Play it Again: Evolved Audio Effects and Synthesizer Programming

Benjamin D. Smith[1]

Indiana University-Purdue University-Indianapolis, Indianapolis, USA,
bds6@iupui.edu

Abstract. Automatic programming of sound synthesizers and audio devices to match a given, desired sound is examined and a Genetic Algorithm (GA) that functions independent of specific synthesis techniques is proposed. Most work in this area has focused on one synthesis model or synthesizer, designing the GA and tuning the operator parameters to obtain optimal results. The scope of such inquiries has been limited by available computing power, however current software (Ableton Live, herein) and commercially available hardware is shown to quickly find accurate solutions, promising a practical application for music creators. Both software synthesizers and audio effects processors are examined, showing a wide range of performance times (from seconds to hours) and solution accuracy, based on particularities of the target devices. Random oscillators, phase synchronizing, and filters over empty frequency ranges are identified as primary challenges for GA based optimization.

Keywords: Sound Synthesis, Machine Learning, Adaptive Genetic Algorithms, Audio Effects

1 Introduction

Programming modern professional grade software synthesizers to discover or recreate desirable sounds requires extensive expertise and a time intensive process. Popular commercial products, such as Massive, FM8, Sylenth, and Ableton Live's native devices offer the user many hundreds of parameters resulting in countless potential combinations. Tuning an audio effects device or plug-in (such as compression, reverb, distortion, etc.), many offering similar complexity and possibilities, presents the user with the same challenge. As a partial solution most software synthesizers have extensive libraries of parameter presets to help novice users, and there is a distinct market for additional libraries targeting specific aesthetics. However exploring these libraries (which may contain thousands of presets) to locate a desirable sound remains a daunting process. If the user has a specific target sound in mind, the recreation process may be time prohibitive and highly disruptive to the creative work flow of composing and producing.

Automating the process of programming synthesizer and audio effects parameters to reproduce a target sound saves studio time and allows creators to focus on the creative process. Ideally this computational process will function

This is the author's manuscript of the article published in final edited form as:
Smith, B. D. (2017). Play it Again: Evolved Audio Effects and Synthesizer Programming. In Computational Intelligence in Music, Sound, Art and Design (pp. 275–288). Springer, Cham. https://doi.org/10.1007/978-3-319-55750-2_19

independent of the specifics of individual software synths or other audio effects processors, allowing users to work with their favorite devices, and it should function regardless of the origin of the the target sound, programming the synth or audio effects processor to reproduce the target sound as accurately as possible (given the device's capabilities and limits).

This work examines the use of Genetic Algorithms (GA) to develop presets for devices hosted in Ableton Live with the goal of being instrument independent, and aiming to flexibly target any VST or Live native instrument or device. While other research has proven largely successful in supporting individual instruments [9, 14, 16], this work's primary challenge is designing a flexible optimization algorithm capable of handling the variety and number of parameters that may be available (simple FM synthesis models, or effects devices, often have a few parameters while commercial implementations, such as Native Instruments Massive and FM8, have over a thousand parameters), without using privileged knowledge about the synthesis model in question or specific parameter assignments.

After background context, this paper presents the GA design and considerations in section 3. The implementation is presented in section 4, followed by evaluations in section 5. Conclusions and future work are found in section 6.

2 Background

The objective of reverse engineering synthesis parameters has been examined by a number of researchers, looking at different synthesis techniques and synthesizers, going back as far as the early 1990s [2–4, 13]. Most of these successful approaches employ evolutionary computing and GAs to automatically recreate specific sounds. The most popular synthesis technique seems to be FM synthesis [2–5, 9, 7, 14, 13, 16], perhaps due to the diversity of possible sounds while minimizing system complexity, enabling effective computation within hardware and CPU constraints.

More recent work has targeted specific FM software synths [16] and modular software synths [9, 14]. The former case looks at relatively simple software synthesizers, comparing the GA's ability to match timbre with that of human programmers, finding that the computer finds more accurate results in significantly less time. In the later cases the OP-1 synthesizer is employed, involving a much more complex process, due to the modular and non-deterministic aspects of this synthesizer.

Employing spectral analysis to inform fitness determination is used from the first examples, although the early cases [2, 4] employ fixed filter bank analysis techniques, moving to FFT analysis of the harmonics in a synthesized signal. Full FFT analysis is employed in recent work [14], along with amplitude envelope characterization for higher GA discrimination. An alternative [10] is to leverage Discrete Fourier Transforms, filling critical frequency bands in a custom perceptual model which aims to reflect human audition characteristics.

