EVOLUTIONARY ACCOMPANIMENT SYSTEMS FOR CREATIVE MUSIC GENERATION

by

SHU ZHANG

(Under the Direction of Khaled Rasheed)

ABSTRACT

In this thesis, two music accompaniment systems are presented. Evac (the **ev**olutionary **ac**companist) is a system that engages in musical improvisation with the user. It uses a novel, implicitly interactive, genetic algorithm (GA), which allows the user's actions to influence Evac's musical performance without the need for explicit rating of individuals. Evac runs in real time, allowing the user to experience the same kind of exploration that happens in real life improvisation scenarios with other musicians. EvolMusic is an accompaniment system involving human preference learning. It allows direct control from the user over the accompaniment by using machine learning techniques to learn a fitness function from the user's preferences. EvolMusic records a piece of the user's musical input, generates different accompaniments, lets the user vote for his or her favorite, adjusts the GA's fitness function, and then generates new accompaniments which can be further used to learn the user's preferences.

INDEX WORDS: Evolutionary Computing, Music Accompanist, Interactive Genetic Algorithm, Real Time, Machine Learning

EVOLUTIONARY ACCOMPANIMENT SYSTEMS FOR CREATIVE MUSIC GENERATION

by

SHU ZHANG

B.E., Chang’an University, China, 2010

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2013

EVOLUTIONARY ACCOMPANIMENT SYSTEMS FOR CREATIVE MUSIC GENERATION

by

SHU ZHANG

DEDICATION

To my loving family and all the people who care about me.

ACKNOWLEDGEMENTS

The presented work benefited from http://www.freesound.org/, which provided all the sounds files that are used in Evac and EvolMusic.

I would also like to thank Dr. Khaled Rasheed for his guidance, encouragement, and assistance in my completing this thesis. I would also like to thank Dr. Walter D. Potter and Dr. Charles Cross for serving on my committee and providing help. Also, I would like to thank Christopher Thomas Bailey and Roi Ceren for their kind help on my research.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS v

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER

1 INTRODUCTION AND LITERATURE REVIEW 1

2 EVAC: AN EVOLUTIONARY ACCOMPANIST 4

2.1 ABSTRACT 5

2.2 INTRODUCTION 6

2.3 BACKGROUND 8

2.4 EVAC AND THE USER 10

2.5 THE GENETIC ALGORITHM APPROACH 12

2.6 RESULTS AND ANALYSES 20

2.7 CONCLUSION 22

2.8 FUTURE WORK 23

2.9 REFERENCES 25

3 EVOLMUSIC – A PREFERENCE LEARNING ACCOMPANIST 28

3.1 ABSTRACT 29

3.2 BACKGROUND 30

3.3 USER INTERFACE OF EVOLMUSIC 35

3.4 SYSTEM DESIGN 39

3.5 RESULTS AND CONCLUSION 43

3.6 FUTURE WORK...44

3.7 REFERENCES 45

4 SUMMARY AND CONCLUSIONS 47

5 BIBLIOGRAPHY..49

LIST OF TABLES

Page

Table 1: Keyboard keys and pitches 11

Table 2: Technical overview of Evac's GA parameters 13

Table 3: Musical intervals and their corresponding similarity values 18

Table 4: Interval categories..41

Table 5: Implementation of the three model manipulations 43

LIST OF FIGURES

Page

Figure 1: Representation for a single music note and an individual.................................14

Figure 2: Musical phrase (above) and corresponding Evac individual representation (below) 15

Figure 3: Music intervals shown on piano keyboard..17

Figure 4: Fitness function: $f(x) = -(x-10)^2$. Input is a similarity rating. 19

Figure 5: Startup window of EvolMusic...36

Figure 6: EvolMusic interface when user inputs music...37

Figure 7: EvolMusic interface when new accompaniments being evolved.......................38

Figure 8: EvolMusic interface when it’s ready to let user play the generated accompaniments and vote...39

Figure 9: Representation for a single music note and an individual.................................40

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

Interactive music systems are computer systems whose output is modified in response to their musical input (Rowe, 1993). There has been extensive research conducted in the area of interactive music systems nowadays. Real-time accompaniment system is one of the popular applications of interactive music system.

In the literature, real-time accompaniment applications have adopted various approaches (Dannenberg, 1994). For example, in conventional taped accompaniment (which is not interactive), the accompaniment is not able to synchronize with the user's input, and it does not follow the user's input.

On the other hand, there are also interactive accompaniment systems that are able to detect the user's input flow and respond to it accordingly. For example in the system proposed in Dannenberg's paper 'An On-Line Algorithm for Real-Time Accompaniment' (1984), the algorithm was able to the follow the user's input by modeling the user's input and the stored score as two sequences of events, detecting the best match between this two sequences, and generating the timing signals used to trigger the correct accompaniment.

However, there were inherent limitations of such an accompaniment system: it was only able to adjust tempos on a small scale, based on only the tempo information, ignoring all the other cues such as articulation, loudness and so on. Moreover, it did not allow matching to music made up of sets of unordered or simultaneous events. In other words, the system was lack of flexibility and freedom of the accompaniment.

One good solution to this problem is to apply the evolutionary computing techniques to generate impromptu accompaniment. Evolutionary computing is a research area which draws inspiration from the process of natural evolution. It reflects the phenomenon of survival of the fittest in nature.

The impromptu accompaniment generated using evolutionary computing techniques has the property of both being stochastic and following the user's input. Following the user's input makes the generated music 'accompaniment' rather than some random music notes. Being stochastic brings flexibility and freedom to the process of generating accompaniment. The process doesn't need to follow a set of rules of aligning and matching, and the resulting accompaniment is no longer limited to a pre-set music scores.

This rest of the thesis is structured as follows. The second chapter proposes a system named 'Evac', which engages in musical improvisation with the user using genetic algorithm. Details on how the system addresses the three main considerations when an evolutionary algorithm is applied to musical composition are discussed. The drawbacks in the 'Evac' system are also discussed, with the main one being a mathematically coded fitness function in the genetic algorithm. In order to offer a possible way to address this problem, another system, namely 'EvolMusic' is proposed in Chapter 3. In spite of the fact that the EvolMusic and Evac both involve genetic algorithm, they are designed to address different issues and hence possess different characteristics. EvolMusic addresses the fitness function part of the algorithm in a different way, which enables the system to learn the user's preference. Details of the implementation of the system, as well as its achievements and drawbacks are discussed. In

Chapter 4, the research conducted in this study is summarized and possible areas for future work are suggested.

