Music Reconstruction Using Genetic Algorithms

Arzoo Jiwani*, Justin Feldman *,Venkata Krishna Rayalu Garapati *,
Annamayya Vennelakanti *, Vishwajeet Hogale *, Jason Zou *
*Khoury College of Computer Sciences, Northeastern University, Boston, MA

Abstract—**This study proposes a genetic algorithm framework for music reconstruction that prioritizes musical coherence across instrumental sections from multiple unassociated songs. The system reads MIDI files to extract relevant musical elements, then uses genetic operations to introduce variations and transitions to improvise or randomize the existing music data to create newer versions. The fitness function assesses a generated song's quantitative musical qualities to objectively rank the best tunes. The algorithm outputs the best songs at the end of the evolutionary process as MIDI files with intact musical coherence.**

Index Terms—**Genetic Algorithms, MIDI Processing, Music Generation, Musical Coherence.**

I. Introduction

Genetic algorithms are a robust genre of algorithms that are inspired by the theories of natural selection and evolution. Given a population of individuals represented as a structured data set, the genetic algorithm architecture creates an artificial natural selection environment in which the strongest individuals survive based on the strength of their "genetics". These surviving individuals remain in the population to produce offspring that carry combinations of the parents' genetics along with novel genetic mutations as they occur naturally in living beings. These offspring potentially possess a stronger genetic composition, improving the population's traits overall as the evolutionary process continues.

Genetic algorithms' strength lies in their capability to analyze the performance of a vast diversity of data manipulations without the use of complex mathematics that other artificial intelligence algorithms rely on. During the evolutionary process, complex data structures or data activations are refined to produce results that trend towards the creator's desired outcomes. Given the biologically inspired elements of genetic algorithms, we were naturally inclined to explore the application of a similarly humanistic and powerful process, musical composition.

Music is timeless and universalized. It transcends geological location, timelines, language, and has a direct impact on human emotions. However, it is as deeply quantitative as it is creative. This dichotomy is what draws us to musical composition as an application for genetic algorithms. At its roots, music is just a combination of sound. Sound is just a product of quantifiable physical vibrations. The field of music theory emerges from these quantitative connections and bridges the gap between music's subjectivity and objectivity. Music theory is what gives us the opportunity to computerize the musical composition process.

While the likes of music generation and artificial intelligence have already converged, the existing methods revolve around deep machine learning models that train on vast networks of existing songs. Our study aims to explore the functionality of genetic algorithms to identify the aligning patterns of smaller subsets of existing songs, and express their overlapping features through the recombination of new musical compositions.

II. Related Works

Genetic algorithms for music production have been examined in terms of decomposing and recombining musical elements including melody, rhythm, and harmony. These methods are based on evolutionary notions similar to those seen in natural evolution, with genetic fitness functions directing compositions. Several studies propose automating music production by integrating human evaluation, digital signal processing, and MIDI-based recombination.

Farzaneh and Toroghi (2021) [1] created a melody generating system with an interactive evolutionary algorithm. Human evaluators grade the generated music. This approach combines genetic algorithms and deep learning using a Bi-LSTM network, demonstrating how iterative feedback can improve musical compositions. This experiment highlights the possibilities of interactive evaluation in music generation.

Matić (2010) [2] created a genetic system for music composition that uses position-based rhythm representation and relative pitch encoding. This technique allows for varied and robust composition, preventing premature convergence with updated genetic operators. The algorithm's adaptability enables the creation of a variety of organized musical outputs.

In Akanksha Satpute et al (2023) [3],the study proposes using GA to compose music and using the fitness function to select more melodic songs. In GA, two musical segments will act as parent nodes for developing new music, and by using genetic operators, the music will change so that the breaks between the tunes are updated. Music that sounds pleasing is chosen with the user's assistance using the fitness function, and if the user is satisfied with the generated tune, the process of generating the music is terminated; otherwise, the selected musical tune by the fitness function will serve as the parent node for the next generation of musical tune. Furthermore, this paper shows which fitness function should be applied to the given problem

In Majumder and Smith (2021) [4], authors presented their genetic algorithm for music recombination by extracting features from different MIDI files. Their method recombines rhythm, melody, and harmony to produce new compositions, demonstrating how genetic algorithms can be used to blend musical components from different sources.