Other work [8, 16] employs Mel-Frequency Cepstrum Coefficient (MFCC) analysis to capture timbrel information and inform fitness decisions. MFCCs are

standard in natural language processing and are shown to work effectively in capturing both spectral and amplitude aspects of time varying sounds. This also has the advantage of providing a degree of fundamental frequency independence, where MFCC data will be close for similar spectra regardless of pitch (proving analogous to human perception which can identify an instrument independently of which notes are being sounded).

Attempting to design the whole synthesizer based on a given target sound is also being examined [1, 8], but presents a much more complicated domain compared to programming a given synthesizer. Optimizing the synthesis model or discovering completely new synthesis algorithms to accurately reproduce a given target sound is showing promise but has significant research ahead.

3 Design

3.1 Genetic Model

This work assumes the reader is familiar with the basics of GAs and presents the specifics of this particular design only. Selection, crossover, and mutation operators are all employed and presented in the following sections. Discussion of adaptive modifications and selection pressure considerations follow.

Each chromosome is a vector of device parameters, represented as floating point numbers, which are non time varying (i.e. the device parameters are set and remain fixed during sound generation and fitness calculation). Each individual is applied to the target device and the resulting audio is analyzed for GA selection and reproduction.

Selection Timbre matching is the primary problem being faced and the fitness function takes a standard approach [8, 16], using the first 13 cepstra of a windowed MFCC analysis. This has the advantage of representing the spectral energy in a consistent and comparable way, capturing both envelope (amplitude) and tone generator (spectral weighting) characteristics.

Fitness is based on the sum squared error (Err) between the target and candidate MFCC data (eq. 1, where t is the time sequence target MFCC data, X is the individual solution MFCC data, W is the number of windows). Window and hop sizes from 10 to 90 milliseconds were empirically tested and found to have negligible impact on solution quality (data in this paper uses a 20 ms window with 10 ms overlap, which is common to many examples in the literature). Normalization is not applied at the audio level as this would preclude optimization for volume, gain, and dynamics parameters. In this case matching based on loudness and amplitude envelopes is desirable.

$$Err(t, x) = \frac{\sum_{i=1}^{W} \sqrt{(\sum_{j=1}^{N_x} (t_{i,j} - x_{i,j})^2}}{W} \tag{1}$$

Unlike FFT or simple spectral-mean based fitness functions, measuring Euclidean distance between cepstra is problematic due to the widely varying ranges

of each order (i.e. the first cepstra has a range of $[-96, 96]$ and the twelfth might be $[-0.001, 0.001]$). Unaddressed this imbalance incidentally weights the orders, diminishing the potential for higher order cepstra to contribute to the selection process. To provide balance to the MFCC vectors the standard deviation of each cepstra over the entire dataset is tracked and used for normalization. While this analysis and fitness evaluation has proven adequate, and is based on findings in other sources, a detailed comparison of fitness variants (using other spectral analysis methods) and additions would be required to prove the most efficient and effective solution.

Roulette-wheel selection is employed, and in-breeding (where an individual is chosen to reproduce with itself) is prevented. A small percentage (4-6%) of *elite* individuals are carried through to the next generation unchanged (individuals with the highest fitness are allowed to survive, in addition to participating in breeding the next generation).

Crossover Three crossover models were applied and examined for this system. Since the chromosomal elements (the device parameters) are independent, their cardinal order is probably coincidental and relationships between neighbor elements may or may not be significant (depending on the software designer's choices in parameter ordering). This would seem to indicate Uniform Crossover as optimal, however alternatively, the target device's parameters may be grouped in a meaningful way (such as in groups of filter bank parameters, oscillator parameters, amplitude envelope parameters all appearing sequentially) and using single point, or dual point crossover may be more effective in maintaining these relationships. These three crossover models (single-, dual-, and uniform crossover) were all examined and the results were inconclusive: all three arrived at equally satisfactory winning individuals, and any variances in efficiency could not be attributed to the crossover model alone.

Mutation Several mutation models were tested for impact on efficiency. Uniform replacement and scaled Gaussian addition with both range clamping and range fold-over were implemented and tested. Gaussian addition (using a normal distribution with center=0 and standard deviation=1.0), clamped to $[0, 1]$ has the effect of increased probability for parameters to sit at the edges of their range, with diminished exploration of the middle ground. Uniform replacement functions better, but this model has trouble, after locating a near optimal solution, of resolving the final small adjustments to find the ideal match. Gaussian addition where the parameter is folded over the range $[0, 1]$ in ping-pong fashion appears to provide the advantages of both the previous two models, although empirically conclusive data has yet to be produced. The lack of efficiency in making the final adjustments may further indicate the need for an adaptive Gaussian distribution to encourage mutations of finer granularity as convergence is being approached.