CHAPTER 2

EVAC: AN EVOLUTIONARY ACCOMPANIST[1]

[1] Shu Z, C. T. Bailey, Khaled Rasheed Accepted by *Genetic and Evolutionary Methods 2013*. Reprinted here with permission of publisher.

2.1 ABSTRACT

Evac (the **ev**olutionary **ac**companist) is a system that engages in musical improvisation with the user. Evac uses a genetic algorithm (GA) to invent musical phrases that are neither too similar to the user's input, nor too different. It is notable for two reasons. First, it uses a novel, implicitly interactive, genetic algorithm, which allows the user's actions to influence Evac's musical performance without the need for explicit rating of individuals. Second, in contrast to many pieces of software in the world of evolutionary music and art, Evac runs in real time, allowing the user to experience the same kind of exploration that happens in real life improvisation scenarios with other musicians. Evac must also deal with the design problems of dynamic environments, since our GA's fitness function relies on the user's input. Sample music resulting from the system is available.

2.2 INTRODUCTION

Our problem is music composition. In particular, our system allows the user to play music while simultaneously generating impromptu accompaniment to go along with the user's input. The musical "brain" of the system is a genetic algorithm implemented in C#.

Evolutionary computing is a research area which draws inspiration from the process of natural evolution. It reflects the phenomenon of survival of the fittest in nature.

The two cornerstones of evolutionary progress are competition-based selection, and the phenotypic variations among members of the population (Eiben & Smith, 2003). The most important components for any evolutionary algorithm includes: representation (definition of individuals), an evaluation function, population, parent selection mechanism, variation operators, recombination and mutation, and a survivor selection mechanism. Evolutionary algorithms (EA's) have three basic features that distinguish them from other algorithms: EA's are population based; they use recombination, mutation, or other genetic operators; and they are stochastic.

In the literature there have been four main approaches: genetic algorithms, evolution strategies, evolutionary programming (EP), and genetic programming. All these dialects of evolutionary computing follow these general outlines, with differences only in technical detail. For example, the representation of individuals is historically strings over a finite alphabet in GAs, real valued vectors in an evolution strategy, finite state machines in the classical EP, and tree structures in genetic programming (Eiben & Smith, 2003). All of them have been successfully applied to a wide range of problems, with a focus on optimization problems.

When an EA is applied to musical composition, there are three main considerations (Burton & Vladimirova, 1999), namely the search domain, the genetic representation, and the fitness evaluation. For musical composition, the search process is analogous to a combinatorial optimization problem, and because of the infinite combination of melodies, harmonies and rhythms, its search space is unlimited (Tokui & Iba, 2000). Therefore, the composition should be guided by some constraints. By artfully choosing our constraints, we aimed to maximize the musicality of Evac's output, as well as its real-time responsiveness. To this end, we limited our search to melodies of length 16 (i.e., strings of 16 single notes). Each note is encoded by 11 binary bits. All together this yields $(2^{11})^{16}$ possible combinations: still quite large, but no longer infinite.

Our goal when designing the representation was to be as simple as possible while maximizing the effectiveness of evolutionary search. Hence we adopted a discrete representation where an integer encodes several properties of a note, and an array of 16 integers represents an individual (musical phrase).

The last topic is fitness evaluation. Since we are generating impromptu accompaniment for the user's music, we incorporate the user's input into the fitness function. Specifically, the music represented by an individual in the GA is compared with the user's input and rated for similarity using music theoretic notions. To determine the individual's fitness, we used a function that assigned low fitness values to those individuals who were either too similar or too different. High fitness individuals were similar but not identical to the user's input. This ensured that the GA favored individuals that were related to the user's input without copying it.

The rest of the paper is structured as follows. The second section gives a brief review of the literature related to evolutionary computation and its application in music and art; the third section describes how the user interacts with Evac; the fourth section offers a detailed description of the Genetic Algorithm used in Evac including: individual representation of music notes, fitness function and genetic operators; the fifth section discusses the results of running and playing with Evac, along with analyses regarding the system; the sixth section presents conclusions about the performance of Evac, and the seventh section discusses future work.

2.3 BACKGROUND

Genetic algorithms have also been frequently used as an approach to music composition. In a previous study (Matic 2010), position based representation of rhythm and relative representation of pitches (based on distance from a starting pitch), were used to allow flexible encoding of music compositions. Use of a mathematical (i.e., non-interactive) fitness function as well as an initial population with pre-defined rhythm made the GA simpler to implement, and also improved the quality of the final result.

However, strictly non-interactive fitness functions preclude human influence on individuals' fitness. While in some cases this might be desirable, one of the primary goals of art is to give voice to human experience; as such, removing humans entirely from the loop of the fitness calculation represents a serious trade-off. Interactive evolutionary algorithms (IEAs) address that exact issue. Interactive evolutionary algorithms explicitly include the user in the evaluation of individuals, typically by allowing them to select the best individual(s) from a group or assign fitness values based on subjective appeal. In another paper on evolutionary music composition, Tokui and Iba (2000) combined genetic algorithms and genetic programming in an IEA. The GA individuals represented short pieces of rhythmic patterns, while the GP individuals expressed how these patterns were arranged in terms of their functions. Both populations were evolved interactively through user evaluation. The integration of interactive GA and GP has the benefit of allowing search for music structures in large search space.

Interactive evolutionary algorithms have also been used in generating artwork. For example, Graca & Machado used an IEA to generate assemblages (similar to collages) of 3D objects (2008). In their evolutionary art approach, users make the initial choice of source image and object library, then guide the evolutionary process in accordance to their artistic preference, until a desirable distribution of 3D objects is evolved. Several interesting points addressed by this system include: developing masks to allow exploration of details and ignore other regions; conveying different artistic notions such as motion; mimicking texture; and developing overall expressiveness. The limitations of this system include human fatigue, as well as the computational effort required to preview and render the individuals.

