A methodology for procedural piano music composition with mood templates using genetic algorithms

ROCHA DE AZEVEDO SANTOS, Luisa, SILLA JR., Carlos and DA COSTA ABREU, Marjory <http://orcid.org/0000-0001-7461-7570>

A methodology for procedural piano music composition with mood templates using genetic algorithms

Luísa Rocha de Azevedo Santos; Carlos Nascimento Silla Jr. *and Márjory Da Costa-Abreu †

Keywords: music composition; music emotion; music mood; genetic algorithm; procedural content generation

Abstract

Creating music in an automatic way has been studied since the beginning of artificial intelligence. One of the biggest obstacles of music generation is the vagueness and subjectivity of the *mood* or *emotion* transmitted by a music piece. In this work, we experiment with the generation of piano music using template pieces, represented in MIDI format, as a mood directive. We generated a population of random pieces for templates of two opposing moods - happy and sad - and evolved them with a genetic algorithm until their intended mood was close enough to their respective templates. The fitness function that we implemented uses MIDI statistical features to calculate the distance between the given piece and the template. The generated music pieces were evaluated by human listeners thorough a questionnaire. This evaluation has shown that the generated music pieces were able to express the same mood as the template. However, they still sounded computer-generated, probably due to the lack of rhythm regularity and synchronicity.

1 Introduction

The area of music generation has been increasingly explored throughout the years [9, 15]. The approaches vary from Markov models or hidden Markov models [22], to neural networks [1], evolutionary algorithms [20], and several other methods [12].

Traditionally, however, music generation researches focus only on generating good music, without considering that whoever uses the system may want to use the music for a specific situation, so they might need it to transmit a particular feeling. Examples of this are the incidental (background) music used in games [23] and movies [2].

It was observed that proprieties in music such as scale modes, presence of dissonance, melody motion and rhythm consistency can significantly influence the mood of a musical piece [8], and that this mood can affect the listener's emotions in many ways [11].

2.PUC-PR, Curitiba, PR, Brazil, carlos.sillajr@gmail.com
1.Sheffield Hallam University. Sheffield, South Yorkshire, UK, m.da-costa-abreu@shu.ac.uk

Keeping that in mind, we decided to study the generation of music with a mood directive. We will experiment with musical properties to see how they can influence the mood of the final piece.

The goal of this work is to design a system that automatically generates a musical piece, given a template piece as mood directive, without requiring in-depth knowledge about music theory and composition. This template piece may be any composition that suggests the intended mood and respects the system's limitations. This composition will be represented in MIDI [7] - Musical Instrument Digital Interface: a widely used file format which can be used both to play music and to represent it as a musical score [10].

The generated music will have two parts: the melody, a sequence of individual notes played one at a time that gives identity to the piece; and the harmony, the background chords that accompany the melody.

A template piece with these two parts will be provided to the system. After analyzing the piece and extracting the most important features, the system will generate another - unique - piece, also with these two parts, with mood that is as similar as possible to the mood of the original template.

The method we will experiment with is the genetic algorithm [5] to evolve an initial population of randomly generated pieces. The fitness function that we will use for the algorithm is a similarity function that compares the mood of the generated piece with the mood of the given piece, using an euclidean distance calculated from musical features. This way, a unique piece should be generated since the starting population is also unique, but the mood should still be similar to the template if it is well represented in the fitness function.

In the next sections, we will briefly present the musical terminology used in this work and some of the previous works that are related to ours (Section 2). Next, in Section 3, we will describe our modelling for this problem, including genetic operations and representation of a piece. In Section 4, we will present the results of the tests that we executed. Lastly, in Section 5, we will summarise the results and propose future works.

2 Terminology and Related Works

In our work, we will use the terminology defined by Crotch [3], where he defines the basis of modern musical theory including **notes**, **pitch**, **interval**, **octave**, **scale** and others.