Termination Convergance of the population is either taken as a solution making a $> 99.99\%$ fitness match, or observing a < 0.001 change in the magnitude of the winning fitness over 200 generations. Both conditions are necessary due to the unknown particularities of the target device. Many digital synthesizers and audio effects employ random oscillators and noise generators, which will render a 100% match virtually impossible to achieve. Further, while the effectiveness of recreating sounds that originated on the same synth or effect is an initial concern, attempting to get a best approximation of target sounds from other sources (such as analog synths, or non-synthetic audio) will have further convergence difficulties.

3.2 Adaptive Genetic Algorithm

Due to the potential range of audio processes that may be targeted, determining fixed crossover and mutation coefficients is problematic. The examination of a single synthesizer or synthesis technique can empirically test these rates and set them for optimal convergence. However, in order to be independent of a specific synthesis engine, and support any number of parameters in the model (typically 6 to 200), an adaptive crossover and mutation algorithm is employed [6, 12]].

The objective of the adaptive model is to maintain adequate population diversity, preventing local minima solutions, and improve movement through exploration and exploitation phases. Here, population diversity is calculated simply using the maximum F_{max}, mean $\bar{F}$, and individual parent fitness F' during an evaluation stage [12]. The adaptive probabilities of crossover P_c and mutation P_m are:

$$P_c = K_c\Big(\frac{F_{max} - F'}{F_{max} - \bar{F}}\Big) \tag{2}$$

$$P_m = K_m\Big(\frac{F_{max} - F'}{F_{max} - \bar{F}}\Big) \tag{3}$$

Both probabilities are constrained in the range $[0, 1]$ and constants are set to allow maximum exploration $K_c = 1, K_m = 0.5$ when population diversity declines.

In practice this adaptive approach results in overpopulation of copies of the best solutions, due to $P_m = 0$ when $F' = F_{max}$, preventing effective solution space exploration. This is partly solved by imposing a minimum P_m to disturb the winning solutions [12], forcing a range of $[0.005, 0.5]$ on the mutation coefficient. Observations showed crossover being applied to around 10% of the parents in each generation, again limiting exploration and exploitation. Using the mean of both parent's fitness values (to encourage crossover of the most fit individuals) brings this rate up to $\approx 30\%$

$$F'' = \frac{F'_1 + F'_2}{2} \tag{4}$$

$$P_c = K_c\Big(\frac{F_{max} - F''}{F_{max} - \bar{F}}\Big) \tag{5}$$

3.3 Selection Pressure

The squared error fitness function frequently fails to make progress towards convergence, based on observational data. This appears to be due to inadequate selection pressure (i.e. the gradient across the population is too flat). Thus the following selection pressure S constant, and coefficient P_s are employed to calculate the fitness of each individual Fit_x:

$$P_s = S\Big(\frac{\bar{F}}{F_{max}}\Big) \tag{6}$$

$$Fit_x = \Big(\frac{1}{Err_x + 1}\Big)^{P_s} \tag{7}$$

The impact of various values of S is examined in the Evaluation section 5, below.

4 Implementation

The system is implemented and tested as a combination of Ableton Live sets using built-in software instruments and audio effects devices, Max for Live devices, and a Max external encompassing the GA model. Due to the nature of the software employed the model has to run in real-time (in Live). First each individual chromosome is used to configure a software instrument or processor, MIDI notes (if the target is a synthesizer) or an audio clip is played (if the target is an audio processor), and the output is recorded. An MFCC analysis is performed and the data is transmitted back to the GA for storage and comparison as fitness features (see above).

A primary challenge facing this implementation is the timing priority afforded by Ableton Live. Due to the asynchronous connection between Max and Live the initiation of audio sample playback can occur with latencies between $0 - 75ms$. This results in some unknown amount of erroneous artifact audio at the beginning of each MFCC analysis. The successful implementation of this project required delaying sample playback to align with the signal vector rate and the hop point of the MFCC analysis (to remove leading zeros in the audio recording), and then a least-squares comparison between each chromosome's MFCC features and the target across up to 20 time offsets to locate the most probable alignment.

