

Composing with Evolutionary Computing

BRANDON YE, DECEMBER STUART, and VIVIAN YAO, Queen's University, Canada

The use of evolutionary algorithms to produce music is a fascinating and ever-improving domain. However, the capacity of evolved music to inform us of the algorithm itself is not widely explored. In this report, we present a novel evolutionary process for a computational population to 'play' digital sounds. We rely on evolving individuals in the frequency-domain of an audio signal - in the Fourier sense - to develop an algorithm that can handle arbitrary audio signals in computationally tractable time. We observe that evolving a population with respect to the spectrogram of the input audio is highly unstable, and present spectrogram traces and audio outputs for our algorithm that elucidate the sensitivity.

Accompanying code and audio files for the figures presented are available at: <https://github.com/dcnbr/ea-orchestra>.

1 PROBLEM DESCRIPTION

Music is a deeply fascinating and uniquely complex subject area in the human arts, with a large number of different aspects of it being rich with potential for exploration and experimentation in computer science. The motivation for creating this algorithm is to use the appeal of music and leverage our own curiosity and combine the creativity of evolutionary computing with the audio result from the algorithm. In summary, we have created an algorithm where it is able to "play" music on its own given existing audio as input, via the application of techniques in evolutionary algorithms.

The problem being solved with creating this algorithm is the question of 'if a genetic algorithm could play music, how would the music sound?'. There are many choices for customizing the algorithm that changes the fitness function that is being presented, which gives bounds but is still a large problem space to explore. As this algorithm is composing music, it has been written with rudimentary understanding of music composition and is instead lead with fourier transformations of musical input, in contrast to other papers solving the same problem which go more in depth with analyzing the music theory beforehand[6]. With this approach, this means that programmers and people who wish to utilize this algorithm in the future are not required to equip the specific domain knowledge of the input audio. Instead, programmers must have knowledge on how the fourier transforms work and interact with the audio input to the algorithm, allowing for a much broader domain to be explored. It also must be considered that the music created can be evaluated subjectively by people, and one future fitness evaluation that can be considered is including human judgement as a fitness function.

2 LITERATURE REVIEW

There are a substantial amount of evolutionary computing algorithm papers that play with music. However, although it is possible to identify and distinguish songs with Fourier transforms[8], there does not seem to be any instances of using evolutionary algorithms in conjunction with composing with representations of the music as fourier transformations. The main articles that will be touched upon in this literature review are those that compose their own forms of music using audio as input with different methodologies.

One of the earliest instances of this can be found in Horner and Goldberg where they applied genetic algorithms to creating a "thematic bridge"[3]. They considered the transition from one musical phase to the next, using a defined

operation set. The individuals within a population would represent the series of transformations from one phase of music to the next. Fitness was evaluated by how good of a fit the final musical transition matched the desired musical transition. This meant that the algorithm was not composing music, but creating patterns based on what was expected.

Another approach to composing music can be found in GPMusic, developed by Johnson and Paul[4] where short musical sequences would be evaluated by users. They utilized the user ratings by then using the data to train another neural network to be able to rate the music without having the slow down process of human interaction. This idea was then extended by Tood and Wener[9] where both the music and critic would coevolve. This system was inspired by birds, where they evolved males (the musical melodies) and females (judgers of the melodies). The females would choose their male partners based off of whichever melodies they liked the best. The male and females were neural networks that would evolve over time, getting better at their respective tasks. They would eventually conclude that while there was musical diversity and structure from their approach, the songs produced overall were not "pleasant to listen to". Both of these models were agnostic to the type of music being created, however, Kunimatsu et al.[5] created a specific model where they specifically noted down the blues style cadences of the chord progressions, creating a tree structure representation where the notes were nodes and the duration of the notes were the level the node was at.

In conclusion, there is a long history of people with very different approaches to compose new music using evolutionary computing techniques. What sets our approach apart is that there is less of a need for in depth knowledge on the domain of music, therefore allowing programmers to be more free in their choices of what music they would like to modify and see their effects on what the resulting algorithm sounds like.