Mcauley [17] described the **time signature** of a piece as the specifier of two numbers: the first, the numerator, is the size of a **beat** and the second, the denominator, is the number of beats per **measure**. The reference time unit used in this work for chords duration. The **tempo** of a piece is the speed in which the notes are played - indicated by the number of beats per minute (**BPM**).

Harmony is the sound that accompany the foreground **melody** in a musical piece. Crotch [3] defines harmony as a sequence of **chords**, where each chord is a set of notes played at the same time or within a limited interval of time. A sequence of chords is also called a **chord progression** [21].

The notes in a chord may be played in an **arpeggio**, in which each note is played one after another instead of at the same time [21].

We will need a method to compute the distance between two music pieces. In this section we review the works that model and evaluate some possible theoretical features to describe a music piece.

There are several ways to extract features from songs, depending of the type of data we have available. In this work, we are using MIDI as a representation and for this reason we are interested in works that deal with symbolic music representation. Meanwhile, [6] also proposed some metrics to classify music. The accuracy varied from 70.66% (12 classes) to 87.75% (2 classes).

The fitness function and generic operations varied a lot in works that used genetic algorithm for music generation. The works [27, 28] used the judgment of a human listener to evaluate the individuals. A very common fitness function is a combination of rule-based objectives [4, 16, 20, 25, 26]. Another common fitness function is a classifier trained with human-made examples [13, 14, 19].

The works [16, 26], like ours, also use a template piece as a guide to the generated pieces, but differ from ours in the representation of a piece and in the genetic operators. Both works represented piece as a string of numbers, while in our it is represented with an abstract model. For the genetic operations, the work [26] used standard crossover and mutation operations, while the work [16] used mutation that altered the notes' pitches and no crossover operator.

In our work, the mutation operations include other variations based on musical theory, and the crossover operator that we use also works in a different way: instead of the standard cut-and-swap crossover, we combine musical parts from three different parents.

3 Modelling

To reduce the scope of our problem, we limited the music pieces (both input and output) to: using only diatonic scales; a single track for the melody (monophonic); a single track for the harmonic chords; no percussion (drums, cymbals, etc.) tracks; piano as the only possible instrument; only one chord per measure; pieces with four measures; a single arpeggio pattern for all four measures. Also, we will not analyze patterns, repetitions or overall high-level musical structure, and instead we will focus on generating a single motif, long enough to communicate a feeling.

Instead of using typical strings of numbers as an individual, here we use an abstract hierarchical model to represent a music piece, which will be an individual in the population. This way, the application of the genetic operators are more intuitive, since they modify abstract musical properties instead of individual numbers.

In this model, a *piece* is as a tuple that contains: the length of the piece in measures (integer value); the time signature (two integer values, numerator and denominator); the BPM (float value); the piece's scale; the foreground melody; and the background harmony. Except for the melody and harmony parts, all the other information is obtained directly from the MIDI file's header. The last three fields are described in the next subsections.

A *scale* is a pair consisting of its *root pitch*, varying from 0 to 11, and its *mode*, varying from 0 to 6. It is important to note that a scale is represented by displacements from the cannon scale, C major. The mode value is the number of times the Major mode's pattern `2-2-1-2-2-2-1` is displaced by one. That being said, each mode is represented as: 0 - Ionian (Major); 1 - Dorian; 2 - Phrygian; 3 - Lydian; 4 - Mixolydian; 5 - Aeolian (minor); and 6 - Locrian.

A *pitch*, which in the MIDI file is represented by an integer from 0 to 127, interpreted as the absolute sound frequency. Here, it is converted to a pitch relative to its scale, represented by: the pitch's *degree*, a number from 0 to 6, which is the pitch's position relative to the scale; the pitch's *accidental* (0 is natural, -1 is flat (♭) and 1 is sharp (♯)); and the pitch's *octave*, relative to the scale's root.