The evaluation commenced by randomizing the target device's parameters and playing 2 seconds of notes (for synthesizers) or one of 8 pre-selected audio clips (through audio effects). The audio clips were selected to encompass a wide range of dynamic spectral content (including drum loops, complex synth tones, environmental noise, etc.). In addition to the real-time constraints, parameter transmission between Max and Live is a primary limiting factor resulting in epoch times between 5 and 50 seconds, depending on population and chromosome size. Reaching adequate epoch counts can thus be time prohibitive (5000 evolutions could take ≈70 hours) with desirable population sizes (of 600-1000 for larger chromosomes)[11, 15], and thus smaller devices and population sizes are employed in this initial exploration.

5 Evaluation

Evaluation of the proposed system is carried out in the following tests, examining selection pressure values and performance over different synthesizers and audio devices with different sound sources. Data for a synthesizer, a distortion effect, a three-band equalizer, a phaser, and a reverb unit is presented and discussed in sections 5.1 and 5.2 below. Identifying the highest degree of accuracy as well as the efficiency of the optimization process (in terms of population, epochs, and total execution time to convergence) is of primary concern.

The primary test for synthesizer and audio device programming is perceptual, proving the sonic accuracy of the arrived at solution(s). This proof can be approached with both empirical and perceptual measures. MFCC analysis is accepted as an accurate reflection of human perception and is used extensively in natural language processing for this reason, thus statistics of the fitness values are presented as a characterization of perceptual accuracy (the alternative of an extensive study with human subjects, is beyond the scope of this initial work).

The accuracy of a reverse engineered solution, in the literature, is typically shown by taking a distance measure between the converged device parameters (chromosome) and the target parameters. However, this measure must be qualified for several reasons. In a constrained space (where each dimension is confined to range [0,1]) the maximum distance is only achievable when the target is at one of the limits. If the target is in the center, the possible error (distance from target) is greatly diminished and thus even very bad solutions will appear to have low error rates. In the metrics presented herein this is accounted for by first computing the maximum possible error using the target parameter vector (of length N with parameters p):

$$E_{max} = \sqrt{\sum_{i=1}^{N}((|0.5 - p_i| + 0.5)^2)} \tag{8}$$

The parameter mismatch error is calculated as a percentage of the maximum possible for a given target.

Secondly, some parameters in a synthesizer or audio device may be part of dependent chains, such as a bypass switch which negates the effect of dependent parameters. Based on the state of the switch it is impossible for the GA to ascertain the state of the bypassed parameters. In extreme cases the target may be completely silent, allowing many parameter sets to achieve 100% fitness but have a high parameter mismatch.

Multiple solutions may be possible, presenting a third consideration. For example, many synths have both macro and micro tuning parameters (half steps and cents) which provide different methods of obtaining the same transposition, multiple gain stages which can obtain equal output levels, or many ways of obtaining the same wave form through different oscillators. These alternative solutions achieve a sonic match but evaluate badly when comparing the parameters (see Figures 2-5).

In order to make this system generalized and abstracted from the specifics of the target audio device all parameters are represented as floating points in range [0,1]. These are then scaled appropriately when applied to the device (most commonly to MIDI control range [0,127]). However, many audio device parameters are treated as switches or selectors with incremental steps (such as bypass switches, filter type or wave form selectors, etc.). In these cases any value within the quantized range is treated equally, i.e. [0,0.5) is "off" and [0.5,1] is "on." If the GA arrives at the correct setting it may still show as being inaccurate due to this incremental quantization.

Finally, a special subset of problems faces audio processing devices wherein the source audio may not reveal the impact of the effect. For example, a filter impacting a range of frequencies that are not present in the source (such as a very low filter on high material), or dynamic effects on a static source.

While this system has been tested with numerous synths and audio devices, an evaluation of three devices is presented here: a 33 parameter synthesizer, a 20 parameter phaser effect, and a 7 parameter distortion.

5.1 Synthesizer: Electric

The Electric synthesizer is a native device in Ableton Live, intended to reproduce a wide range of electric keyboard and piano sounds. It offers 33 parameters (see fig. 3 & 5). For evaluation purposes all target parameters were fully randomized for each run, and a 4 note pattern (Cs in 3 octaves) with different velocities for each note is employed. Two example executions (maximum and mean fitness for each generation) are shown in the following figure (fig. 1).

