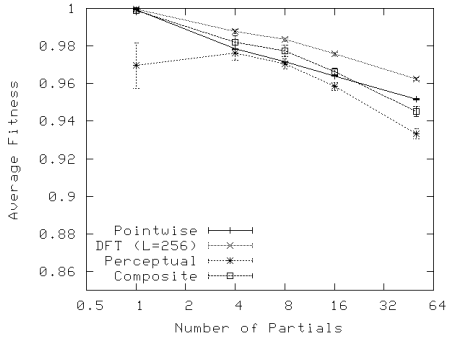

Fig. 4. DFT evaluation

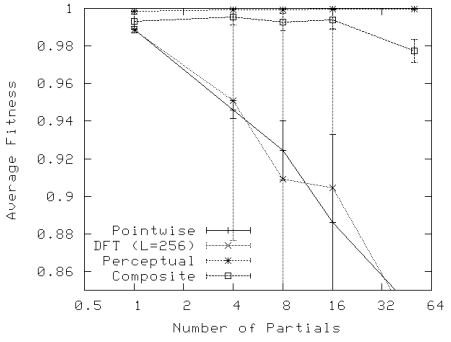

Fig. 5. Perceptual evaluation

Since our purpose is to compare the fitness functions (and find "the best"), we must be careful not to compare the fitness values they report against each other directly. Each fitness function performs better than all others when it is used as the evaluator.

The results are open to interpretation: there does not seem to be a single right way to objectively interpret them to decide which fitness function is better than the others. However, the main result is that the composite and perceptual fitness functions score quite well overall. They also exhibit smaller standard deviations,

as indicated by the error bars, while the DFT-based fitness function gives very high standard deviations.

In general all fitness functions are successful for the simplest targets, and performance drops off for the complex targets composed of many partials. Also, there is a strong anti-correlation between the perceptual and pointwise functions: the perceptual function scores well when evaluated by itself, and badly when evaluated by the pointwise function – and vice versa. This reflects the different kinds of information exploited by the two functions.

3.2 Scatter plots showing contrasting fitness scores for individuals under different measures

Each of these scatter plots shows 600 random individuals and 600 "best" individuals (30 for each of the 5 targets and 4 methods of driving evolution), evaluated under the *two* fitness measures indicated in the captions.

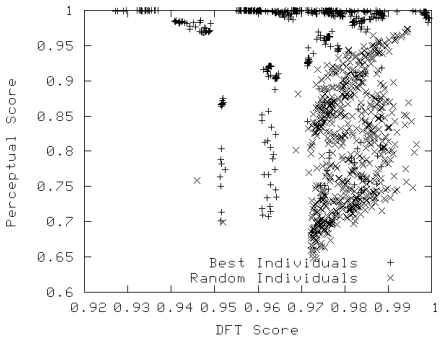

Fig. 6. Perceptual v. DFT

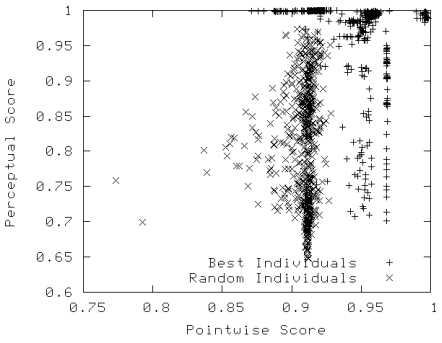

Fig. 7. Perceptual v. Pointwise

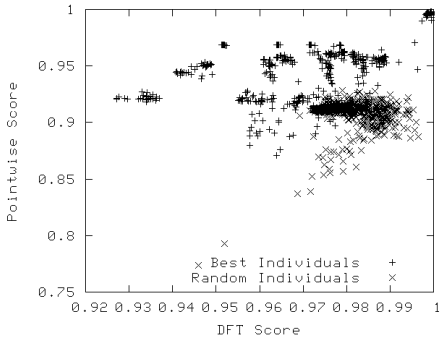

Fig. 8. Pointwise v. DFT

The fact that these plots do not exhibit a strong concentration on the bottom-left–top-right diagonal is evidence that the fitness functions are not strongly correlated: i.e. an individual scoring highly in one measure doesn't necessarily score highly in another. This is expected: if all fitness functions were highly correlated there would be no need to use more than one.

Where random individuals clump together, we can infer that large areas of the search space contain individuals whose scores coincide under one measure and under another: these features warrant further investigation. Where "best" individuals clump together, we can infer either of two things: local maxima in the search space, or patterning according to the target being pursued.