3 EA DESIGN

In order to evolve an audio signal, we need a genotypic representation of sound. Traditionally, digital audio files store sound waves as a fine-grained sampling of the wave over time. This is convenient for playback devices, as the waveform described by the samples directly maps to the actuation of a playback device. However, for modern sound signals of average quality, the number of samples per second is usually 41,000 or 48,000, chosen to be able to represent the full frequency range that is perceivable by the human ear. If we were to model a sound's genotype as a sequence of samples, this model would be intractably large for even very short audio files. Further, it is unclear how mutation and crossover could be defined in such a high-dimensional space in a way that preserves the underlying sound. In order to achieve a relevant and computationally feasible genotypic representation, we turn to the frequency-domain representation of a signal via the fourier transform.

3.1 Fourier Series

The Fourier series [1] of a periodic function is an equivalent expression of that function in terms of an infinite sum of basis functions. It is a related idea to expressing points in space as a linear combination of basis vectors, only for continuous periodic functions, which are infinite-dimensional.

The fourier expansion of a signal $s(t)$ that varies with time t is a linear combination of sinusoidal and co-sinusoidal signals of increasing frequency. Each term in the series expansion is a pair of two fourier coefficients A_n and B_n per basis function n that weight each basis function according to how 'important' it is to reconstructing the original signal $s(t)$. Larger values of these coefficients for a particular basis function n indicate the higher-presence of that basis function's frequency in the input signal. Taken in aggregate, the full range of fourier coefficients are considered the *frequency-domain* representation of a periodic signal, dual to *temporal-domain* (waveform) representation.

Transforming a signal into the frequency domain is done via the *fourier transform*. The fourier transform systematically computes the coefficients for each basis signal f by projecting the input signal onto each basis:

$$A_n = \frac{1}{P} \int_{-P/2}^{P/2} s(t) \cdot \cos\left(\frac{2\pi}{P} \cdot nt\right) dt \quad B_n = \frac{1}{P} \int_{-P/2}^{P/2} s(t) \cdot \sin\left(\frac{2\pi}{P} \cdot nt\right) dt$$

These equations represent the projection of signal $s(t)$ onto sin and cosin waves that make n cycles across fundamental period P . As such, each A_n and B_n correspond to the projection of the signal onto a basis wave of frequency $\phi = \frac{n}{P}$. Often, coefficients are instead computed via a complex exponential basis, which Euler's formula [2] shows is an equivalent representation.

$$C_n = \frac{1}{P} \int_{-P/2}^{P/2} s(t) \cdot \exp\left(-\frac{2\pi}{P} \cdot ft\right) dt$$

In this coordinate space, each complex coefficient is $C_n = \cos(s(t)) + i \cdot \sin(s(t))$. A different interpretation of coefficient C_n arises; the amplitude $\alpha_n = |C_n|$ represents the amplitude of a cosine basis wave, and the angle $\omega_n = \angle C_n$ is the periodic *phase*, or horizontal shift of the signal along the time axis.

3.2 Short-Time Fourier Transforms

Audio recordings, while certainly signals, are not periodic in the above sense. In audio processing, it is common to overcome this problem via the short-time fourier transform (STFT). In essence, the STFT is a sequence of instantaneous fourier transforms at each point t in the audio signal $s(t)$. Computationally, the input signal is divided into small 'frames', and each frame considered to be its own periodic signal. For sufficiently short frames, the short-time fourier transform and its inverse produce identical signals to the input. In this report, we use slightly overlapping frames of 93ms, which gives good empirical results for general audio files.

3.2.1 Spectrograms. One way of visually representing the STFT is via a *spectrogram*. A spectrogram plots each frame of the STFT in sequence along the x-axis, and each basis frequency along the y-axis. Spectrograms use shade or hue at each (t, ϕ) point in the graph to represent the amplitude α_n of the fourier coefficient for that frequency at that timeframe. Examples of spectrograms are given in Figure 1, for simple audio files. Notice that for audio generated from the basis-frequency 'instrument' in 1a, the spectrogram closely resembles a music roll, which are rolls of instructions used to operate mechanical musical instruments like player pianos. In a sense, a spectrogram of an audio signal can be viewed as the music roll for the fourier-series 'instrument'; said another way, the spectrogram (and by extension, the STFT) is *sheet music for a general-audio music machine*.

This report is primarily concerned with audio itself, but we will use the spectrogram visualization in this report as a stand-in for audio in-text. Readers are encouraged to consult corresponding audio provided with this report for the 'true' materials of discussion.