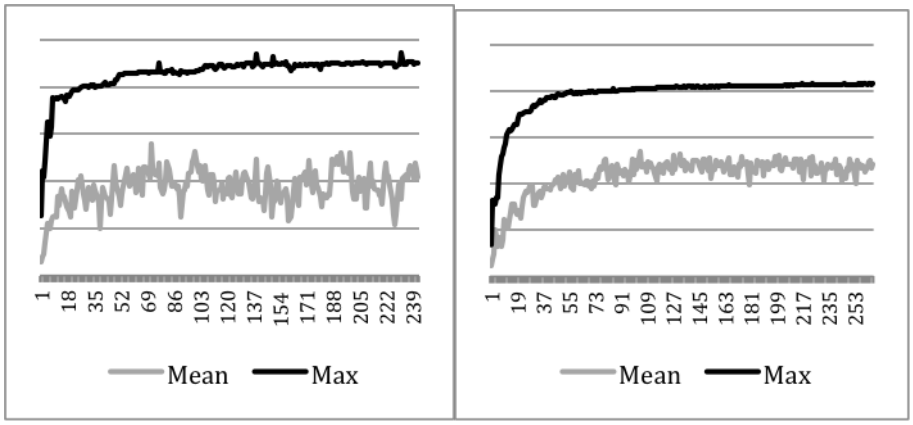


Fig. 1. Maximum and Mean fitness over epochs, for start of two sample runs (scale of [0,1]). Diversity is reflected by gap between mean fitness and maximum fitness.

As expected, the adaptive crossover and mutation rates ensure population diversity throughout the runs (as reflected by the mean and maximum fitness

across all epochs, fig. 1). The sonograms for the first target and solution show a high degree of accuracy (fig. 2), with a parameter mismatch magnitude of 2.113 (91.1% accuracy in parameter settings). However the solution arrived at, while perceptually very similar, is different from the target (see fig 3, Note: Mallet parameters appear to be compensated by Fork parameters in solution).

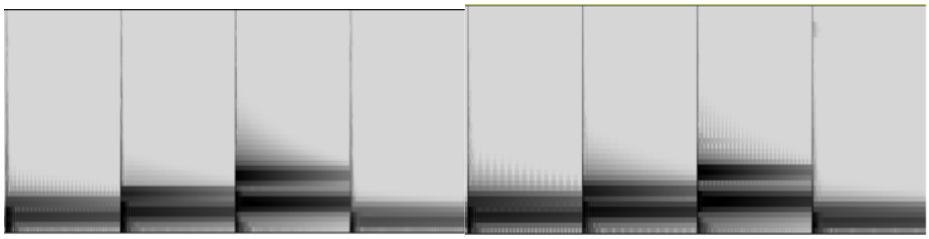


Fig. 2. Sonogram (frequency over time) of Electric target (left) and best fit solution (right).

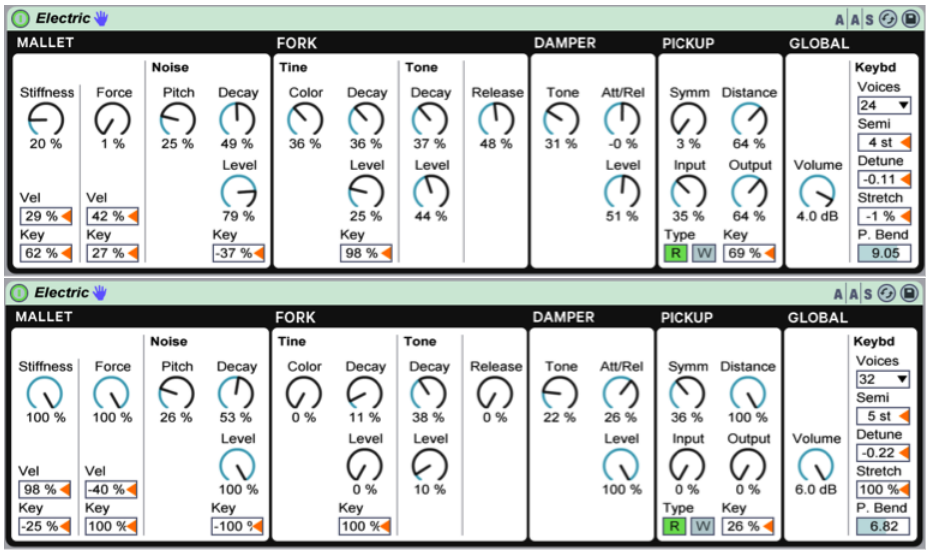


Fig. 3. Parameters for target (top) and best fit solution (bottom), resulting in spectra shown in fig. 2.

A second selected example (fig. 4 and 5) finds a closer parameter solution with a mismatch magnitude of 1.823373 (93.14% accuracy in parameter settings). Note in both example cases the solution is played at a different pitch (see Semi and Detune parameters), yet the timbrel match is very close.