User interface design has also been the subject of interactive evolutionary algorithms (Masson, Demeure, & Calvary, 2010). In an evolutionary system called Magellan, the traditional model-based (task-based) approaches and the interactive genetic algorithms were combined to foster the exploration of the design space and inspire the designer. The input of the system was the given user task model, and the output was sketches of UI's, which could be later tuned by human designers. However, as with the previous example, the authors ran up against the human fatigue problem.

To address the issue of human fatigue, one approach that has been used is to hardcode mathematical heuristics of aesthetics. For example, in a proposed jewelry design system, several heuristics functions evaluating aesthetics and morphology were included, which reduced the amount of feedback needed from the user by two orders of magnitude (Wannarumon, Bohez, & Annanon, 2008). It was basically a hybrid approach in which evaluations rely partially on an encoded fitness function (the algorithmic aesthetics), and partially on subjective human feedback. However, this approach is still limited in its need for a hard-coded aesthetical evaluation function, which is something that may not be possible or practical for every situation.

In order to overcome both the human fatigue problem and the hard coded fitness function, Hornby and Bongard (2012) developed The Approximate User (TAU) system, in which a model of the user's preference was built and refined continuously during the search process. This preference model could then be used to drive the search algorithm, decreasing the demand placed on the user. Two variations of a user-modeling approach were compared to determine if this approach can accelerate IEA search. The first approach involved learning classification rules to determine which of two designs is better. The second involved learning a model to predict fitness scores. These two variants were compared against the basic IEA and it was shown that TAU was 2.7 times faster and 15 times more reliable at producing near optimal results.

2.4 EVAC AND THE USER

Evac is simple to operate – after opening the program, a very minimal user interface is displayed and the user begins to hear the tick of a metronome. From this moment, Evac is ready to accept user input. The user treats the Tab, Q, W, E, R, T, Y, U, I, O, P, open-bracket ('['), and close-bracket (']') keys on their computer keyboard like keys on a piano, pressing them to trigger the sound of a flute and releasing them to stop the sound. One note can be played at a time. Each key corresponds to a specific pitch, shown in Table 1.

Table 1: Keyboard keys and pitches.

Keyboard Key	Pitch
Tab	A2
Q	B2
W	C3
E	D3
R	E3
T	F3
Y	G3
U	A3
I	B3
O	C4
P	D4
[	E4
]	F4

After a certain amount of “prep” time, whether the user has played anything or not, Evac will begin playing the output of the genetic algorithm. (In fact, Evac will happily play along with silence forever - the fitness function, described below, has no problem with comparing against silence.)

Evac addresses the human fatigue problem in a novel way. The system does not require the user to evaluate music pieces directly, saving them the mental fatigue of making choice after choice (though the trade-off is that the user has no way to directly control the software’s performance). Evac, however, *does* constantly compare its output against the user’s input. As a result, the user is kept in the evaluative loop. In this way, Evac dodges the issue of user fatigue - the only human activity involved is playing music, which is quite enjoyable. As a matter of fact, our system can be used as a tool to assist musicians in developing creative accompaniment without much effort; all one needs to do is to keep playing.

2.5 THE GENETIC ALGORITHM APPROACH

2.5.1 Overview

A technical overview of Evac’s GA configuration is given in table 2. The major concern when designing Evac’s GA was the need to balance the competing demands of high performance with the limited resources available in a real-time environment. A clear microcosm of this issue presents itself in deciding an appropriate population size. Our population size of 100 was chosen to be small enough to prevent undue resource consumption: excessively large populations require more computation per generation, leading to more inertia and less responsiveness. Since Evac operates in real time, this would decrease the value of the system. On the other hand, small population size limits the diversity of the population. This means fewer musical ideas present in the population and a greater risk of stagnation.

Table 2: Technical overview of Evac's GA parameters.

Representation	Discrete. Integers represent pitch. Relative position of integers represents rhythm. (See below.)
Parent selection strategy	Tournament selection with 4 competitors.
Survival strategy	Generational + elitism. (Best individual retained between generations.)
Crossover	Uniform crossover; probability: 90%
Mutation	Uniform mutation; mutation probability per gene: 20%
Population size	100

We were reluctant to use "on-line" forms of parameter control (e.g., self-adaptation, dynamic parameter control) for two reasons. First, some forms of control pose additional challenge to the evolutionary process. Second, others depend on a priori reasoning that was not relevant to Evac's task. For example, a common technique is to use dynamic parameter control to decrease mutation rate as the algorithm progresses. Since Evac operates for as long as the user is entertained, this technique is not appropriate. With that in mind, parameter tuning (i.e., trial and error) was used to determine a reasonable balance between the two demands.

2.5.2 Representation

Figure 1 shows the representation for a single music note, and for an individual of the GA. As mentioned earlier, the representation for a single note was an 11-bit integer encoding of several values. Specifically, the lowest 6 bits were used to present the pitch of the note, which gave us a range of 64 different pitches (more than sufficient for the thirteen pitches we used). The next 4 bits were used to represent the velocity of the note (i.e., volume, or the strength of the note), which gave us a range of 16 different levels of velocity. If the velocity for a note was 0, that meant it was silent (i.e., a break or rest). The next bit marked whether this note was a new note or a continuation of the previous one. This bit was only meaningful if there were two consecutive notes with the same pitch. In that case, a new note means two notes will be played. If the second one is not a new note, it will not be played; instead, the first note will be played for the duration of two notes.