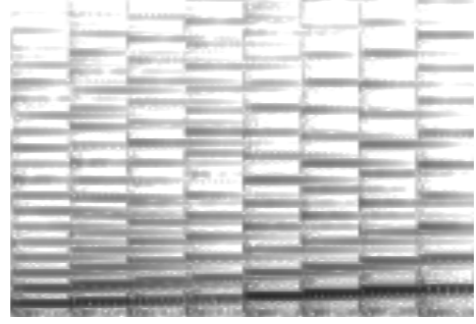
3.3 Genotypic Representation

We model an audio signal as a population of individual signals in the frequency-domain, using a simple 3-tuple:

$$(frequency, amplitude, phase) = (\phi, \alpha, \omega) \begin{cases} 0 \leq \phi \leq 20500 \\ 0 \leq \alpha \leq \infty \\ 0 \leq \omega \leq 2\pi \end{cases}$$



(a) Spectrogram of a sine wave playing one octave. As sine waves are the basis waves onto which the audio signal is projected, we see only one frequency component at any time, corresponding to the frequency of the note being played.



(b) Spectrogram of a piano playing one octave. The lowest (fundamental) frequency of each note played match the shape of the sine wave spectrogram, but many other frequencies are present above. The (higher) harmonic frequencies give the piano its particular sound.

Fig. 1. Spectrograms of different instruments playing the notes of one octave in the CMaj scale in an upwards run. Time increases along the x-axis, frequencies increase up the y-axis. ‘Louder’ amplitudes are shaded darker.

In audio signal processing for human listening, an upper frequency bound of 20500 is commonplace, as this is generally the limit of the human hearing range [7]. Amplitude can grow arbitrarily large, and phase is rotationally symmetric around 2π degrees.

3.4 Evolutionary Algorithm

At the outset of our project, our intention was to allow for arbitrary choices in evolution strategies so that interesting acoustic differences might be observed. We have since learned that this much freedom is numerically unstable for general audio input, which we discuss further in Section 4. For the purposes of this report, we present a single evolutionary algorithm with static choices of strategies, but present comparative results by varying parameters within this algorithm. Comparative results are presented in 5.

Our evolutionary algorithm follows the usual structure. In pseudocode, our meta-process is as follows.

- 1: Let \mathcal{A} be the input audio signal
- 2: Let \mathcal{P}_k be the population of individuals \mathcal{X}_i for generation k
- 3: $\mathcal{F} \leftarrow \text{stft}(\mathcal{A})$
- 4: Initialize \mathcal{P}_0
- 5: $k \leftarrow 1$
- 6: **while** $k < m$ **do**
- 7: Compute the fitness of every $\mathcal{X}_i \in \mathcal{P}_k$ with respect to fourier frame \mathcal{F}_k
- 8: $\mathcal{P}_{k+1} \leftarrow \text{evolve}(\mathcal{P}_k)$
- 9: $k \leftarrow k + 1$
- 10: **end while**
- 11: $\mathcal{O} \leftarrow \text{stft}^{-1}(\mathcal{P})$

We discuss the particulars of evolution strategies in turn.

3.4.1 Initialization. We chose a population size that mirrors the total numbers of fourier coefficients used in the frequency-domain representation of the input signal. This is to keep the relative density of frequency information in the evolutionary population similar to the input. For this report, frequency-domain representations are computed with 1025 coefficients (a power of 2 plus 1, chosen because of the computational benefits these give to fourier transform algorithms), and so our population is a static 1025 individuals in size.

Individuals are initialized within the following bounds:

$$\mathcal{X}_i = ([0, 22025], [0..1], [0..1])$$

3.4.2 Fitness. The fitness of the every individual is recomputed every generation with respect to the next ‘frame’ k in the STFT of the input audio.

Let $\mathcal{X}_i = (\phi_i^X, \alpha_i^X, \omega_i^X)$ be the i -th individual in our population and $\mathcal{S}_j = (j, \alpha_j^S, \omega_j^S)$ be the fourier coefficient of basis frequency j . We compute the fitness of individual \mathcal{X}_i as the amplitude of that individual’s nearest frequency neighbour \mathcal{S}_g :

$$\text{Fitness}(\mathcal{X}_i) = \alpha_g^S \quad \text{such that} \quad g = \arg \min_j |j - \phi_i^X|$$

3.4.3 Parent Selection. We select parents from a uniform probability distribution. The mating pool formed with 50% of the full population. Parents are paired randomly to form parent groups.

