EvoMedley: Interactive Melody Generator with Multi-song Fusion

Shouyue Hu, X, X

Writing Sample 2021-2022

Abstract. People often express their individuality through unique musical preferences. As such, we present EvoMedley, an interactive music mixing method that enables users to create personalized remixes from their playlists. By extracting and combining indivisible melodic units into a "gene pool," EvoMedley generates remixes using an evolutionary algorithm (EA)-based framework. This framework dynamically adapts to user preferences through a real-time fitness function, emphasizing desired features in each iteration.
Our evaluation shows that EvoMedley effectively retains the essence of parent tracks while introducing novel variations guided by user input. This approach highlights how computational creativity can support personal expression in music creation. Music files can be found here.

Keywords: Evolutionary Algorithm · Music Generation · Feature Extraction · Music Representation · Interactive

1 Introduction

In recent years, sharing music on social media has been considered a way of self-expression for contemporary users, and music taste has become a part of the sociological portrait [19]. Studies have shown that 90% of the average user engages in some music-related activities on social media platforms, such as listening to music, watching music videos, and sharing music [10]. For people who post music online, self-expression is considered the most influential personal motivation for willingness to share music on social media [9]. In addition, users may want to show their individuality and artistry by sharing music with their personalities. In 2020, the Yearly Music Profile on NetEase Music went viral by generating users' playlists of the ten most listened to songs in the year.[24] The Annual Music Report was hugely shared online during 2020, primarily by teens who want to show their taste in music and their personalities reflected by the composition of the playlist. However, these mainstream ways of sharing singles and playlists are less representative of users to other ways of self-expression, such as writing, where users can only choose from existing music that might not fully represent the user. Hence, we want to explore the possibility of interactively generating unique music from multiple songs with different genres.

This paper proposes a novel evolutionary music generation method, as shown in Fig.1, that allows users to generate a unique music track that represents the

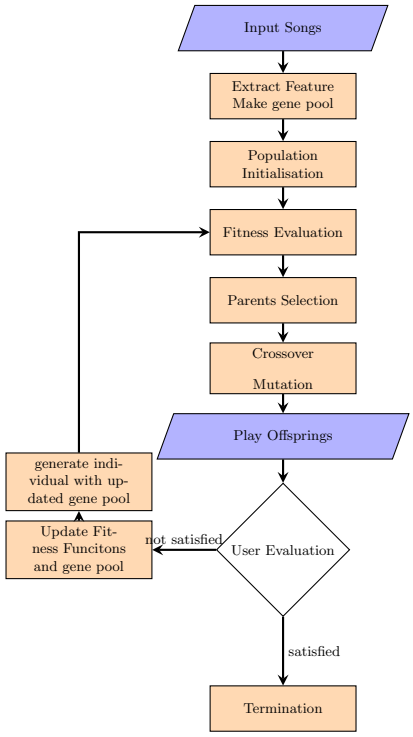


Fig. 1: Workflow of EvoMedley

user by mixing multiple favorite music tracks, where the user's choice provides guidelines on the detail of the generated music. This work aims to introduce an interactive evolutionary approach to music fusing and present a framework for how the user interface is combined with EA. Hence, the focus of the discussion will be on music feature extraction and mixing rather than achieving state-of-the-art level music generation. As a background to the study, in Section 2, we present several broad classes of music generation models and recent research. In Section 3, we present feature extraction algorithms that use multiple songs to form a gene pool, and the EA proposed. In Section 4, we present the experiments on music generation and analyze the effectiveness of our music generation algorithm. In Section 5 we discuss evaluation methods in music and suggest potential future optimizations and directions.

2 Related Work

EvoMedley aims to contribute by 1) allowing users to generate representative digital music based on a self-selected music playlist, 2) allowing the user to evaluate the effect while the algorithm is running to customize it according to the user's sensibilities, 3) taking advantage of editing, 4) extracting features in the

music. EvoMedley is mainly concerned with extracting musical features, generating music, and evolving algorithms. In this section, related work is presented while pointing out the advantages and disadvantages of each.