The *melody* is a set of *notes*, each one defined by: the *pitch* being played; the time in beats in which this note is played (also known as *attack*); and the *duration* in beats of the note.

The *harmony* is a pair containing a sequence of chords and the *arpeggio pattern*, a set of notes just like the melody, used for the chords. Each chord is represented by a pitch, which is the chord's tonic relative to piece's scale (in this work, we use the lowest pitch in the chord as its tonic). The tonic is then used to generate a scale, which is the base scale for the arpeggio's notes.

In our proposed genetic algorithm, each initial individual will have the same length of the target piece, the same number of notes in the melody and the same notes in the arpeggio pattern. The generation of a random piece

follows the steps below:

1. The scale, BPM and tempo will be the same as the target's. The arpeggio structure - notes played relative to the chord's root - were copied from the template.

2. The start of each note is chosen by a random number from the start of the piece to the end of it. The number of notes in the piece will be the same as the number of notes in the template.

3. The pitch and the duration of each note are uniformly generated from the range in the original piece - from the shortest to the longest duration, and from the lowest to the highest pitch.

After that, the musical features of each individual are calculated and stored in a vector. This vector is used to calculate the individual's proximity to the template, which is an Euclidean distance between the piece's feature vector and the template's feature vector.

For the features, we selected some of the features proposed by [18] that are applicable to our problem restrictions. The complete list of features that we implemented is detailed in [24]. They are separated by classes: 10 *rhythm* features, 14 *pitch* features, 13 *note* features and 15 *interval* features. The rhythm features are calculated once, the pitch and notes features twice each, and the interval features three times, totalizing 109 features.

The final fitness of a piece p given the template t is described as: $f = \sum_{i=1}^{109}(\alpha_i * p_i - \alpha_i * t_i)^2$, where each α_i is the weight given to the feature i; p_i is the feature i extracted from piece p; and t_i is the feature i extracted from the template. We did not compute the square root of the sum because the exact value of the distance was not used; the summation is calculated only to compare and sort individuals by distance.

Next, the population is sort by fitness, and a fraction of the population with the fittest individuals (the elite) is separated. The remaining individuals are replaced by new ones in the next generation, created from a crossover of elite individuals.

For the mutation, we defined three sets of mutation operations. Each mutation operation is applied independently, according to its rate.

The first set of mutation operations are applied to the whole piece at once, since it changes the scale and rhythm signatures. These operations are **Change mode** which changes the scale mode, keeping the root key and the degrees of each note in the melody, using a random number from 0 to 6; and **Change BPM**: changes a new BPM value, from 50% to 150% of the current BPM.

This set of operations is applied to the notes in the melody are **Change start time** which moves the note start time, using a random float between the end of the previous and the next notes in the melody; and **Change duration**, which changes the note end time, using a random float from the end of the previous note to the start of the next note in the melody.

This set is applied to a set of pitches - both to the melody and the chords' tonics. These operations are: **Change pitch degree** that changes scale degree of the pitch, decreasing or increasing it by one scale step; **Change pitch accidental** that changes the accidental of the pitch, increasing or decreasing the pitch by a half tone (one semitone); and **Change octave** which increases or decreases all pitches by one octave.

There is a single crossover operation, which will occur with three selected individuals: the first one will provide the signature (scale and tempo), the second one will provide the melody notes and the third one will provide the chords. These three individuals are selected uniformly from the set of elite individuals. To separate the elite set, the population is sorted by fitness, and the n fittest individuals are selected as the elite.

4 Experiments and Results

During this section, we will indicate links to our music showcase site, where reader can listen to the mentioned pieces, visualize their music sheets and download the MIDI files.

We implemented our system using a Java library called jMusic. The code is published on GitHub. For all tests, we used four templates that were hand-made by us, two "happy" pieces and two "sad".

All the experiments were executed with a population size of 60, and an elite size of 15. Due to time limitations, we were not able to test variations of these values.