3.4.4 Crossover. We use bounding-box crossover, in which offspring are chosen uniformly-randomly in a bounding box around two parents. As a default, we choose a bounding box that is twice the size of the minimal-bounding-box around the parents, though we vary the ratio in Section 5. We use \mathbf{H}_b to represent this hyper-parameter, so, $\mathbf{H}_b = 2$ by default.

3.4.5 Mutation. For an individual $\mathcal{X}_i = (\phi_i^X, \alpha_i^X, \omega_i^X)$, we mutate amplitude and frequency by a uniformly-random amount up or down. Amplitude is always mutated up or down by 1db, but we use the hyper-parameter \mathbf{H}_m to represent the range of the mutation to frequency. By default, $\mathbf{H}_m = 100$ Hertz, but we vary this range. Phase is not mutated, as phase is generally not an audible characteristic to humans.

$$\mathcal{M}_i = (\phi_i^X \pm \mathbf{H}_m, \alpha_i^X \pm 1, \omega_i^X)$$

3.4.6 Survivor Selection. We form the next generation from the fittest 1025 individuals from the combined list of parents and offspring.

4 EA RESULTS

Through the implementation of our evolutionary algorithm design, and our use of Fourier transform to represent our population’s evolution in the form of audio signal output, we were able to execute our algorithm using a variety of different configurations. By testing our algorithm with different evolutionary techniques and parameters for our implementations of mutation, parent selection, recombination, and so forth, we sought after the best performance in our results. Ultimately, we discovered that our problem space was more challenging to explore than we initially foresaw, and the resulting audio output of our evolutionary algorithm was subject to some limitation, such as interference from background noise in the input audio file as well as premature convergence. In this section, we will describe the results

of our best configuration of evolutionary techniques, some of the major challenges we faced in dissecting our problem space, and how they contributed to our results.

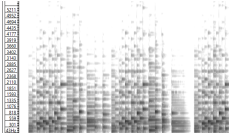
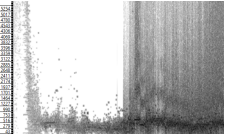
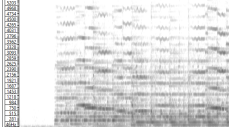
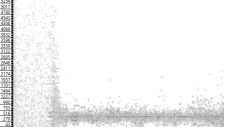
	Original Audio	Evolutionary Algorithm Output
Beethoven "Bagatelle" Op. 119 No. 9		
Vivaldi "Winter" Op. 8 No. 4		

Table 1. Spectrograms of the first 10 seconds of two classical piano pieces and the output of the evolutionary algorithm. Initial convergence to the fundamental 'tone' of the piece is fast, but the algorithm is unable escape this tone afterwards. In the Beethoven piece, the later spectrographic complexity is acoustic distortion from the output audio becoming too loud.

4.1 Digital Noise

One significant obstacle we faced after executing our first few tests of our evolutionary algorithm implementation was the issue of interference from background noise in the input audio file. Several of our early experiments resulted in audio output that mostly consisted of seemingly random, jumbled, electronic-sounding notes that did not seem to resemble any particular melody or rhythm, or follow a particular pattern that was reminiscent of the melody of the input audio file. We believe that the underlying audio signal of these results may instead have been representative of the sound of background digital audio noise itself. When analyzing the interactions between our algorithm and the input audio file given to it, we know that humans listening to a song will only hear and process the notes in the audio that are significant to the human ear, however the underlying audio can contain signals of many different amplitudes, some of which might be extremely quiet and not audible by humans. Our initial approach to implementing the fitness function for our evolutionary algorithm was to match an individual to its nearest neighbour in the input audio spectrogram, and compute its "distance" from that nearest neighbour in terms of frequency and amplitude to represent the individual's fitness. We discovered that with this approach, it would be possible for an individual's nearest neighbour to be identified as a background noise note with very low amplitude that would be insignificant to a human's experience of listening to the audio. This would result in our fitness function not appropriately calculating an individual's fitness as a measure of how close the individual is to a note in input audio that is actually musically significant, and instead would lead to the presence of random digital noise overwhelming the population.