For the study of music generation, experts and scholars have given results from different perspectives, yet there are still challenges. With the recent development of deep learning models, deep learning models such as generative adversarial networks, transformers, and Diffusion models can be applied to automatically learn music styles from music corpus and generate music of specific styles.[18] Jukebox[3] uses multi-scale VQ-VAE[1,5] to compress the original audio into discrete codes and uses an autoregressive transformer is used to model the music generation. Liu proposed using the generative adversarial networks (GAN) model to generate multi-track music.[11]. Recently, the popular Diffusion model has also been shown to have good performance in music generation. Mittal et al. [15] propose a technique for training diffusion models on symbolic music data by parameterizing the discrete domain in the continuous latent space of a pre-trained variational autoencoder. The model generates music that is excellent in terms of musical quality, coherence, length of audio samples, and adaptability to artists, genres, and lyrics, but there is still a gap between it and human-created music. However, these developed methods cannot be applied to selected music tracks from multiple sources since their music generation is model-based. A major limitation of it is that decoders are seeded with a random initial state, and the user has limited choice and freedom over the generated tracks. Respectively, our approach uses a gene pool of features for individual generation, which in principle allows any number of music tracks to be mixed, simply by parsing them into the gene pool in parallel.

The above music generation models mainly use two ways for evaluation: Heuristics and human evaluation .[12] Heuristics evaluation methods can be rule-based or Similarity-based: Rule-based scoring requires the application of rules containing music theory and musicality-based rules to generate music, which has the advantage of generating music with a greater degree of rule-based tendencies. However, the disadvantage is that music as abstract content is challenging to quantify in terms of rules. Similarity scoring is an approach of music classification, where the representative application is OpenAI's Jukebox [3]. This kind of music generation uses a neural network to generate music with commonality using big data. It can be suitable for generating the same type of music but reply to the classification of good data. For example, if the training dataset contains several styles of music, it will affect the quality of the generated product.

Hence in music mixing, it is not appropriate to generate similarities to neural network music, considering that the user's favourite songs can be multiple from different genres. The main reason is that although it is difficult to rate music using a predefined set of rules when the input is from different genres, the users can mark out the best music out of a list so that the rules and directions of the generated ratings can be standardized according to the user's preferences. Therefore, we propose the use of an evolutionary algorithm to achieve a combination of multiple styles of music.

An Evolutionary Algorithm(EA) is a subset of evolutionary computation[4], inspired by the evolutionary mechanisms. EA has good performance solving complex problems, where the diversity of its population allows it gradually optimize the solution in multiple generations of iterations. [4]Musicianship, as a subjective criterion, is difficult to be judged by a single heuristic. So EA can be used to compose while preserving a diversified candidate pool for a user to select.

There have been multiple works using evolutionary algorithms to generate music[13]. There are several studies on the interactive generation of music by genetic algorithms in the early twentieth century: Johansen et al. proposed an interactive system allowing users to generate short music using interactive GPs

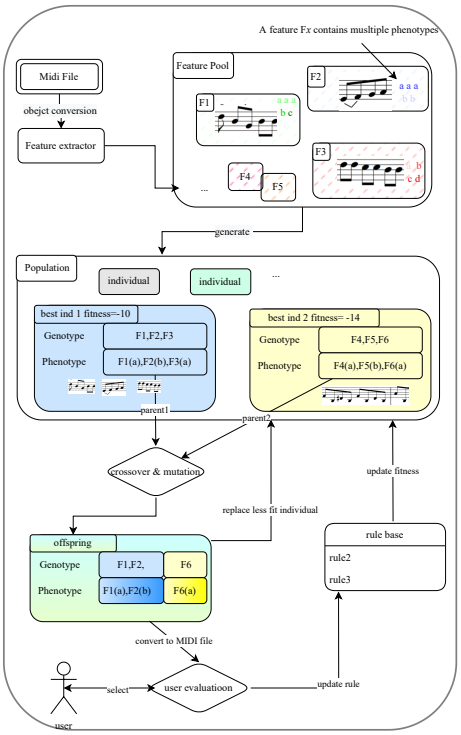


Fig. 2: Schematic representation of EvoMedley workflow 1) the user-selected MIDI file is converted into an operable object. 2) Multiple objects are extracted to form a gene pool. 3) Individuals assembled by random features constitute the population. 4)EA selects parents through tournament selection. 5) The parents are recombined and mutated to obtain the offspring. 6) User evaluates current offspring and gives a winner track; their choice will be used to modify the fitness function by giving more weight to features in the winner track. 7) New fitness functions and reevaluates the population 8) Offspring replace fewer fit individuals in the population. 10) If the user is satisfied with a specific track presented in the evaluation, the program ends and outputs the selected music.