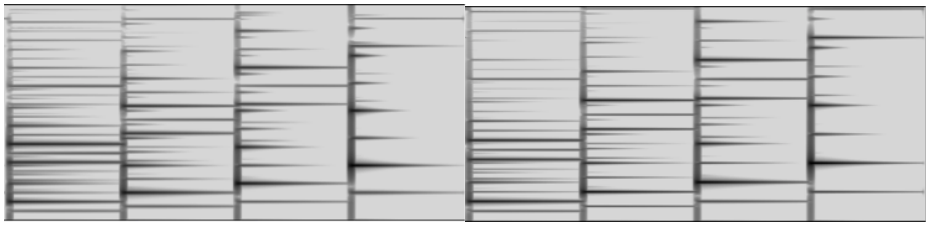


Fig. 4. Sonogram (frequency over time) of Electric target (left) and best fit solution (right).

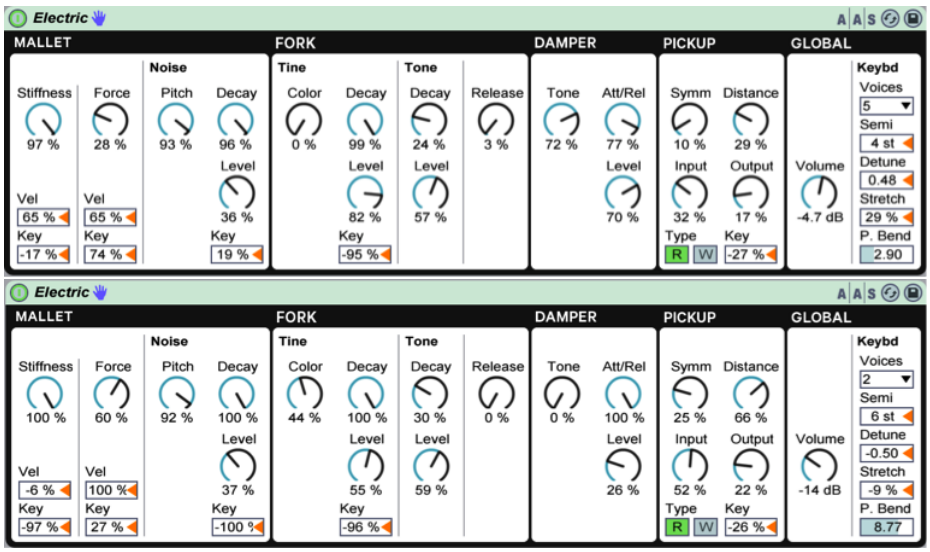


Fig. 5. Parameters for target (top) and best fit solution (bottom), resulting in spectra shown in fig. 4.

5.2 Audio Effects

Two audio effects are examined below comprising a distortion and a phaser.

Distortion The Live native Overdrive implements a typical distortion effect and has 7 parameters. This system arrives at a solution in an average of 57.47 epochs (N=109, 12 failures to converge), using a population of 128, comparing with a single input sound file. Unlike the Electric synthesizer there do not appear to be any alternative solutions: the maximum fit solutions are nearly perfect matches with the target parameter set. The failures are from extreme frequency ranges versus the sonic content of the source material (e.g. the distortion filter frequency $< 60hz$ on a middle C4 note).

Comparison of time-to-converge for different selection pressure coefficient values is shown in Figure 6. Based on this data values for S around 6 appear

optimal, for this device. Both higher and lower values take longer to converge, in both worst-case and average scenarios.

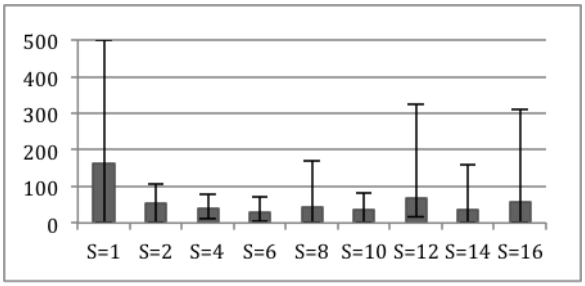


Fig. 6. Epochs to converge for values of S on the Overdrive/Distortion device. Gray bar is mean, error bars indicate maximum and minimum epochs-to-converge ($N = 109$). $S = 6$ is the best mean and maximum point.

Phaser The Phaser device in Live is a series of all-pass filters designed to create phase shifts in the frequency spectrum of a sound. It includes an attack/release envelope in addition to a low frequency oscillator (LFO) to drive the stereotypical phase sweep effects. This device poses a challenge to the GA due to the synchronization of the LFO. The same chromosome evaluated many times produces different fitness values depending on the starting phase of the LFO (which is purely a function of time since launch of Live). However, the GA arrives at sonicly accurate results, yet they statistically compare poorly.