We acknowledge that various research works inspired our work and gave useful insights into the methodology and concepts underlying our investigation. However, it is vital to highlight that we did not use or replicate any of the content, data, or findings from these sites. Instead, we built our technique independently, drawing on our knowledge of the current literature to add new ideas and insights to the subject. This ensures that our work stands out while being guided by the fundamental studies in this field.

III. Problem Statement

Music composition is an inherently creative process that is full of subjective choices of rhythm, harmony and emotions. We wanted to take this process and break it down into a quantitative framework and apply genetic algorithms to recombine pre-existing songs to generate new ones. By representing musical elements such as tempo, velocity, time signature and keys as data derived from MIDI files, we aimed to build a system that uses genetic algorithms to analyze, understand and manipulate these components. This would allow us to create new songs that are a blend of creativity and computational logic.

IV. Methods

A. Genetic Algorithm

The genetic algorithm is made of five main categories: initialization, fitness calculation, selection, crossover and mutation. The initialization stage generates random population from the pre-processed dataset. This initial population is passed through the fitness calculation to generate a baseline population fitness distribution. In the selection stage, best fit individuals are selected and passed to the crossover and mutation stages. During the mutation and crossover stages, individuals are combined to create unique offspring comprised of their parent's genetic features. Once the new generation is initialized, it returns to the fitness calculation stage. This cycle continues until the specified generation limit when the optimal solutions are identified.

B. Preprocessing

1) MIDI File Exploration: A MIDI file, or Musical Instrument Digital Interface file, is a set of instructions for musical playback – like tempo, time signatures, notes, velocity, track names and other information. Unlike audio files, MIDI files do not store sound directly but instead encode how a track should be played. Since it does not contain the sound but only the information on how the track is played, it sounds different on different devices or software used.

MIDI files are ideal for our application as with the information they extract, they enable us to manipulate the musical elements like tempo, velocity, time signatures, keys and track IDs. This structured format enables easy recombination and transformation of the musical elements, thus making it a good fit for the genetic algorithm.

Also, MIDI files are compatible with digital audio workstations like Garageband and Audacity, which ensures easy application.

2) Dataset Construction: Given our input MIDI songs, we proceeded to break each of them into three unique 20-measure sections at the beginning, the middle, and the end of the song. This ensured that we were sampling a diversity of musical features within the individual song itself. Each of these sections were split further into their individual musical instruments. These individual instrumental tracks resembled the foundation of our genetic pool.

Our gene pool is organized through a binning process that put relatively similar instruments into the same bin. For instance, an electric guitar and an acoustic guitar would both be binned as a guitar. We organized the genes into five instrument bins that are commonly used in four-piece bands: drums, guitar, piano, bass, and other instruments. Each of these genes contains data on the notes that are played within them. We chose to extract singular features such as note pitch and note velocity. We also extracted combinatory features such as track tempo, time signature, and the track's identification. Through this binning structure and the quantifiable features that MIDI files allow us to extract from every gene, we created the data structure that the rest of the program's functions aligned to.

V. Architecture

A. First Generation

To conduct genetic algorithms, an initial random population is required. At this stage, we are unable to deduce what combinations of genes would be most highly rated by the fitness function, so it is imperative that we compose individuals made of random genes from across our available dataset.

In the preprocessing stages, we extracted the instrumental tracks of each input MIDI file and placed them into their respective instrument bins. By selecting a random track from each of the five bins and combining them, we can form one new individual. Hence, an individual is formed by five randomly selected music tracks. Repeating this process allows us to populate our first generation with a group of randomly generate individuals, which is the prerequisite for generating further populations.

B. Fitness Function

1) Velocity Scoring: Velocity scoring is used to maintain a consistent level of loudness among all instruments in a song. It involves determining the average velocity of all tracks within a parent and comparing each track to that average velocity.