[7]; Moroni et al. proposed Vox Populi, where a population of chords codified according to MIDI(Musical Instrument Digital Interface) protocol evolves through the application of genetic algorithms to maximize a fitness criterion based on physical factors relevant to music.[16]. In recent years, Hofmann has proposed a GP approach to music generation that represents music as a set of constraints over time [6]. Scirea et al.[22], on the other hand, proposes a GP point-based composition framework and allows users to provide feedback through a feedback survey tool. Evocomposer de2020evocomposer uses NSGA-II policy-based multi-objective evolutionary algorithm which is is an elitist non-dominated sorting genetic algorithm to generate polyphonic music. However, generating music via GPs mentioned above still has difficulties, such as Symbolic heuristic evaluation methods do not have enough rules because they are hard-coded from limited knowledge and not able to compose all kinds of music,[12]. Our approach solves this problem by using Heuristics plus user interaction as a scoring method: adding real-time interaction selection as a scoring method based on similarity Heuristics ensures the accuracy of the judging criteria through the subjectivity of users.

3 Proposed Method

This section is divided into four subsections. In the first section, we introduced how the features are extracted as the minimum feature fragments of music to construct the gene pool. Secondly, Evolutionary Algorithm introduces the basic structure of the EA. In the third and forth subsection, we introduce the use of operators in the EA and the design of fitness functions, respectively. We also provide Fig. 2, the flow chart of the algorithm's operation, and Algorithm. 1 shows the EA in pseudo-code.

3.1 Feature Extraction

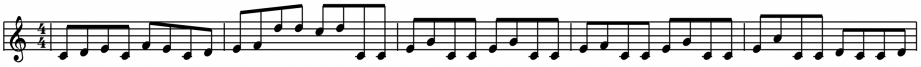
In this sub-section, we propose a method for MIDI(Musical Instrument Digital Interface) music feature extraction. The features obtained in this way will be used for EA initialization, which will be described in Section. 3.2. The feature extraction algorithm aims to analyze and obtain all the regularly occurring minimal indivisible melodic fragments from the MIDI file, which are considered to carry the characteristics of the MIDI file. Salamon at el. describes the existing research and difficulties in extracting melodies from music [21].

Past music studies that used discrete format notes are commonly implemented with the file extension of MIDI, which is a standard protocol for describing music files and connecting electronic instruments and computers. The tones of music in MIDI are represented discretely as a symbolic model of natural audio. For simplicity, all music files used in this article use this MIDI representation. the MIDI format can also be converted back and forth to waveform music[8].Structure-wise, MIDI file will contain multiple instrument tracks, each with multiple notes distributed in time, each with its duration and pitch. Our

conversion algorithm reduces each track to an object, flattens out the sequence of notes it uses, and enters it into a list object to be stored for further use.

Similarity calculation methods commonly used in deep learning are not very applicable to music similarity; whether Euclidean distance, Jaccard, or Pearson correlation coefficient cannot reasonably calculate the similarity of music at both ends. Therefore, in order to count all melody clips that appear in a piece of music, we propose a 3-step algorithm for extracting features for MIDI phrases. The first step is to extract the melody in the music and normalize the input. The process can be repeated in loop for multi-track MIDI. The MIDI music mentioned in this paper is unified in the key of C, and there is no transposition, so the pitch standard is unified; only the notes are converted into numbers, each number represents a note, and the duration value corresponding to each note is recorded. In reality, some music pieces are transposed in mode several times and need to be stored as sub-phrases to record the parent-child relationship. The second step is to extract all the melodic fragments in the melody with lengths between a-b via an iterative process and record the number of times each fragment appears. a-b here represents the ideal length of the smallest rhythm section of a piece of music, such as a guitar/piano riff. And it is possible to use harmony to search for fragments quickly. In the third step, the fragments are divided into buckets by harmonic characteristics, with fragment length $\mathbf{a}-\mathbf{b}$, and minimum occurrence frequency $\mathbf{x}$. The inverted index is established, and weights are given according to the occurrence frequency.