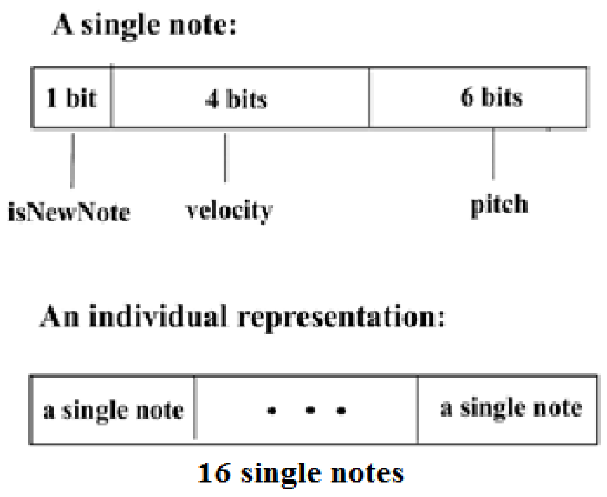
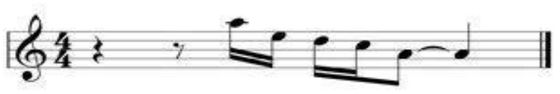


Figure 1: Representation for a single music note and an individual

However, in Evac's GA, an individual is not merely a note, but an entire musical phrase, i.e. a series of notes. We took each individual to be 16 notes long; as in music, these sixteen notes could be played over an arbitrary length of time, depending on Evac's settings. This allowed the user to determine for themselves whether they wanted to play a fast song or a slow song. An example of a musical phrase and the corresponding GA individual representation is shown in Figure 2.



Position	1-5	2	3	4	5	7-16
Velocity	0	1	1	1	1	1
Pitch	12	7	5	3	0	0
isNewNote	n/a	true	True	true	true	False

Figure 2: Musical phrase (above) and corresponding Evac individual representation (below).

2.5.3 Similarity Rating and Fitness Function

An individual's fitness was achieved by transforming a similarity function. The similarity function was computed by comparing each individual against the user's input for the previous 16 beats, and depended only on the harmonic "distance" (musical term: *interval*) between the pitches at each index. (In our representation, such harmonic distance can be computed by simple subtraction of the integer pitches.) Our similarity weighting function was informed by a musical technique called counterpoint, in which two separate melodies (strings of notes) interweave to form a larger texture. These two separate melodies are harmonically interdependent, but independent in pitch contour and rhythm. In particular, we used a simplistic approach where consonant, "at rest" intervals had a higher similarity rating, while dissonant, "unstable" intervals had a lower similarity rating. Generally for most people, a consonance sounds pleasant whereas dissonance sounds unpleasant or harsh. As Roger Kamien said in his book (2008), "An unstable tone combination is a dissonance; its tension demands an onward motion to a stable chord. Thus dissonant chords are 'active'; traditionally they have been considered harsh and have expressed pain, grief, and conflict." We want our music to be more pleasant to listen to, hence we gave consonant intervals higher similarity ratings, which will lead to higher fitness values in the evolutionary process.

Table 3 presents the 12 different musical intervals and their corresponding similarity values. We adopted these values based on music theory knowledge and our own musical intuition. If two notes have an interval that is higher than 12, their similarity value was calculated in this way: first, a similarity value was calculated based on Table 3 using the remainder of that interval value divided by 12, then the resulting similarity value was decreased by 0.2 to get the final similarity value. Rests have a similarity value of 1 to other rests, and a similarity value of 0 to non-rests.

The similarity value of an individual was the sum of the similarity values calculated for each of the 16 notes within that individual. Once a similarity value had been obtained for an individual, that similarity value was passed through a weighting function (i.e., the fitness function, shown in Figure 4) to calculate the individual's fitness value. The decision of weighting functions was based on the principle that individuals with excessively low or high similarity should receive low fitness values. Individuals with mid-to-high range similarity values should receive higher fitness values. Several fitness functions were tested and the one with the best performance among them, a quadratic function, was chosen.

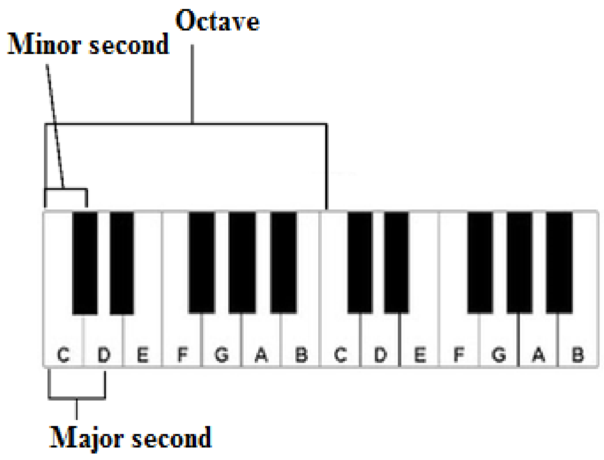


Figure 3: Music intervals shown on piano keyboard

Table 3: Musical intervals and their corresponding similarity values.

Interval : Musical Name	Similarity value
0 : Unison	1
1 : Minor second	0
2 : Major second	0.3
3 : Minor third	0.4
4 : Major third	0.6
5 : Perfect fourth	0.1
6 : Tritone	0
7 : Perfect fifth	0.8
8 : Minor sixth	0.6
9 : Major sixth	0.4
10 : Minor seventh	0.3
11 : Major seventh	0.1
12 : Octave	0.9

To provide a brief idea of how Table 3 works out in practice, assume we are interested in the individual: [0, 3, 5, …]. Furthermore, assume the first three notes of the user's input for the last 16 beats was [0, 10, 22, …]. To determine the similarity rating, simply calculate the absolute value of the differences between each note: [0 - 0, 10 - 3, 22 - 5, …] = [0, 7, 17, …], then compare to table 3. We can see distances of 0 (a unison - i.e., the same note) have a similarity value of 1, distances of 7 have a similarity value of 0.8. For the distances of 17, we first get the remainder of 17 divided by 12, which is 5, and from Table 3 we get distances of 5 have a similarity value of 0.4. We then decrease 0.4 by 0.2 (which is a static value got from experimentation) and get 0.2. Thus, for each interval we have similarities [1, 0.8, 0.2, …]. To obtain the similarity score for an individual, simply sum the similarities of each interval. Thus, assuming the individual in question consisted of only those three intervals listed explicitly above, our example would have a similarity value of 2. To obtain the individual's final fitness score, apply the fitness function shown in Figure 4: $f(2) = -(2 - 10)^2$, yielding -64.