4.1 Human Evaluation

We published a questionnaire via Google Forms showing 4 templates, two "happy" ones and two "sad" ones, and also 2 procedurally generated pieces based on each template, one using the random seed 0[1] and other using seed 1[2], totalizing 12 pieces. We selected the fittest pieces from the 4000^{th} generation. For each piece - either a template or a generated piece -, we asked the participants to: (1) Rank the mood transmitted by the music piece, from 1 (sad) to 5 (happy); and (2) Evaluate how they thought the piece was composed, being 1 (certainly by a computer) to 5 (certainly by a human). In total 40 different people answered: 20 with no music theory background, with 8 a basic background, 9 intermediary and 3 advanced.

In general, the generated pieces transmit a mood similar to the template it is based on. The pieces generated from Happy2 and Sad2 were pretty close to the intended mood, while with Happy1 they sounded slightly sadder, and with Sad1 they sounded slightly happier.

[1] https://luisaras.github.io/piano-repo/#seed0
[2] https://luisaras.github.io/piano-repo/#seed1

In the case of the Happy1 pieces, we believe that the chords in the harmony contributed for the sadder mood. Both random seeds generated the same chord progression (vi-I-vi-IV), but this progression was not the same as the one in the template (I-vi-IV-V). The first progression may sound sadder because it contains two minor chords (vi), while the second one only contains one. Besides this, there may be something in the melody intervals that was not captured by the features we calculate. The irregular rhythm could also have contributed to the "weirdness" in the piece, which disturbs the happy mood.

In the case of the Sad1 pieces, both pieces had the same chord progression as the template, so it was not responsible for the mood difference. The generated pieces have a little more note pitch variety, which usually makes music sound happier, so this could have been the reason. This, and the fact that the notes in the template are longer than the notes in the generated pieces, and longer notes usually sound sadder. Just like the Happy1 pieces, it could also have been some interval information that was not taken into account.

The listeners seemed to prefer the template pieces. Among the templates, Happy2 was the only one that sounded more like a computer-generated piece then like a human-made one, and one of the pieces generated from it sounded more natural than the original. It has likely happened because the beginning of the piece sounded slightly dissonant.

It was reported by one of the people who answered the questionnaire that sometimes the procedurally made pieces could be identified by its lack of regularity and melody-harmony synchrony in the rhythm, so this problem may be investigated in future works to make the pieces sound more natural.

However, according to the results obtained from the questionnaire, the piece with the smaller distance did not necessarily perform the best to the human ear. This suggests that the weights in the fitness function could still be improved.

4.2 Feature Weights

After the results of the questionnaire, we decided to test our fitness function separately. The initial weights were chosen subjectively between a few combinations tested. Some features could be normalized, but some could not, so we had to manually try different configurations. What piece actually sounds better is really a subjective matter, so we did not try to find exact values for the weights.

For the initial tests, these weights were good enough to evolve the population. Comparing the fittest individuals from generations 1, 100, 1,000 and 4,000. We used the template Sad1 to demonstrate it since it was the one that showed the largest differences from one generation to another. From generation 1,000 to 4,000 it did not show a great improvement, but it is clear that from 1 to 100 the fittest individual was a lot different, sounding sadder. From 100 to 1,000 it improved a little the intervals and note durations.

After the results from the human evaluation, we compared the performance of two versions of the fitness function: the one with the weights we proposed and the one with all weights equal to 1. For this, we only generated the initial population, without evolving it, and sorted it by fitness using these weights.

We first compared these two weight configurations using the template Happy1. For each one, we uploaded a pair of pieces, the best and worst. Being the first generation, it is expected that neither of the pieces are satisfactory results, but we can still compare the best with the worst of each configuration.

Both configurations selected reasonable pieces for the fittest. The weighted fittest sounds a little less random than the non weighted fittest, probably because of the biggest weights given for the attack variation. The most noticeable difference, however, is between the two least fit pieces: the weighted one sounded way more dissonant because of the high weights given for accidentals and dissonant intervals, which means that the dissonant pieces are way more penalized than the more harmonious ones.