With these three steps, a gene pool consisting of different lengths of harmonic fragments is created, for example:



(a) 12314312349989113511351134113511361121 12

Fig. 3: An example input MIDI track

In Fig. 3, the distance between 1 and 2 is closer than the distance between 1 and 5 due to the distance in sound frequency 1(do) being one tone lower than 2(re) but 4 tone lower than 5(sol). The harmonic combination of 1135 tones occurs several times in this melody, and the tones 1125, 1134, and 11351 are similar to it. Therefore, this fragment can be extracted with the result: [1135,112]. These short phrases are repeated in loop. By putting these fragment features into a gene pool with the features extracted from other MIDI fragments, So we can initialize a population for the evolutionary algorithm by combining a specific number of features as individuals.

Algorithm 1 EvoMedley Algorithm

Require: $population\_size \geq 0, iteration\_gap \geq 1$
  $a\ b \leftarrow query\ user\ for\ 2\ favor\ music$
  $extract\ features$
  $generate\ population$
  $g \leftarrow 1$
  while $user\ not\ satisfied$ **do**
    $i \leftarrow 1$
    while $i < iteration\_gap$ **do**
      $update\ fitness$
      $selection$
      $crossover$
      $mutation$
      $i \leftarrow i + 1$
    end while
    $a\ b\ c \leftarrow 3\ top\ candidates$
    $S \leftarrow [S \mid query\ user\ for\ best\ favor\ music]$
    $g \leftarrow g + 1$
  end while

3.2 Evolutionary Algorithm

A gene pool of features has been obtained in the previous section, and the initialization stage generates individuals from this gene pool by combination. Each genotype can have multiple phenotypes, where genotypes are notes in the individual([D,D,C]) and phenotypes are how long each notes are played([1s,2s,1s]). Each individual consists multiple genes, because the number of notes contained in each gene may be different, and even if each individual has the same number of genes, they may represent different lengths of music.

Initialization: The initialization shortlists the gene pool whose elements are taken from the original song data in the previous step to randomly copy features as the smallest unit of gene to form a population of individuals, as shown in Fig. 4 (a), of specific population size, thus forming a population in which each individual has a phenotypic and genotypic gene. The genotypic gene is the feature that constitutes the individual, e.g. [feature1,feature4,feature4]. Given that each feature is a combination of notes, each occurrence in the original song may have a different duration pairing, taking the dominant gene [feature 1, feature 4, feature 4] distance = for example, where the three notes of feature 4 = [do, do, so] can be interpreted in an equal length or a one-to-two length relationship, and the choice of this interpretation is its phenotypic gene.

Selection Pressure: Typical EA takes hundreds of generations to obtain better individuals. But It is unreal to let the user evaluate for every epoch. Alternatively, the algorithm introduces an iteration gap **i** which suggests that for ***i*** generations user's choice list are same until the next query. By memorizing the characteristics of the music selected by the user, such as the order and speed of the features, the algorithm runs for a considerable number of generations to

produce a significant evolution before the user makes further decisions based on it. However, this approach has drawbacks; bias exists because the number of user choices is limited as they are not able to manually give appropriate selection regarding the slight change between generations. In addition, there also exists cases where all choices are undesirable and the user has to choose the one they dislike. The algorithm has limited information to obtain from the user's preferential choices, so there should be a mechanism to function in this missing supervision. Long Short Term Memory-based mechanism[20]can be applied to mock the selection force in the generation when the user's choice is absent.

3.3 Evolutionary Operators

This section describes the configuration of the evolutionary operator in the proposed evolutionary algorithm, including crossover, mutation, and parent selection.

Crossover: Shown in Fig.4 (b), one-point crossover is employed to allow a piece of music to permute its feature list. Some features represent the verse part of music while others may represent the intro, chorus, or outro part. A feature-level modification of order maintains the narrative and emotional diversity of music.