4.2 Convergence to a Single Tone

Another obstacle that we faced during the development of our algorithm was pre-mature convergence to a single tone. We found that in our early runs of the evolutionary algorithm, the playback of the output audio file would begin by resembling digital audio background noise, but then quickly start to converge towards becoming a single digital audio signal with a consistent volume and frequency, within a few seconds. We identified two probable causes for our

algorithm's convergence into a single tone in our implementation, one being our implementation of recombination and the other being how parents in our population were being paired to create offspring. In our first implementation of recombination, we used a version of whole arithmetic crossover to create a approach that would generate offspring that would represent the average of its two parents in terms of frequency and amplitude. This solution proved to be problematic in the sense that it was subject to the bounding-box problem in evolutionary algorithms, where because the crossover mechanism always generates an offspring that is "between" its two parents, the population will grow closer and closer together over each generation. When it came to pairing individuals from our pool of parents to produce offspring, we also discovered that the way the parents were paired might've had an influence on our algorithm's performance. Our initial approach was to simply pair the parents up randomly after performing parent selection, but we learned that with this solution, it would be possible for a parent with a very high frequency/amplitude to be paired up with a parent with a very low frequency/amplitude, and using crossover to produce an offspring of the two parents might simply lead to the meaningful attributes of both parents being trivialized as the offspring becomes the middle of their two extremes. To try and solve this, we attempted to implement an intelligent parent-pairing method that sorted the parent pool by frequency or by amplitude, and paired parents with similar values together, but this didn't seem to have a significant impact on the algorithm's convergence. Overall, after running into these obstacles concerning crossover in our algorithm, we have also theorized that the idea of applying crossover to our population may be problematic in and of itself, as there is no guarantee that interpolating two parents into an offspring will create a meaningful mix of the best characteristic of the two parents.

5 COMPARISON

To elucidate the properties discussed in 4, we investigate our algorithm on simpler input audio signals. We present the corresponding spectrograms of these experiments in Table 2. The input audio in this investigation is limited to only one or two input frequencies, and with slow variations to frequency.

In this relatively straightforward task, we observe the algorithm is highly efficient to converge from a random initialization to the signal that it tracks, and preforms adequately in adapting to changes in a single frequency. However, the algorithm struggles to differentiate between multiple input frequencies, especially when they are relatively close together. This behaviour persists for other relatively small perturbations in two arbitrarily-chosen hyper-parameters, though the overall behaviour and sound of the algorithm is highly sensitive to these small changes. We theorize that the inability of the algorithm to differentiate between two separate signals is due in parts to:

- (1) No encouragement for crossover to occur between two parents tracking the same frequency
- (2) A mutation intensity that overlaps the distance between the two frequencies

In theory, these challenges could be overcome with extensive tuning of the relevant hyper-parameters, but a need to tune to that degree is a limitation of the methodology for which we do know a solution. Future work could investigate intelligent on-the-fly adjustment of mutation intensity or parent pairing methodology to alleviate instability and allow different mutations strategies to produce meaningful sound.

6 CONCLUSION

Ultimately, while we were not able to fine-tune the implementation our evolutionary algorithm to the point where our program could very recognizably learn to play an existing input song, we were able to explore our problem space and gain many insights on the interactions between algorithmic problems solving and music. Over the course of our project's