They are scored based on their individual velocity in comparison to the average velocity within the pair; those closest to the average velocity receive a higher score with a maximum of 20, while those who deviate receive a lower score with a minimum of 1. The overall score has already been adjusted to 100 because a parent can only have a maximum of five tracks, and each track can only receive a maximum score of 20.

2) Tempo Scoring: Tempo scoring is used to ensure rhythmic consistency across all tracks in the parent. It is the rate at which an instrument is performed; hence, songs with tracks that play at the same rate sound better. We compute the average tempo over all five tracks of a parent and set three target tempos: half the mean, the same as the mean, and double the mean. For each instrument, the absolute difference between its tempo and the nearest target tempo is calculated and subtracted from 100 to get a score, with perfect alignment to any target yielding 100 points and bigger deviations yielding lower scores.

The final tempo score is normalized to a 0-100 scale by dividing the sum of all scores by the maximum possible score (5 tracks × 100 points) and multiplying by 100, which ensures that the work is rhythmically coherent while allowing for musically compatible tempo variations.

3) Time signature scoring: Time signature scoring is used to ensure metric consistency between tracks within a parent. It refers to a pattern in which each track is played. The score is determined by the highest number of tracks sharing the same time signature. If all five tracks share the same time signature, we give 100 points. For four matching tracks, we give 80 points. For three matching tracks, 60 points. For two matching tracks, 40 points. If no tracks match or only one track matches others, we give 20 points.

4) Key compatibility: Key compatibility scoring employs the Camelot Wheel technique, which was created by Mark Davis in the 1990s to assist DJs in harmonically mixing music. In this approach, traditional musical keys are translated to a number-letter format (e.g., ”C major” becomes ”8B”), where numbers (1-12) signify wheel positions and letters indicate whether the key is major (B) or minor (A). The score uses three rules to determine the compatibility of each track's keys within a parent.

- Keys are compatible if they share the same letter and adjacent numbers or are the same number.
- Keys are compatible if they share the same number but distinct letters.
- A key is always compatible with itself. For each compatible pair found, we add 1 to our compatibility score.
  The final key score is calculated by dividing the total compatibility score by 20 (maximum possible score) and then multiplying by 100 to get a percentage. This ensures compositions maintain harmonic relationships while allowing creative variations.

5) Recombination Scoring: Recombination scores ensure that no tracks within a parent come from the same song. The function searches for unique track IDs connected with each track and scores such that if any of the track IDs match, a penalty of -400 is imposed, and if none match, a neutral score of 0 is assigned. The inclusion of a penalty score ensures that the genetic algorithm handles tracks from different tunes, lowering the number of redundant tracks.

C. Combined fitness calculation

Thus, the total of these five scoring components—velocity, tempo, time-signature, compatibility key, and recombination is the final fitness score.

Since there are four scoring components with value and one with neutral scoring, all of which are scaled to a maximum of 100, the highest theoretically attainable score is 400. Songs with a higher score conform to the fitness function's objective definition of a good song.

D. Mutation

The mutation function is an important cog in the workings of a genetic algorithm, as it helps introduce randomness and variation in the population. It enables the algorithm to explore a wider range of solutions by avoiding getting stuck in local minima. In our project, the mutation function manipulated musical features such as keys, tempo and velocity. In a mutated individual, we randomly chose a gene and randomly altered one of its features. This introduced variations in the song that deviated from the parent songs. For example, the mutation function might change the tempo of a song to speed up or slow it down or adjust the velocity to alter the intensity of the songs or transpose it to a different key altogether.

E. Crossover

The crossover function combines two parents' genetic information to produce children. The crossover function iterates across the complete population to execute a crossover between two parents that makes two children. The function finds a random crossover point based on the length of the parent such that at least one gene has swapped out with the other parent, making two offspring. We use an ordered crossover method, starting with the first two parents and moving on to the next set of parents. If the population is odd, we crossover the first child with the odd one out.