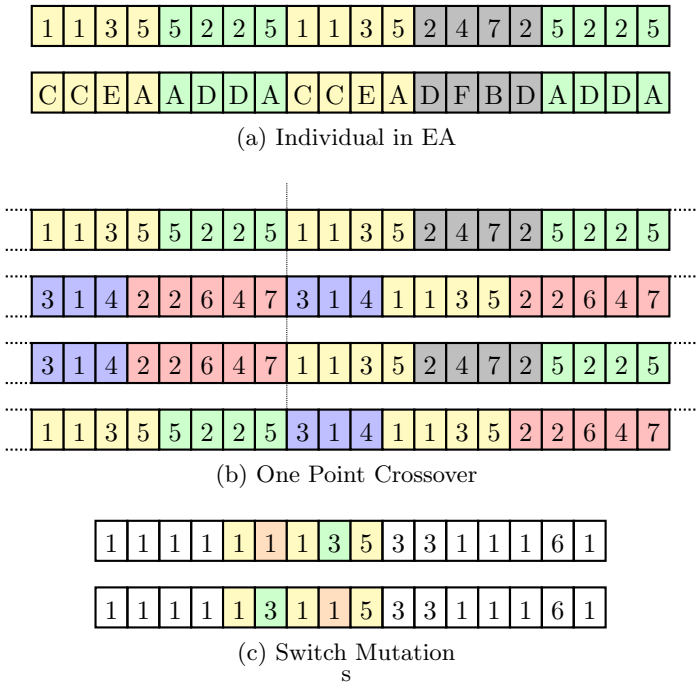
Mutation: Mutation is an operation at the gene level and creates new traits for the gene pool[17]. Therefore, our mutations will modify the note sequence which is the base element of the genotype. In EvoMedley, a switch mutation is used. Shown in Fig.4 (c) Every offspring have a mutation rate $\boldsymbol{m}$ to randomly exchange the position of two notes in a single feature. Those mutations are musically meaningful because the notes in a feature are expected to follow the same tonality for smooth soundness. The notes gene pool is already in a considerably harmonious melody, so the pitch leaves us little to modify; otherwise, it is very likely to break the consistency of the melody.

Selection: We utilize tournament selection to reserve a probability of survival of less competitive candidates. Three randomly chosen individuals complete with their fitness, followed by the best individual replacing the worst one. Multiple tournaments can be held during a generation.

3.4 Fitness Function

EvoMedley uses a interactive interface as shown on Fig.5, to let the user guide the direction of evolution by choosing best candidate in the current generation. The fitness function enumerates the performance of an individual regarding musical soundness and the user's preference. Therefore, the similarity S between the user-selected music and candidates equals the fitness of the candidate in the evolutionary algorithm. The similarity is measured by the Levenshtein Distance, which is the minimum number of steps required to switch, add and append to transform one string into another.

In the music scheme, there are similarities at the micro and macro levels. The former level is the least number of note modification steps between two



(a) Individual in EA

(b) One Point Crossover

(c) Switch Mutation
s

Fig. 4: Evolutionary Operators used in EvoMedley.In (a) each color represents a feature.In (b) first two lines are parents last two lines are offspring,dash line is the crossover pivot. In (c) coloured parts is a feature

genotypes. The latter level counts the least number of feature(a block of note) modification steps.

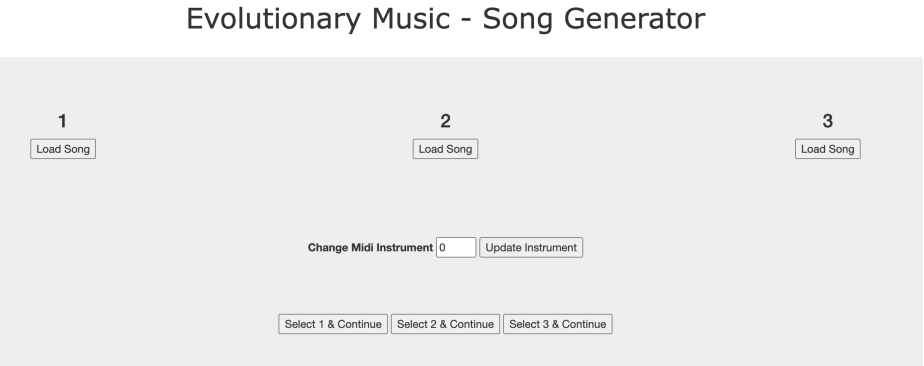
The fitness F is the weighted sum of similarities according to a list of patterns $\mathbf{S}$. Each time a choice is made, the individual chosen by the user as the best candidate of the current generation will be added to the model for all future fitness evaluations. To achieve that, a normal weight vector $\mathbf{w}$ with length l denotes the importance of choices influencing the computation of the fitness function in decaying order.

$$F = \mathbf{S} \cdot \mathbf{w} = s_{n-l} \times w_{n-l} + s_{n-l+1} \times w_{n-l+1} \cdots s_n \times w_n$$

Extremely, when l equals n, all user decisions will be taken into consideration as contributors for fitness calculation. As the user makes an increasing number of choices, the weight vector also grows its dimension in this case.