Fig. 7. Sonogram of example unprocessed original sound sample (synthesizer tone) used in Phaser device tests.

An example is shown in figure 7 (the original, unphased source sample), where the highly processed target (fig. 8, left) is answered by a best-fit solution (fig. 8, right). Visually, and aurally, the solution appears to be a match, however it is only 72% accurate in MFCC fitness and has an 8.64% parameter mismatch (fig. 9). Due to the inability of the GA to arrive at a 100% match a comparison is made across values of S (selection pressure coefficient) using the fitness of the

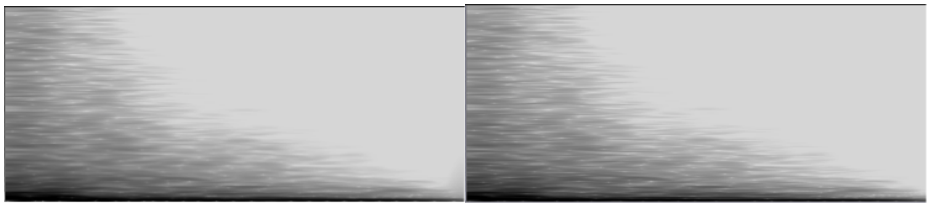
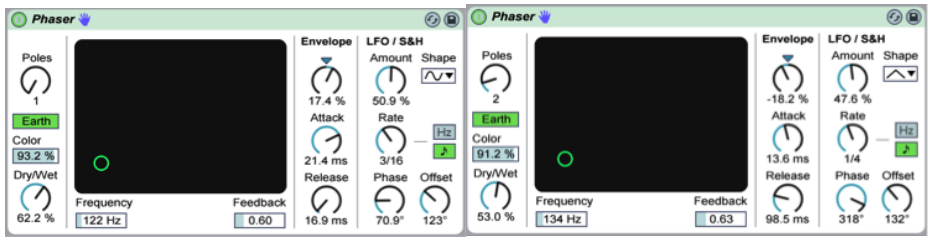


Fig. 8. Sonogram of phased target (left) and best solution (right), 72% average MFCC match.



h

Fig. 9. Phaser device parameters for target (left) and best solution (right), 8.64% average parameter mismatch.

best final solution (Fig. 10). The peak is around $S = 8$, close to the empirical evidence from the Distortion data, above.

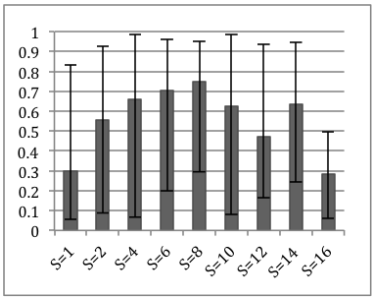


Fig. 10. Final fitness for Phaser with different Selection Pressure coefficient values. Gray bar indicates mean, error bars show maximum and minimum final fitness. (N=46). The best mean performance is seen with S=8.

6 Improvements

Examining convergence failures informs further improvements to the system, leading to the identification of several contexts that provide additional chal-

lenges. The following conditions either cause convergence on an inaccurate solution, or require extreme numbers of generations to resolve.

1) When an audio effect (or synthesizer parameter) is minimally present, such as a dry/wet parameter that is $< 4\%$, the original sound is very nearly unaffected and while the GA appears to identify the presence of some audio manipulation it does not have enough impact to properly discriminate an accurate solution.

2) As previously noted, random oscillators or noise generators pose particular problems due to the non-determinism of their output. The GA continues to search, exploring and going over old areas in an attempt to locate a stable match. In these cases even the best match may evaluate poorly in consecutive epochs depending on the state of the random generators. Other audio analysis models could characterize the overall timbre, augmenting the MFCC analysis and allowing the GA to accept these noisy conditions.

3) Oscillators, particularly low frequency effects, that are not phase locked pose a specific challenge. In cases where the phase is reset on a key trigger the GA is successful, but if the oscillator runs freely there is a high probability that even the best solution will compare unfavorably with the original. For many native devices Live provides the option for oscillators and generators to be synchronized to the master clock and configured in units of beats rather than hertz. Because the GA is evaluating continuously, regardless of the master clock, these generators will produce phase mismatch errors and disrupt identification of the optimal solution. Synchronizing to the master clock is generally undesirable since this would require delaying chromosome audition, greatly increasing operation time per epoch. However, this may be implemented as an optional operating mode based on user preference.