F. Next generation construction

While creating the successive generations, we make sure that optimization, diversity, and retention of best-fit individuals is prioritized. Each new generation is comprised of four discrete populations: two of them are taken from the mutation and crossover functions, a third one is a subset of the best-fit individuals from the previous generation, and the fourth is a newly generated randomized population. This approach helps us to retain the features that maximize the fitness of our newer population while also with the goal of introducing novel combinations that can benefit the population's fitness overall.

VI. Results and Discussion

Music quality is subjective and therefore, no objective “ear test” of the algorithm's output songs exists to provide data driven insights. However, given our algorithm's objective definition of a quality song, we were able to identify trends in the distributions of a population's fitness scores and validate the intended performance of our genetic algorithm architecture. Users can alter two inputs within our program, the number of generations, and the composition of the populations (see Next Generation Construction). Figure 1 and Figure 2 highlight the population distribution differences with generation count as the independent variable and population composition as the control. Figure 3 compared to either of the other two figures highlights the population distribution differences with population composition as the independent variable and generation count as the control.

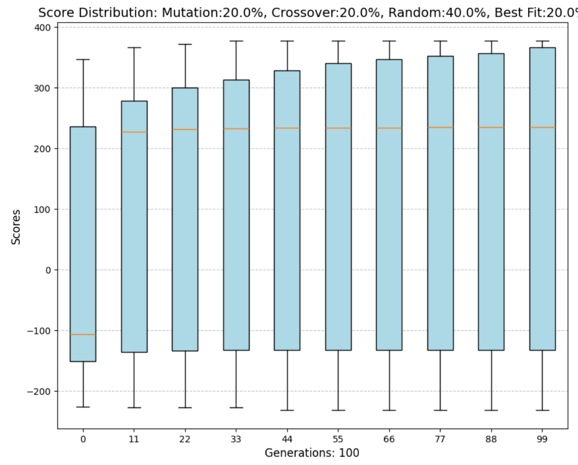

Fig. 1. 100 Generations, Composition 1

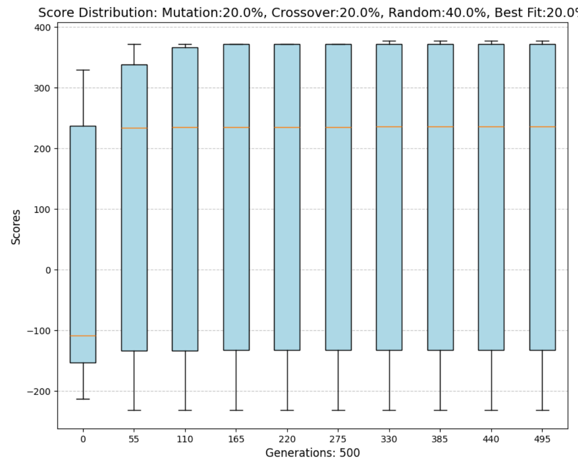

Fig. 2. 500 Generations, Composition 1

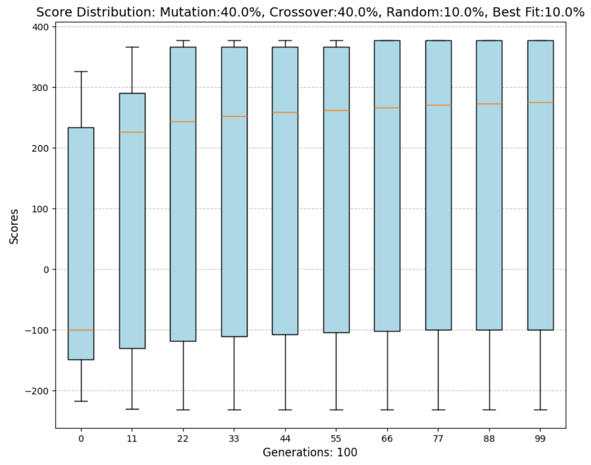

Fig. 3. 100 Generations, Composition 2

A. Discussion

The overall population fitness score distributions resemble our expected trends of a functional genetic algorithm. The 0th generation in each figure (1, 2, 3) is widely distributed with a poor median value, indicating a higher density of individuals in the lower quartile. Recall, the 0th generation is a strictly randomized generation. Sensibly, the random population would present a wide range of fitness scores and poor median fitness since the random function has no intrinsic instruction to favor generating highly rated individuals.