The vertical and horizontal bands, especially in the first two plots, indicate the non-uniqueness of the perceptual measure. (see section 1.2).

The small concentrations of best individuals in the top-right corner of each graph are probably the collections of individuals which were very successful in imitating the 1-partial target. For this target (only), all fitness functions do seem to be correlated.

4 Conclusions and Further Work

We consider that evolution has been successful enough to justify further work. Investigation is required to understand the paths which the GA follows towards its targets, and to explain some patterns in the search space suggested by the scatter plots.

Informal, subjective listening tests do suggest that all measures can drive successful evolution; and that the perceptual and combined measures might be the best by a slight margin. However, more rigorous listening experiments will be required to test this.

The perceptual measure performs adequately and of the three fitness measures it is the most transferable to evolving novel sounds rather than mimicking existing sounds. The version used in our experiments is a very simple one, being based on only 3 of the more important perceptual measures: attack time, centroid, and mean amplitude. It could be extended by adding components based on *harmonicity*, *roughness*, *irregularity*, the *odd/even harmonic ratio*, and perhaps others. Also, the components are currently weighted equally, but listening experiments performed with human subjects are expected to show that some components should be given more weight than others. After making these improvements, we hope to show that a perceptually-motivated measure can drive more successful evolution than any other measure.

The GA parameters mentioned in section 2.5 have fairly generic values. Tweaking one or more of them may help evolution to proceed more efficiently. Of particular interest will be the changes required to make evolution practical when under user control.

Other planned work includes applying the same methods to the parameters of a non-linear filter, instead of a synthesizer; implementing synthesizer and filter

plug-ins with GUIs; and running experiments in the area of timbral invariance (as explained in section 1.2).

5 Acknowledgements

Co-author James McDermott is supported by IRCSET grant no. 9262003.

References

1. Blackwell, Tim, Young, Michael: (2004) Swarm Granulator. *EvoWorkshops 2004 Proceedings*, 399–408. Berlin: Springer-Verlag.
2. Goldberg, D.: (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Reading, MA: Addison-Wesley.
3. Grey, John M.: (1976) Multidimensional perceptual scaling of musical timbres. *J. Acoust. Soc. Am.*, **61** (5)
4. Grey, John M., Moorer, James A.: (1977) Perceptual evaluations of synthesized musical instrument tones. *J. Acoust. Soc. Am.*, **62** (2)
5. Horner, A., Beauchamp, J., and Haken, L.: (1993) Machine Tongues XVI: Genetic Algorithms and their Application to FM Matching Synthesis. *Computer Music Journal* **17** (4)
6. Horner, A., Cheung, N.-M., and Beauchamp, J.: (1995) Genetic Algorithm Optimization of Additive Synthesis Envelope Breakpoints and Group Synthesis Parameters. *Proceedings of the 1995 International Computer Music Conference*. San Francisco: International Computer Music Association.
7. Horner, A., and Goldberg, D.: (1991) Genetic Algorithms and Computer-Assisted Music Composition. *Proceedings of the 1991 International Computer Music Conference*, 479–482. San Francisco: International Computer Music Association.
8. Jensen, K.: (1999) Timbre Models of Musical Sounds. Ph.D. Thesis, Dept. of Computer Science, University of Copenhagen.
9. Johnson, C. G.: (1999) Exploring the sound-space of synthesis algorithms using interactive genetic algorithms. *Proceedings of the AISB'99 Symposium on Musical Creativity*, 20–27. Brighton, UK: Society for the Study of Artificial Intelligence and Simulation of Behaviour.
10. McAdams, S. and Cunibile, J.-C.: (1992) Perception of timbral analogies. *Philosophical Transactions of the Royal Society* **336**, London Series B
11. Miranda, E. R.: (2004) At the Crossroads of Evolutionary Computation and Music: Self-Programming Synthesizers, Swarm Orchestras and the Origins of Melody. *Evolutionary Computation* **12** (2), 137–158
12. Miranda, E. R.: (2000) The Art of Rendering Sounds from Emergent Behaviour: Cellular Automata Granular Synthesis. *Proceedings of the 26th EUROMICRO Conference*, 350–355. Maastricht, The Netherlands: IEEE.
13. Takala, T., Hahn, J., Gritz, L., Geigel, J., and Lee, J.W.: (1993) Using physically-based models and genetic algorithms for functional composition of sound signals, synchronized to animated motion. *Proceedings of the 1993 International Computer Music Conference*, 180–185. San Francisco: International Computer Music Association.
14. Wall, M.: GALib, a C++ Genetic Algorithm Library. `http://lancet.mit.edu/ga/`, viewed 26 October 2004.