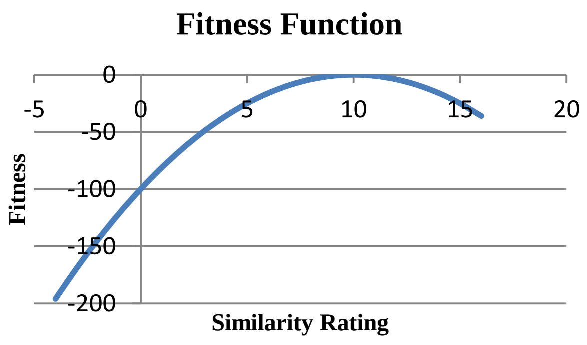


Figure 4: Fitness function: $f(x) = -(x-10)^2$. Input is a similarity rating.

2.5.4 Genetic Operators

For mutation we used uniform mutation. For each individual (i.e., series of notes), the algorithm goes through each one and decides whether that note will be mutated. Once a certain note is chosen to be mutated, a new random integer will be generated to replace that note. Since all three components (pitch, velocity, isNewNote) are encoded within this integer, we don't need to deal with the components separately. A high mutation rate (20%) was chosen to prevent the algorithm from converging on a simple parallel imitation of the user (i.e., shifted up some number of pitches, but otherwise identical).

For crossover, we used uniform crossover. Specifically, we did not do crossover between notes; instead, we performed crossover between different individuals. There are two reasons to adopt this strategy. First, it is simple. Second, and more importantly, we suspect that, in keeping with ideas like the Building Block Hypothesis, recombining effective individuals will allow us to more rapidly reach higher levels of quality.

2.6 RESULTS AND ANALYSES

Four excerpts of Evac sessions are available on the website https://www.dropbox.com/sh/h5cbz756tcyxy6n/vUXrcnmjKg , labeled "Evac Demo 1", "Evac Demo 2", "Evac Demo 3", and "Evac Demo 4." Again, the flute sound is played by a human user and the piano sound is played by the computer. What follows are subjective evaluations of Evac's performance; further research would involve more rigorous measures of quality (e.g., systematically conducted surveys regarding Evac's effectiveness, rating by experts on its degree of musicality and quality of accompaniment, etc.)

The first thing to be said about Evac is that it is surprisingly fun to use. We opted to use a flute as the voice for the user's input, and a piano as the voice for the program. The result was both musical and entertaining. While Evac does not provide hours of entertainment, there are just enough moments of unexpected beauty to keep one engaged for half an hour or an hour. It is particularly exhilarating when, instead of simply complementing the user's phrases, the evolution yields phrases that seem to demonstrate some initiative of their own.

On a similar note, the major goal of implicit interactivity was to minimize the amount of user fatigue suffered while controlling the algorithm. In our experience, this was a total success. Working with Evac felt much more like playing a game than interacting with an algorithm. While we mentioned that Evac could keep one engaged for something on the order of an hour, we suspect, if there were good reason, people would be able to continue significantly longer than that.

We did notice that it takes Evac some time to "catch up" when the user changes musical directions. This is to be expected - the population had optimized for a certain fitness function, and when the user changes their behavior, the algorithm needs time to adapt to the new fitness landscape. Nor, all told, is the time it takes to adapt excessive. Typically within one or two cycles of 16 beats, it has found its way into something that is at least not offensive, if not truly complementary.

Evac also seems to yield a better experience with slower songs. To some degree, this can be attributed to the processing demands - for slower songs, fewer computations need to be performed per second, and as a result the program responds more smoothly and immediately to user input. At the same time, this is also a function of the latency present in the system. Even at modest speeds, there is still appreciable (if not crippling) delay between when the user presses a key and when the corresponding flute sound plays. Slower tempos mean the latency introduces relatively less error, thereby making the delay less intrusive. This latency also limits the user's play style, to some extent. Because of the latency, it is difficult to perform quick, intricate movements while keeping time with the rest of the system. As such, it is simpler and easier for users to play mostly long, sustained notes with some flourishes, either at the beginning or the end. Highly rhythmic melodies are essentially a non-option because of this.

Though latency introduces some difficulties, perhaps Evac's greatest weakness is its utter lack of phrasing. This means that, while Evac almost always sound good with what the user plays, it is difficult to let Evac "take the lead." This is not ideal both because the user must always be actively determining where the song will go, and because if the user runs out of ideas, the session effectively comes to an end. Of course, this is not surprising; Evac's fitness function only takes into account the "harmoniousness" of an individual melody in relation to what the user has done, so there is no reason to expect that it would demonstrate phrasing behavior.

2.7 CONCLUSION

Evac is excellent for what it is - a first approximation that demonstrates the power of implicit interactivity. Evac demonstrates satisfying performance on musical improvisation with the user. Evac's ability to run in real time allows the user to experience the same kind of exploration that happens in real life improvisation scenarios with other musicians.

On one hand Evac is able to follow the music that the user plays; on the other hand, Evac is more than a simple reharmonizer. It never copies the user, nor does it repeat its own previous melody. When the user stops playing (while keeping the program running), Evac will also slow down gradually, but it will never completely stop playing – it will play few notes every now and then, as if it is asking and waiting for the user to respond. Once the user starts playing again, Evac will re-start the cooperation with the user, with very short amount of time needed at the beginning to adapt to the user's music style.

2.8 FUTURE WORK

The most immediate need is for reduced latency. If Evac responded immediately and effortlessly to user input, we anticipate using it would become even more fun. This in turn would improve the extent to which it fulfilled its original purpose: eliminating user fatigue. There are also some small audio rendering issues (particularly at the end of notes) that could use fixing. These small improvements, together with a strong graphical user interface, would make Evac worthy of public distribution.

If its other operations could be suitably optimized, the next greatest need is for more sophisticated parameter control. Parameter tuning has many downsides, and there is a reasonable chance that well-designed parameter control could reduce or eliminate the issue with Evac not taking the lead (e.g., by increasing mutation rate and number of notes played when user isn't playing much, etc.). Furthermore, sophisticated parameter control could throttle back the GA's resource usage in the event that other, non-Evac processes begin demanding CPU time.

There is also an entire branch of research that we have yet to exploit, despite its great potential relevance: evolutionary computation in dynamic environments. Techniques like storing good solutions to be used in case their environment returns, or using random immigrants to help adapt to a new fitness landscape could be easily applied to our problem domain, since our fitness function changes whenever the user moves in a new musical direction. Using these kinds of strategies would yield serious improvements in Evac's ability to simulate a real human improvisation partner, since a key part of musical improvisation with other people is recognizing song sections which can be returned to or modified.