However, our next test shows that this fitness function might be too "afraid" of selecting individuals with a high incidence of accidentals and dissonance. Testing with pieces generated with template Sad1 the non-weighted version was able to select better the pieces that sounded that most similar to the template - that is, it performed better for the mood.

This demonstrated one of the most difficult parts for us in choosing the weights: the balance between the importance of sounding natural and the importance of transmitting the correct mood. The higher weight given to attack variation and the accidental incidence were chosen to avoid dissonance and randomity in the rhythm, both elements that sound unpleasant and unnatural to the human ear. However, increasing these weights sometimes makes the algorithm select pieces that do not have the right mood. A possible solution to this is to separate the generation of the melody, which will be focused on the pleasantness of is sound, from the modification until it sounds like the template. Another possible improvement is to remove the weight given to dissonance and prevent its occurrence by other means, like conditional mutation.

In summary, these results suggest that the fitness function that we initially proposed may work for some types of template better than others. Different weight configurations for the feature classes could be tested and evaluated by other questionnaires in the future.

4.3 Mutation Rate

Now we will discuss the effects of the mutation rates in the final results and the convergence of the individuals. We noticed a great importance of the signature mutations for the final mood. Even with static melody notes and

harmony chords, just changing the scale and the BPM of the piece significantly change the mood, sometimes even making the melody unrecognizable.

Instead of random individuals, we populated the system using clones of one of our hand-made templates, Happy1. With only these two mutation operations, both with the weights 0.25, we evolved it to the 100^{th} generation, using as a template the Sad1 music piece to make it sound sad, and compared it to the original Happy1 piece. The modified version sounds way slower, resulting from the BPM mutation, and more dissonant, resulting from the scale mode mutation.

As expected, note features did not appear in the list, since they were kept the same in the modified version. The biggest differences were in the intervals. The Sad1 piece has a lot more chromatic motion than Happy1 piece, as well as other dissonant intervals, which helps the piece sound more unsettling. Note density were the only rhythm feature in the list, but had a really important role in the mood, since it was what made the piece slower.

The inverse was also tested and we saw the biggest differences between the original Sad1 piece and the fittest piece from the 100^{th} generation using Happy1 as template. In this case, the final piece did not sound as happy as the template, but it was probably because of the lack of harmony mutation, since the falling progression in the sad piece did not change and has a significant influence in the mood. However, it is noticeable that the modified version sounds happier than the original. It sounds faster, because of the higher note density, and brighter, because of the lower dissonance ratio and higher presence of stepwise motion and other harmonic intervals.

In both cases, we can see some details that could still be improved, like the excessively higher ratio of tritones - an interval that sounds dissonant -, which could have happened because of the lack of mutation in chords' tonics and melody pitches.

Next, we defined initial melody and harmony note mutation rates, and also the melody rhythm mutation rates (for change attack and duration, both, with 0.025). Next, we evolved the population that were created without a base piece for the melody, that is, randomly, through the method described in Section **??**. We executed the same tests as the previous subsection.

The modified Happy1 version did sound a little sadder than the original, but, compared to the one generated with the signature mutation, this one is happier. This is probably because of the higher note density, which did not appear in the table, since it did not change. The falling motion in the chords and the more dissonant intervals were the most responsible for the sadder mood in the modified version in this case.

The modified Sad1 version, however, still sounded too sad. Since it could not change the note density, it would always sound too slow.

The features that changed the most are basically the same in both cases, but the qualities of the results were very different. We believe that a greater number of occurrences of the tonic note would also make piece sound happier, since the tonic note does occur more in the Happy1 template.

We noticed, however, that the distances from these two pieces to their templates where smaller than the distances from the pieces generated with the signature mutation, despite the latter ones sounding better. This is another indicative that the weights given for the features are not accurate.