After the 0th generation, the most notable behavior is the immediate improvement of the population's median fitness value and the continued improvement of the upper quartile. The improving upper quartile indicates that the introduction and continued inclusion of the mutation, crossover, and best fit logic to the population is properly incentivizing fitness score growth towards higher scores. The median score's trajectory appears to flatten after the second sample despite the continued improvement of the upper quartile indicating that the upper quartile individuals continue to improve while the quantity of upper quartile and lower quartile individuals remain consistent.

In Figure 1, the 100-generation plot's upper quartile continues to improve towards the best fit song value. In Figure 2, the 500-generation plot's upper quartile reaches the best fit song values of the population around 165 generations and does not improve much further. It appears that after 165 generations, given the features of the input files, randomly generated individuals and crossed over children are unable to provide significant improvements past the plateaued fitness score. Only the introduction of novel feature values through mutations could provide further improvements. The mutation function's logic is randomized resulting in minimal and unpredictable best fit song improvements.

Figure 3 best highlights how the plateauing behavior differs between differing population compositions.Figure 3's upper quartile plateaus by the 66th generation compared to Figure 2's upper quartile that plateaus in the 165th generation. Relative to Figure 1 and Figure 2, Figure 3 uses a higher percentage of mutated and crossed-over individuals with a lower percentage of best fit and randomized individuals in each generation. We can infer that the higher mutation and crossover rates contribute to a faster convergence to the algorithm's performance limits. Additionally, since the best fit individual value does not significantly differ between composition types, we can infer that there is an empirical maximum fitness value bounded by the features of the input songs themselves along with our fitness function's logic.

VII. Conclusion and Future Works

The positive results of our algorithm raise multiple topics of interest for further exploration. One particular area is the development of the fitness function. While our fitness function manages to define the criteria for a proper euphonious song, it does not capture the essence of popular song genres, such as pop and rock and roll. It may be possible to alter the fitness function so that it can capture the essence of a genre of music and reflect that in the genes of our population. For example, finding the approximate range of velocities, pitches, and keys most employed by the rock and roll genre and implementing those into our fitness function can lead to the mimicry of rock and roll music in our final population.

Limited by our amateur music theory backgrounds, we were only able to use rudimentary musical features to objectively define the fitness of our generated songs. Combinations of the musical features we expressed result in more advanced topics such as phrasing, rhythm, meter, and harmony. If we added these advanced features to our fitness function, we could conceivably create more complex and musically appealing outputs.

Currently, our algorithm successfully produces songs with harmonious qualities by filtering a population of songs through a predetermined fitness function. This provides us with an elementary proof of concept that genetic algorithms and mathematics can produce cohesive songs, as well as multiple avenues for future development.

VIII. Additional Deliverables

Please find the link to our repository: https://github.com/vishwajeet-hogale/Music-Reconstruction

A. Contributions Paragraph

All of us collaborated weekly to define intermediate deliverables and the technical structure of our code. Everyone was delegated coding responsibility. While some functions may have been more involved than others, we made sure that everyone had equal opportunity to contribute. Vishwa was essential in the integration of our various functions. All of us equally contributed to the report and presentation. Overall, there was respect for each other's time and efforts.

References

[1] M. Farzaneh and R. Mahdian Toroghi, *Music Generation Using an Interactive Evolutionary Algorithm*. Springer International Publishing, Dec 2019, pp. 207–217. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-37548-5_16

[2] D. Matić, "A genetic algorithm for composing music," *Yugoslav Journal of Operations Research*, vol. 20, no. 1, pp. 157–177, 2010.

[3] A. Satpute, M. Bajbalkar, M. Velankar, S. Gurav, and P. Abnave, "Soft computing for music generation using genetic algorithm," *Journal of Soft Computing Paradigm*, vol. 5, pp. 11–21, 03 2023.

[4] S. Majumder and B. D. Smith, "Music recombination using a genetic algorithm." ICMA, 2018.