Finally, the fitness function has several weaknesses that could be addressed in future research. Most obvious is that both the fitness function and the similarity function are hardcoded, and thus rely on human expertise. Minimizing the human involvement in these features would improve Evac’s ability to participate in improvisation across different musical traditions and with different individuals. Such an innovation might also be engineered to address the issue with phrasing, or to other musical problems like how to coordinate the actions of more than one automated instrument.

Once Evac or its peers are advanced enough, one might even connect them to one another, and see what kind of music machines would make just for themselves.

2.9 REFERENCES

[1] Burton, A. R., & Vladimirova, T. 1999. Application of Genetic Techniques to Musical Composition. *Computer Music Journal , 23*.

[2] Eiben, A. E., & Smith, J. E. 2003. *Introduction to Evolutionary Computing.*

[3] Graca, F. d., & Machado, P. 2008. Evolving Assemblages of 3D Objects. In *Applications of Evolutionary Computing* (pp. 453-462).

[4] Hornby, G. S., & Bongard, J. C. 2012. Accelerating Human-Computer Collaborative Search through Learning Comparative and Predictive User Models. *GECCO* .

[5] Masson, D., Demeure, A., & Calvary, G. 2010. Magellan, an Evolutionary System to Foster User Interface Design Creativity. *EICS* .

[6] Matic, D. 2010. A Genetic Algorithm for Composing Music . *Yugoslav Journal of Operations Research* , 157-177.

[7] Tokui, N., & Iba, H. 2000. Music Composition with Interactive Evolutionary Computation. *Proceedings of the 3rd international conference on generative art* , 215-226.

[8] Wannarumon, S., Bohez, E. L., & Annanon, K. 2008. Aesthetic Evolutionary Algorithm for Fractal-Based User-Centered Jewelry Design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* , 19-39.

[9] Kamien, Roger 2008. *Music: An Appreciation, 6th Brief Edition*, p.41. ISBN 978-0-07-340134-8

[10] Roger B. Dannenberg 1984. An On-Line Algorithm for Real-Time Accompaniment. *International Computer Conference,* 193-198

[11] Roger B. Dannenberg, Thom, Belinda, & Watson, David 1997. A Machine Learning Approach to Musical Style Recognition. *Computer Science Department.* Paper 504

http://repository.cmu.edu/compsci/504

[12] Bishop, C. M. 1995. *Neural Networks for Pattern Recognition.* Clarendon Press.

[13] Rowe, R , 1993. *Interactive Music Systems*. MIT Press.

[14] Rubine, D. 1991. The automatic recognition of gestures. Tech. rep., School of Computer Science, Carnegie Mellon University

[15] Adam Linson, Chris Dobbyn, & Robin Laney, 2012. Critical Issues in Evaluating Freely Improvising Interactive Music Systems. *International Conference on Computational Creativity.*

[16] Clarke, E. 2005. Creativity in Performance. *Musicae Scientiae* 9:1, 157-182

[17] Clarke, E. 2005. *Ways of Listening: An Ecological Approach to the Perception of Musical Meaning.* Oxford University Press

[18] Collin, N. 2008. The Analysis of Generative Music Programs. *Organised Sound*, No. 3: 237-248

[19] Lehmann, A., R. Kopiez, 2010.The Difficulty of Discerning between Composed and Improvised Music. *Musicae Scientiae* 14: 113-129

[20] MacDonald, R., G. Wilson, & D. Miell. 2011. Improvisation as a Creative Process within Contemporary Music. *Musical Imaginations: Multidisciplinary Perspectives on Creativity, Performance and Perception,* D. Hargeaves, D. Miell, and R. Macdonald, Eds. Oxford University Press

[21] Pressing, J. 1987 The Micro- and Macrostructural Design of Improvised Music. *Music Perception* 5(2): 133-172

[22] Pearce, M. T., & G. A. Wiggins, 2001. Towards a Framework for the Evaluation of Machine Compositions. *Proc. of the AISB'01 Sym. On Artificial Intelligence and Creativity in the Arts and Sciences.* Brighton: 22-32

[23] Stowell, D., A. Robertson, N. Bryan-Kinns, &M. D. Plumbley. 2009. Evaluation of Live Human Computer Music Making: Quantitative and Qualitative Approaches. *Int. J. of Human Computer Studies* 67, No. 11: 960-975.

[24] Smith, H. &R. T. Dean. 1997. *Improvisation Hypermedia and the Arts since 1945.* Routledge.

CHAPTER 3

EVOLMUSIC – A PREFERENCE LEARNING ACCOMPANIST[2]

[2] Shu Z, Roi C, Khaled Rasheed To be submitted to *International Center for Academic Integrity 2014*

3.1 ABSTRACT

This project is an extension of a previous work “Evac: An Evolutionary Accompanist (Zhang & Bailey, 2013)”, in which a computer improvisational accompaniment system, Evac, was built and described. In spite of the fact that the EvolMusic and Evac both involve genetic algorithms (GA), they are designed to address different issues and hence possess different characteristics.

The two main differences are as follows. Firstly, Evac uses an implicitly interactive genetic algorithm, which has a hardcoded mathematical fitness function and allows the user’s music input to influence Evac’s response without the user’s explicit inference, while EvolMusic allows direct control from the user over the accompaniment using machine learning techniques to learn a fitness function from the user’s preferences. Secondly, Evac runs in real time, i.e., the accompaniment is generated and played online, while EvolMusic records a piece of the user’s musical input, generates four different accompaniments, lets the user vote for his or her favorite, adjusts the GA’s fitness function, and then generates new accompaniments which can be further used to learn the user’s preferences.

3.2 BACKGROUND

The increasing popularity of interactive music systems in computer music have led to extensive research nowadays. In his book "Interactive Music Systems – Machine Listening and Composing," Robert Rowe defined interactive computer music systems as computer systems whose output is modified in response to their musical input. In the book "Composing Interactive Music: Techniques and Ideas Using Max", Todd Winkler defined interactive music as "a music composition or improvisation where software interprets a live performance to affect music generated or modified by computers" (Winkler, 2001). The systems affect the output music by altering musical parameters such as pitch, tempo, rhythm, orchestration and so on. Modeling of human hearing, understanding and response are involved in the interactive systems (Rowe, 1993).