With our initial mutation rates, we plotted the distance of the best individuals from generations 1, 100, 1000, and 4000. The initial distance is very high and variable in the first generation since it depends entirely on the random seed. However, all pieces improved extremely fast from the first generation to the 100^{th}, to less than 10,000, no matter how unfit was the first generation. From generation 100 to 1000, all of them improved significantly, but some more than others. And from 1000 to 4000, some of them still improved, but some had already converged, like Happy2 (seed 1) and Sad2 (seed 0).

Except for Sad2 (seed 1), which reached a distance way smaller than the others, all pieces have reached a similar distance in generation 4000, which means that the mutation rates are probably generic enough to work for both happy and sad moods with little dependence on the random seed.

Lastly, we tried to vary a little the mutation rates from the initial parameters described in the previous subsections, to see if the populations would evolve faster. We used the template Happy1 and evolved until generation 4000, using random seed 0. The distances are showed in Table 1.

Mutation	New Value	Fittest Distance
-	-	282.76
Signature (scale mode and BPM)	0.0005	88.56
Signature (scale mode and BPM)	0.025	228.43
Melody note attack	0.25	158.29
Melody note duration	0.25	73.80
Melody note degree	0.025	144.66
Melody note degree	0.25	1890.33
Melody note accidental	0.25	13772.98
Melody note octave	0.025	222.84
Melody note octave	0.25	297.79
Chord tonic degree	0.025	111.67
Chord tonic degree	0.4	569.49
Chord tonic accidental	0.25	183.77
Chord tonic octave	0.025	222.84
Chord tonic octave	0.4	297.79

Table 1. Distances of the fittest individuals to the template, for each mutation rate variation.

From all the combinations, the best results seemed to be the ones with a lower signature and higher note duration mutation rates. We believe that this happened because a change in the signature can drastically change the mood, for better but also for worse, so many potentially good pieces may be ruined by a wrong signature. Also, a higher duration mutation rate could have helped to find fitter individuals because the initial population has

all notes with full duration and, since the Happy1 has some silent time in the melody, the initial rate of 0.025 could have made it too difficult to generate a piece with better note duration.

The combination that performed the worst was the change in the melody note accidental. This is expected, since it was given a high weight to the accidental feature. The only time this mutation is useful is when the template has an accidental: after the piece gets the same number of accidental notes as the template, this mutation ideally should never happen again. If this rate is too high, though, there is a high chance that some note in the melody receives an accidental, which would increase its distance a lot.

These results suggest that the mutation rates should be defined according to the feature weights. For this reason, a conditional mutation - such as increasing the accidental mutation rate when the difference in accidental ratio is high, and decreasing it when the distance is low - could help the population to converge faster.

5 Conclusions

Our main objective with this project was to provide a way to generate music automatically with some kind of directive for the mood expressed by the piece. Therefore, we proposed a system where the user can provide a template piece to direct the mood of the piece she or he needs, and then unique pieces would be generated by the system according to this mood.

We modelled a musical piece as an individual from a population evolved with a genetic algorithm. The results of our experiments demonstrated to be relatively effective in its original goal, which is to generate pieces according to the given mood template. We noticed that more weight on the dissonance features helped to select the most pleasing pieces, but not the most fitted for the mood described by the template.

Moreover, the fittest piece was not always the most natural-sounding either. The responses showed that, in general, the generated pieces sound more artificial than the hand-made ones. We made a few tests varying the mutation rates to see if they could generate fitter individuals in the same number of generations.

References

[1] A.E. Coca, R.A.F. Romero, and L. Zhao. Generation of composed musical structures through recurrent neural networks based on chaotic inspiration. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 3220–3226. IEEE, 2011.

[2] A.J. Cohen. Music as a source of emotion in film, 2011.

[3] W. Crotch. Elements of musical composition, 1812.