4 Experimental Results

This section describes the experiments and results. The most challenging problem is blending music from various styles. The experiments include three parts,



(a)

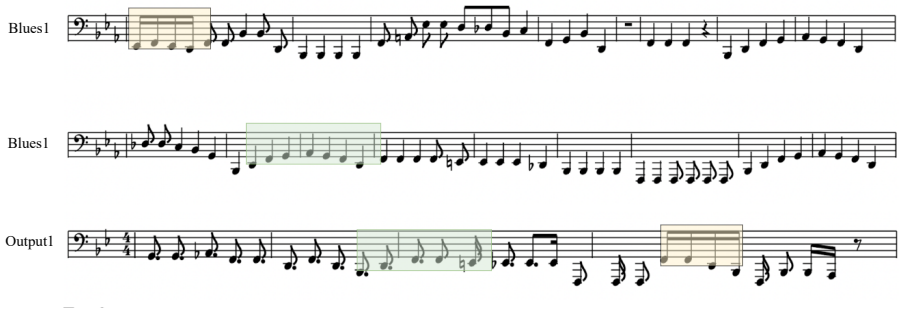
Fig. 5: In the interactive interface, users select from three top musical compositions with the highest fitness of the current generation to allow their performance to be used for future fitness assessments. Users continually make decisions until they successfully evolve an ideal individual.

feature extraction, music characterization experiments and different fitness functions. Files mentioned in this subsection can be found here.

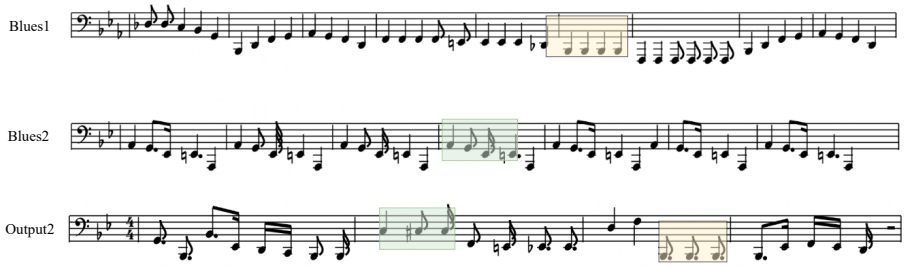
In the evolutionary algorithm section shown in Fig. 6, our preliminary experiment (a) explores using the same Blues melodies as input. During the evolution process, it was noticed that the generated individuals had the melodies of the new sequential templates, and as a result, the new music had the mood of the original music, we can see that the EvoMedley extracts the main melodies of the music and generate variations without violating the music theory. In experiment (b) we used two different similar music types of melody, respectively Blues1, and Blues2. The evolutionary process can be observed that unlike experiment (a), the generated individuals need to mix the two parents' characteristics in a uniform rhythm and tempo. Experiment (b) resulted in a random combination of different melodies in the output. In contrast, some of the melodies were played using a new uniform rate to maintain overall consistency. We can see that EvoMedley's evolutionary algorithm mixes features in the same genre of music thus generating new moods, and probabilistically maintaining musicality. In the third experiment (c) we used different styles of music melody as input to see the effect, the inputs were BluesB and metal riff. The evolutionary process generates individuals with a blend of different styles, with the result that the song may sound strange at rates that do not belong to that style of music. However, user choice largely eliminates these bad variants because choosing the best remix is to dispose of all other bad candidates. And the acceptable remix is likely to appear by running considerable a number of random epochs.

In the fitness function experiment, we compared two l values in the fitness function. In the first experiment where l equals n calculates the individual scores based on time decay for all user choices. The second experiment fixes the size

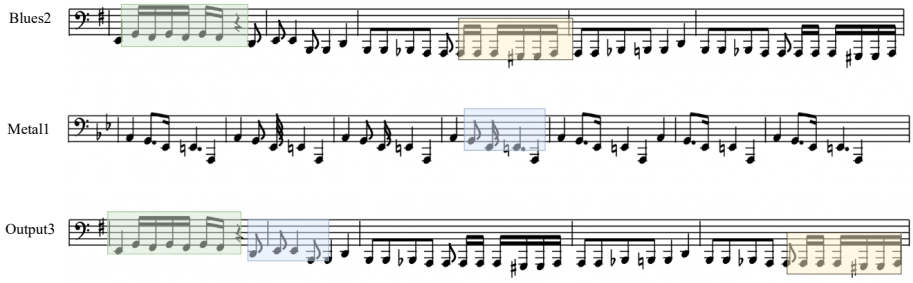
of **S** to forget obsolete choices. The first has more templates in the list and maintains individual diversity. However, the users are more able to control the



(a) In the first experiment, two ancestor parents are the same blues music. It is witnessed that the child music heritages the sixteenth notes from the first music clip and fluctuating melody from the second clip with some extent of mutation.