Rowe also identified categories for interactive computer music systems. The first way to classify interactive systems is based on whether they are 'score-driven' or 'performance-driven' (Rowe, 1993). Score-driven interactive systems involve music alignment, that is, the computer being able to detect and match the music input, based on stored event sequences or pieces of music scores; whereas performance-driven music systems do not expect input music that matches any stored files, or has any known patterns.

The second way to classify interactive systems is based on how the systems produce new output music – there are transformative, generative, and sequenced interactive systems. Transformative music systems take any form of music input and transform them into new pieces of music to output; the transformation can be minor changes or changes that cause the resulting music completely different from the input music. Generative computer system, on the other hand, generate music base on musical rules. They don't take any music input, though they may use some raw music elements as source material. Sequenced computer music systems can detect real-time music input and choose to output their stored music pieces based on the events they detected.

The third way to classify computer music systems is to distinguish them as player or instrument paradigms (Rowe, 1993), i.e., some systems aim at constructing an artificial music performer, while others aim at behaving as a musical instrument themselves. For example, a player paradigm system played by a single human performer would behave like a duet, whereas an instrument paradigm system played by a single human would behave like a soloist. Instrument paradigm systems are interactive in that they take the performer's musical input and generate their own musical output. For example, consider a system that mimics a guitar, but is "played" by singing.

According to Todd Winkler (2001), there are generally five steps in the interactive process: human performer or musical instruments input; computer listening; software interpretation of the computer listener and detecting useful data for composition; computer composition based on the interpretation from last step; and finally sound output by hardware.

In the literature, real-time music accompaniment applications have adopted different approaches (Dannenberg, 1984) ranging from conventional taped accompaniment (one does not follow user input, i.e., one that is unable to synchronize the user's input with the accompaniment the computer plays back), to interactive improvisation involving algorithms such as evolutionary computing algorithms. Roger B. Dannenberg at Carnegie-Mellon University published a series of papers in which he proposed and developed computer music systems that can listen to a live monophonic performer, align his or her input to a stored music score, and then play back pre-composed accompaniment simultaneously.

In the paper “An On-Line Algorithm for Real-Time Accompaniment” published in 1984, he proposed his computer accompaniment system for the first time. He noted that there are three sub-processes involved, namely taking input from a live soloist and processing it, aligning the processed input with a stored musical score, and determining the timing signals used to trigger the correct accompaniment from a pre-composed score. Since the live performer is not immune to making mistakes while performing, and the computer’s input detection is not perfect, the accompaniment system must maintain a certain level of tolerance in matching the input with the stored score.

Specifically, the way the system was implemented was based on modeling the user’s input and the stored score as two sequences (or streams) of events. Then the problem was boiled down to finding the “best” match between these two sequences. There are many ways to seek the ‘best’ match, and the one Dannenberg adopted was rather intuitive – to find the ‘longest common subsequences’ of the two sequences (Dannenberg, 1984). The concept of a “virtual clock” was used to generate the timing information to play the accompaniment based on the matching. Two experimental systems were constructed: one with an AGO keyboard for input and the other one with trumpet input.

Limitations of these systems included (Dannenberg, 1984): it was only able to adjust tempos on a small scale rather than attempt to adjust tempos in a certain musical manner; it was only able to adjust the accompaniment based on the tempo information, ignoring all the other cues such as articulation, loudness and so on; and last but not least, it assumed that the input is an ordered set of events, thus it did not allow matching to sets of unordered or simultaneous events. Despite the limitations, the system was able to follow and accompany a live performer and had a desirable level of tolerance for different tempos, extra notes and omitted notes.

To overcome the shortcomings inherent in a mathematical approach to music composing or music understanding, we can adopt machine learning techniques. Machine learning can use multi-dimensional training data and take into account a large number of low-level features such as pitch and tempo, thus building effective classifiers which are able to undertake higher-level music processing.

In the paper "A Machine Learning Approach to Music Style Recognition" (Dannenberg, 1997), three types of machine learning classification techniques, namely naïve Bayesian classification, linear classification, and neural networks, were used to classify an improvisation as certain music styles (Rubine, 1991). For each type of classification, two sets of experiments were carried out. The first experiment was to classify the piece of music into four styles: "Frantic," "lyrical," "syncopated," or "pointillistic". In the second sets of experiments, four additional musical styles were added for consideration: "quote", "blue", "low," and "high." Training data were labeled for use in supervised training, and 13 lower-level features were identified based on the MIDI data. Cross-validation was used for the training. Confidence measures were used to reduce false positives.

Results showed that all three types of classifiers can classify music with a high accuracy rate, ranging from 77% to 99%. The experiment with four target features showed higher accuracy rate. Neural networks took hours to train while the other two types of classifiers took minutes (Bishop, 1995). The technique developed in the paper was believed to have applications in the field of music composition, understanding and performance.

Evaluation is another important step to consider in computer music generation, though it was omitted for EvolMusic. In the paper "Critical Issues in Evaluating Freely Improvising Interactive Music Systems" (Linson 2012), several approaches to evaluating interactive computer music systems were reviewed, with their strengths and weaknesses analyzed and compared.

Specifically, it discussed the advantages and disadvantages of computer based quantitative evaluation (Pressing, 1987), and also concluded that human expertise, which conducts qualitative evaluation, is more appropriate and in some circumstances, indispensable (Pearce and Wiggins, 2001). The paper pointed out that quantitative evaluation has advantages in evaluating music in which the melody and/or the rhythms are constrained by a set of 'well-defined style-based rules'.

On the other hand, for music like freely improvised music, the quantitative evaluation loses its advantages because freely improvised music differs in evaluation criteria (Lehmann and Kopiez, 2010). In other words, definable musical rules are not enough to describe or to judge freely improvised music, because of the spontaneous characteristics and real-time negotiating interactions of this music style (MacDonald, et al, 2011). Other shortcomings of quantitative analysis include that it cannot decide which musical features from a performance have the most significance in the analysis process. Also, it is not good at detecting large scale structures within a music piece, nor does it take into account the listening context or the subjectivity of the audience (Clarke, 2005).