[4] P. Donnelly and J. Sheppard. Evolving four-part harmony using genetic algorithms. In *European Conference on the Applications of Evolutionary Computation*, pages 273–282. Springer, 2011.

[5] L.J. Fogel, A.J. Owens, and M.J. Walsh. Artificial intelligence through simulated evolution, 1966.

[6] J. Grekow and Z.W. Ras. Detecting emotions in classical music from midi files. In *International Symposium on Methodologies for Intelligent Systems*, pages 261–270. Springer, 2009.

[7] R. Guerin. Midi power! the comprehensive guide, 2010.

[8] K. Hevner. Experimental studies of the elements of expression in music. *The American Journal of Psychology*, 48(2):246–268, 1936.

[9] Hiller, L.A. and Isaacson, L.M. Experimental music: composition with an electronic computer, 1959.

[10] T. Huang, G. Xia, Y. Ma, R. Dannenberg, and C. Faloutsos. Midifind: fast and effective similarity searching in large midi databases. In *Proceedings of the 10th International Symposium on Computer Music Multidisciplinary Research*, pages 209–224, 2013.

[11] C.L. Krumhansl. Music: A link between cognition and emotion. *Current directions in psychological science*, 11(2):45–50, 2002.

[12] Y. LiChia, C. SzuYu, and Y. YiHsuan. Midinet: A convolutional generative adversarial network for symbolic-domain music generation, 2017.

[13] M.Y. Lo and S.M. Lucas. Evolving musical sequences with n-gram based trainable fitness functions. In *IEEE Congress on evolutionary computation*, pages 601–608, 2006.

[14] B. Manaris, P. Roos, P. Machado, D. Krehbiel, L. Pellicoro, and J. Romero. A corpus-based hybrid approach to music analysis and composition. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22:1, page 839. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[15] H.H. Mao, T. Shin, and G. Cottrell. Deepj: Style-specific music generation. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, pages 377–382, Jan 2018.

[16] D. Matic. A genetic algorithm for composing music. *Yugoslav Journal of Operations Research*, 20(1):157–177, 2010.

[17] J Mcauley. Tempo and rhythm. *Music Perception*, pages 165–199, 08 2010.

[18] C. McKay. *Automatic genre classification of MIDI recordings*. PhD thesis, McGill University Canada, 2004.

[19] P. Mitrano, A. Lockman, J. Honicker, and S. Barton. Using recurrent neural networks to judge fitness in musical genetic algorithms, 06 2017.

[20] E. Ozcan and T. Ercal. A genetic algorithm for generating improvised music. In *International Conference on Artificial Evolution (Evolution Artificielle)*, pages 266–277. Springer, 2007.

[21] W.A. Palmer. Scales, chords, arpeggios and cadences, 1994.

[22] D. Ponsford, G. Wiggins, and C. Mellish. Statistical learning of harmonic movement. *Journal of New Music Research*, 28(2):150–177, 1999.

[23] A. Prechtl. *Adaptive music generation for computer games*. PhD thesis, The Open University, 2016.

[24] L.R.A. Santos. An analysis of procedural piano music composition with mood templates using genetic algorithms, 2018.

[25] M. Scirea, J. Togelius, P. Eklund, and S. Risi. Affective evolutionary music composition with metacompose. *Genetic Programming and Evolvable Machines*, 18(4):433–465, 2017.

[26] C.K. Ting, C.L. Wu, and C.H. Liu. A novel automatic composition system using evolutionary algorithm and phrase imitation. *IEEE Systems Journal*, 11(3):1284–1295, 2017.

[27] N. Tokui and H. Iba. Music composition with interactive evolutionary computation. In *Proceedings of the 3rd international conference on generative art*, volume 17:2, pages 215–226, 2000.

[28] H. Zhu, S. Wang, and Z. Wang. Emotional music generation using interactive genetic algorithm. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 1, pages 345–348. IEEE, 2008.