(b) In the second experiment, two ancestor parents are different music but in the same style -blues. The child music fluctuates generally like blues1 while having similar tunes of blues2, and specifically, heritages features of melody from both ancestors.



(c) In the third experiment, the child is a remix of metal music and blues. After mutations and crossover, musical elements are well merged.

Fig. 6: Music Characterization Experiments

direction of music generation in the second approach. In the first case, the user cannot escape the influence of early choices with resignation.

5 Discussion

5.1 Music Evaluation

Although human evaluations should always be the final choice for evaluating creative outcomes, human evaluation statistics obtained from inconsistent subjective experimental designs face challenges in terms of the reliability and replicability of results. However, objective metrics for music information [23], based on the dataset, we extract features rooted in music domain knowledge, output metrics, and quantify the features of each dimension of the dataset for users. However, these methods depend on the training data, and insufficient and unbalanced data can lead to different generated metrics. In this paper, we use a hybrid heuristic and human assessment method to obtain the fitness of the music in the EA system by combining the similarity between the generated music and the original music and the user's preference. Due to the lack of an objective measurement of musical soundness and the variety of personal preferences, the soundness mentioned here is defined by the user who makes the choice.

5.2 Future Improvement

For better soundness The current musical features should be aligned with multiples of bar length, resulting in rhythmic inconsistencies, where the musical features fragments that make up an individual should correspond to bar length. If two parents have the same bar length, the features can be aligned with multiples of the bar length to obtain a better rhythm. It is common that a measure occupies a different number of notes; some music has 2 quarter notes in a measure (Beethoven's 5th movement), while another s has 4 quarter notes in a measure (a twinkle twinkle little star). The least common multiple can be found to Aline different fragments in the same tempo. Also, we note the current fitness function is based on the similarity of the selected piece and the individual, where every single note is compared. However, in the scenario of music, some micro differences may not influence soundness. Therefore, a tolerance rate is suggested to allow two pieces of music to be "equally fit" with some minor differences.[14] Harmonization accompanying melody also improves the soundness and increases the diversity of music. One solution is to employ a composer of harmony that can be used to improve soundness. A bass evolutionary composer is introduced with predefined harmony data.[2] Our architecture is extendable because multimelody or harmonization can the track list preserves space for other instrument tracks.

6 Conclusion

In this paper, we propose EvoMedley, a system to explore the use of an interactive EA to mix music according to user preferences, generating unique tracks for

users that represent their style. We investigate the application of evolutionary algorithms to guide the generation of music that retains the features in the original version. We propose a method to extract features from the melody, reorganize them into a new melody, and modify the fitness function of the evolutionary algorithm with user guidance to generate the melody that the user wants more accurately. We use the similarity between the generated individual and the original music as the basis of the fitness function, and the fitness is updated as the algorithm runs.

To test the effect of EvoMedley on mixing multiple songs, we built an interactive interface. The EA outputs multiple potential individuals for each specific epoch, where the user selects the best one and updates the fitness function similarly to the best individual of that generation. From experimental results, we show that EvoMedley provides a promising direction for generating mixed-style music.

In future work, we will bring the algorithm closer to realistic scenarios of music generation, such as replacing the simple single-track melodies used with multi-track melodies of actual music, even including lyrics; investigating the effect of EvoMedley on multi-track song mixing; attempting to extend the interactive web page to allow users to select multiple instruments for multi-track free combinations. One of the exciting things about this work is that in the future it could offer the possibility for users to mix their favourite songs exactly the way they want and use them as a part of their social portfolio.