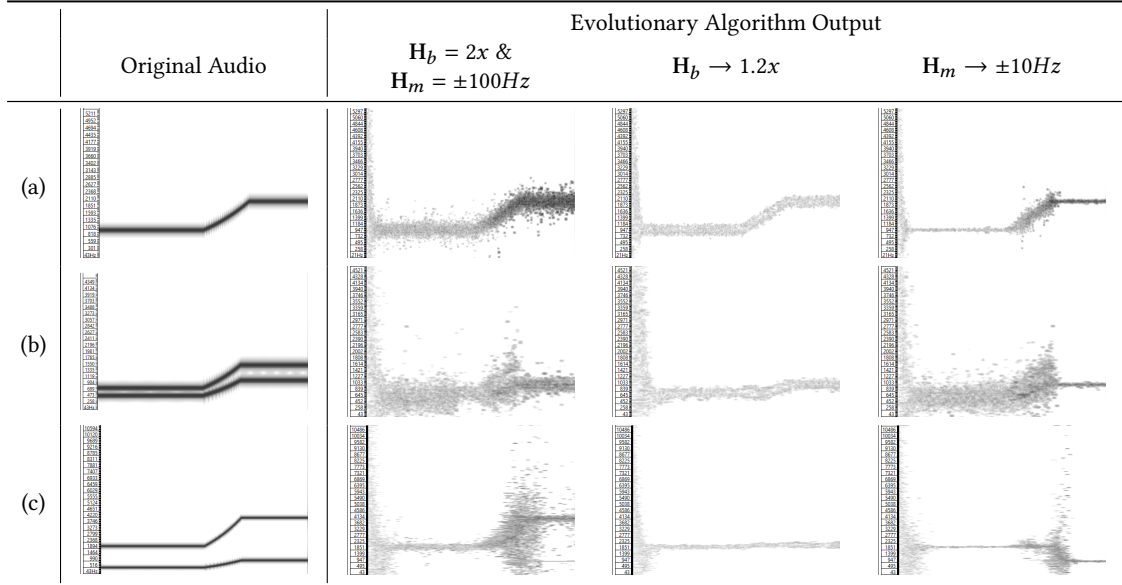


Table 2. Spectrograms for three simple audio files, and the results of our evolutionary algorithm for arbitrary configurations of hyperparameters, which demonstrate the instability of the algorithm in this domain. H_b represents the hyperparameter controlling the size of the bounding box in crossover, and H_m represents the hyperparameter controlling the intensity of mutation. For the single tone (a), the evolutionary algorithm tracks the audio signal well. However, the algorithm struggles to capture any additional complexity in (b) or (c) in their entirety.

research and experimentation, we were able to explore the background and pre-existing literature of our problem space, combine the application of Fourier transform with our evolutionary algorithm to give life to our results, and discuss the different obstacles we faced and insights we gained through the iteration of the design of our evolutionary techniques. We theorize that the underlying audio signal of music may simply be too chaotic for an evolutionary algorithm to effectively operate on due to the overwhelming presence of noise and audio data not relevant to what humans would perceive as the main significant sounds of a song. In the future, our work could be expanded upon through the use of hyperparameter tuning to attempt to find the correct parameters and configurations that may be precise enough to develop an algorithm that could overcome these obstacles.

REFERENCES

- [1] Ronald Newbold Bracewell. 1986. *The Fourier transform and its applications*. Vol. 31999. McGraw-Hill New York.
- [2] Leonhard Euler. 1748. *Introductio in analysin infinitorum*. Vol. 2. Apud Marcum-Michaellem Bousquet & Socios.
- [3] Andrew Horner and David E Goldberg. 1991. *Genetic algorithms and computer-assisted music composition*. Vol. 51. Ann Arbor, MI: Michigan Publishing, University of Michigan Library.
- [4] Brad Johanson and Riccardo Poli. 1998. *GP-music: An interactive genetic programming system for music generation with automated fitness raters*. University of Birmingham, Cognitive Science Research Centre.
- [5] Kanae Kunimatsu, Yu Ishikawa, Masami Takata, and Kazuki Joe. 2015. A music composition model with genetic programming-a case study of chord progression and bassline. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*. The Steering Committee of The World Congress in Computer Science, Computer ..., 256.
- [6] Roisin Loughran and Michael O'Neill. 2020. Evolutionary music: applying evolutionary computation to the art of creating music. *Genetic Programming and Evolvable Machines* 21 (2020), 55–85.

- [7] Dale Purves, G Augustine, David Fitzpatrick, L Katz, A LaMantia, J McNamara, and S Williams. 2001. *Neuroscience 2nd edition*. Sunderland (MA): Sinauer Associates, Chapter 13: The Auditory System. <https://www.ncbi.nlm.nih.gov/books/NBK10924/>
- [8] Prem Seetharaman and Zafar Rafii. 2017. Cover song identification with 2d fourier transform sequences. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 616–620.
- [9] Peter M Todd and Gregory M Werner. 1999. Frankensteinian methods for evolutionary music composition. *Musical networks: Parallel distributed perception and performance* 3, 4 (1999), 7.