4) Effects that operate on frequency ranges not present in the source material prevent convergence on an optimal solution. This was the primary cause of failures with Distortion and EQ effects. This is a challenge for the randomized parameter testing method employed herein, but does not pose an issue for actual application.

7 Conclusions

This work shows a high degree of success in being able to recreate or reverse engineer parameter presets, with performance varying based on specifics of the target audio process. In the best conditions an audio effect can be fit in less than one minute. However, the limits and idiosyncrasies of the system remain to be further investigated, beyond anecdotal observation. Certain conditions prevent an optimal solution from being discovered, but the particularities of those circumstances remain to be fully identified and addressed.

We have shown the ability to match randomly generated configurations or presets, however human musical preferences are rarely random, and may exhibit tendencies that either simplify or extend the challenge to a GA based solution. Further examination of performance versus human generated presets and sound samples is required to determine efficacy in more realistic applications. Another

common approach and interest in GA based synthesizer programming is to employ targets originating from other sources (either other synthesizers or natural, sampled sounds) [8]. Recreating presets from samples of analog synthesizers, or acoustic instruments, can pose additional challenges and remains to be examined in the context of this system.

The potential for creators and producers to employ this system in creative musical and sound work is distinct and may provide both time saving and transformational benefits. Rather than spending hours programming synths and audio devices this system can quickly provide a ready starting point for tweaking. Further, it may be able to provide new sounds based on matching non-synthetic sounds (such as vocalizations), or by using targets as guidance where one is looking to identify sounds that combine elements from multiple sources in one result. The optimal solutions between samples with contrasting spectral signatures has the distinct potential to discover unique and compelling sounds.

References

1. Garcia, R.: Growing sound synthesizers using evolutionary methods. In: Proceedings ALMMA 2001: Artificial Life Models for Musical Applications Workshop,(ECAL 2001) (2001)
2. Horner, A.: Double-modulator fm matching of instrument tones. Computer Music Journal 20(2), 57–71 (1996)
3. Horner, A.: Nested modulator and feedback fm matching of instrument tones. IEEE Transactions on Speech and Audio Processing 6(4), 398–409 (1998)
4. Horner, A., Beauchamp, J., Haken, L.: Machine tongues xvi: Genetic algorithms and their application to fm matching synthesis. Computer Music Journal 17(4), 17–29 (1993)
5. Johnson, A., Phillips, I.: Sound Resynthesis with a Genetic Algorithm. Imperial College London (2011)
6. Karafotias, G., Hoogendoorn, M., Eiben, A.E.: Parameter control in evolutionary algorithms: Trends and challenges. IEEE Trans. Evolutionary Computation 19(2), 167–87 (2015)
7. Lai, Y., andDer Tzung Liu, S.K.J., Liu, Y.C.: Automated optimization of parameters for fm sound synthesis with genetic algorithms. In: Proceedings of the International Workshop on Computer Music and Audio Technology. Citeseer (2006)
8. Macret, M., Pasquier, P.: Automatic design of sound synthesizers as pure data patches using coevolutionary mixed-typed cartesian genetic programming. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation. pp. 309–16. ACM (2014)
9. Macret, M.M.J.: Automatic Tuning of the Op-1 Synthesizer Using a Multi-Objective Genetic Algorithm. Doctoral dissertation, Simon Fraiser University, Vancouver, CN (2013)
10. Riionheimo, J., Välimäki, V.: Parameter estimation of a plucked string synthesis model using a genetic algorithm with perceptual fitness calculation. EURASIP Journal on Advances in Signal Processing 8(1–15) (2003)
11. Rylander, S.G.: Optimal population size and the genetic algorithm. Population 100(400) (2002)

12. Srinivas, M., Patnaik, L.M.: Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics 24(4), 656–67 (1994)
13. Tan, B., Lim, S.: Automated parameter optimization of double frequency modulation synthesis using the genetic annealing algorithm. Journal of the Audio Engineering Society 44(1/2), 3–15 (1996)
14. Tatar, K., Macret, M., Pasquier, P.: Automatic synthesizer preset generation with presetgen. Journal of New Music Research 45(2), 124–44 (2016)
15. Weise, T., Wu, Y., Chiong, R., Tang, K., Lässig, J.: Global versus local search: The impact of population sizes on evolutionary algorithm performance. Journal of Global Optimization pp. 1–24 (2016)
16. Yee-King, M., Roth, M.: Synthbot: An unsupervised software synthesizer programmer. In: Proceedings of the International Computer Music Conference. Ireland (2008)