References

1. Brunner, G., Konrad, A., Wang, Y., Wattenhofer, R.: Midi-vae: Modeling dynamics and instrumentation of music with applications to style transfer. arXiv preprint arXiv:1809.07600 (2018)
2. De Prisco, R., Zaccagnino, G., Zaccagnino, R.: Evocomposer: An evolutionary algorithm for 4-voice music compositions. Evolutionary computation **28**(3), 489–530 (2020)
3. Dhariwal, P., Jun, H., Payne, C., Kim, J.W., Radford, A., Sutskever, I.: Jukebox: A generative model for music. arXiv preprint arXiv:2005.00341 (2020)
4. Eiben, A.E., Smith, J.E.: What is an evolutionary algorithm? In: Introduction to evolutionary computing, pp. 25–48. Springer (2015)
5. Gârbacea, C., van den Oord, A., Li, Y., Lim, F.S., Luebs, A., Vinyals, O., Walters, T.C.: Low bit-rate speech coding with vq-vae and a wavenet decoder. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 735–739. IEEE (2019)
6. Hofmann, D.M.: A genetic programming approach to generating musical compositions. In: International Conference on Evolutionary and Biologically Inspired Music and Art. pp. 89–100. Springer (2015)
7. Johanson, B., Poli, R.: GP-music: An interactive genetic programming system for music generation with automated fitness raters. University of Birmingham, Cognitive Science Research Centre (1998)
8. Leão, H.B., Guimarães, G.F., Ramalho, G.L., Cavalcante, S.V., et al.: Benchmarking wave-to-midi transcription tools. In: VII Brazilian Symposium on Computer Music, Curitiba, Brazil (2000)

9. Lee, D., Park, J.Y., Kim, J., Kim, J., Moon, J.: Understanding music sharing behaviour on social network services. Online Information Review (2011)
10. Liu, Q., Luo, M.: Cross-cultural examination of music sharing intentions on social media: A comparative study in china and the united states. International Journal of Human–Computer Interaction pp. 1–11 (2022)
11. Liu, W.: Literature survey of multi-track music generation model based on generative confrontation network in intelligent composition. The Journal of Supercomputing pp. 1–23 (2022)
12. Lopez-Rincon, O., Starostenko, O., Ayala-San Martín, G.: Algoritmic music composition based on artificial intelligence: A survey. In: 2018 International Conference on Electronics, Communications and Computers (CONIELECOMP). pp. 187–193. IEEE (2018)
13. Loughran, R., O'Neill, M.: Evolutionary music: applying evolutionary computation to the art of creating music. Genetic Programming and Evolvable Machines **21**(1), 55–85 (2020)
14. Madsen, S.T., Widmer, G.: Exploring similarities in music performances with an evolutionary algorithm. In: FLAIRS Conference. pp. 80–85 (2005)
15. Mittal, G., Engel, J., Hawthorne, C., Simon, I.: Symbolic music generation with diffusion models. arXiv preprint arXiv:2103.16091 (2021)
16. Moroni, A., Manzolli, J., Zuben, F.V., Gudwin, R.: Vox populi: An interactive evolutionary system for algorithmic music composition. Leonardo Music Journal **10**, 49–54 (2000)
17. Pant, M., Zaheer, H., Garcia-Hernandez, L., Abraham, A., et al.: Differential evolution: A review of more than two decades of research. Engineering Applications of Artificial Intelligence **90**, 103479 (2020)
18. Rawat, U., Singh, S.: Challenges in music generation using deep learning. In: 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE). pp. 553–558. IEEE (2022)
19. Rentfrow, P.J.: The role of music in everyday life: Current directions in the social psychology of music. Social and personality psychology compass **6**(5), 402–416 (2012)
20. Sak, H., Senior, A., Beaufays, F.: Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. arXiv preprint arXiv:1402.1128 (2014)
21. Salamon, J., Gómez, E., Ellis, D.P., Richard, G.: Melody extraction from polyphonic music signals: Approaches, applications, and challenges. IEEE Signal Processing Magazine **31**(2), 118–134 (2014)
22. Scirea, M., Togelius, J., Eklund, P., Risi, S.: Metacompose: A compositional evolutionary music composer. In: International Conference on Computational Intelligence in Music, Sound, Art and Design. pp. 202–217. Springer (2016)
23. Yang, L.C., Lerch, A.: On the evaluation of generative models in music. Neural Computing and Applications **32**(9), 4773–4784 (2020)
24. Zhou, Z., Xu, K., Zhao, J.: Homophily of music listening in online social networks of china. Social Networks **55**, 160–169 (2018)