In contrast, human experts don't have as many inherent problems in these aspects (Smith, 1997), although it is conventionally thought to be lack of scientific rigor. For example, human experts are able to avoid measuring merely the compliance to musical rules.

That said, the paper still noted that no evaluation methods should be used to replace the others for all the music systems, and it is important to find an appropriate and flexible evaluation method for a specific music system (Stowell, 2009). For example we can collect listeners' opinions through a survey, if we have a population of listeners available; or focus on the underlying software to find more results in answer to our research questions (Collins, 2008). In conclusion, the author said that there has been no ‘well-established evaluation method that is widely recognized in the literature’ for interactive computer music systems.

For this paper, no formalized evaluation system was built due to the limitation of time and resources. In spite of that, one can keep playing and listening to the responsive real-time accompaniment, maybe in different styles, and have some idea of the quality of the resulting music pieces.

3.3 USER INTERFACE OF EVOLMUSIC

The user interface of EvolMusic is simple and clear. It has nine buttons in total and the layout is intuitive: one bigger button to let the user input music, four buttons assigned to play each of the four generated accompaniments, and four other buttons to let the user vote for the respective accompaniments. On startup, the interface shows the greeting words at the message area, the top right of window, and only the button for inputting music is clickable, with all the other eight buttons greyed out (as shown in figure 1).

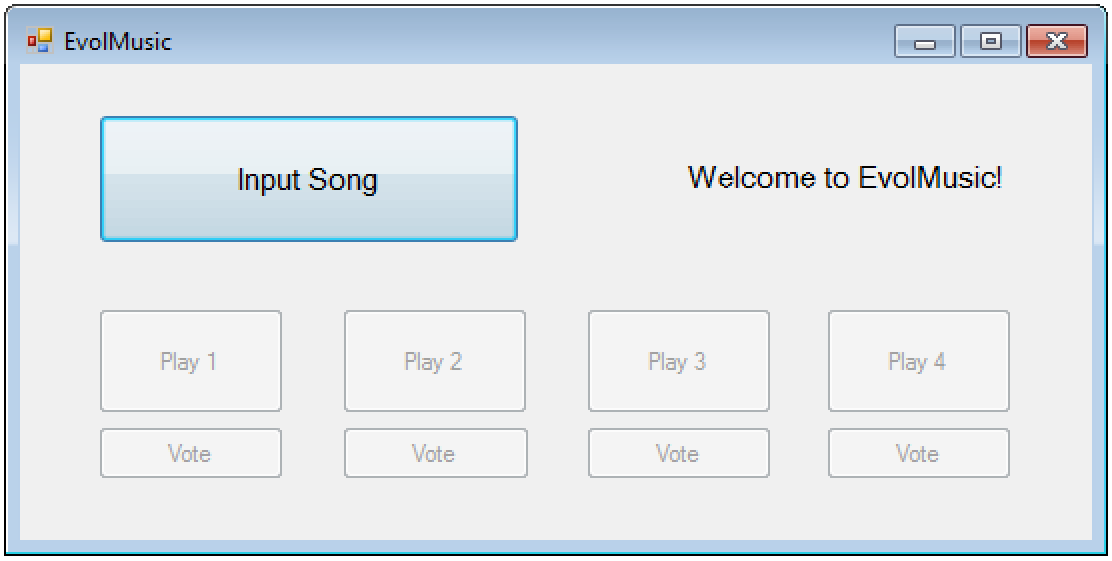
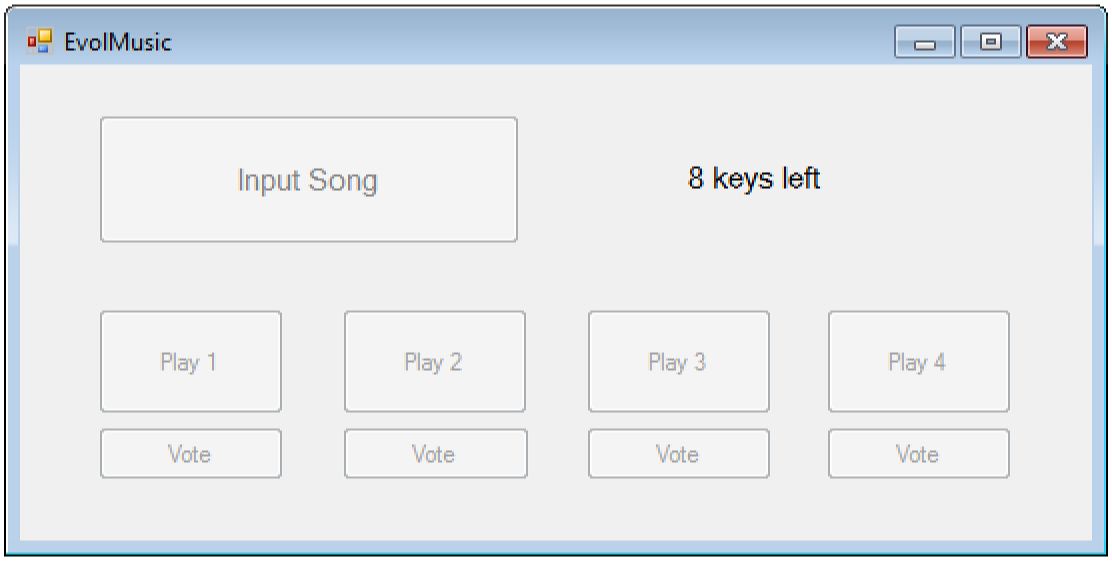


Figure 5: Startup window of EvolMusic

Once the user starts inputting music by typing on the keyboard, EvolMusic starts counting notes the user has input, and correspondingly the number of notes left for the user to input is shown in the message area of the window. At this stage, all the buttons on the window are greyed out (as shown in figure 2), to suggest the user to complete the music input before any further action.



8 keys left

Figure 6: EvolMusic interface when user inputs music

After the user has played 12 notes, the window shows the message “Generating songs!” (as shown in figure 3) while EvolMusic works on generating four pieces of accompaniment. All the buttons are still greyed out at this point.

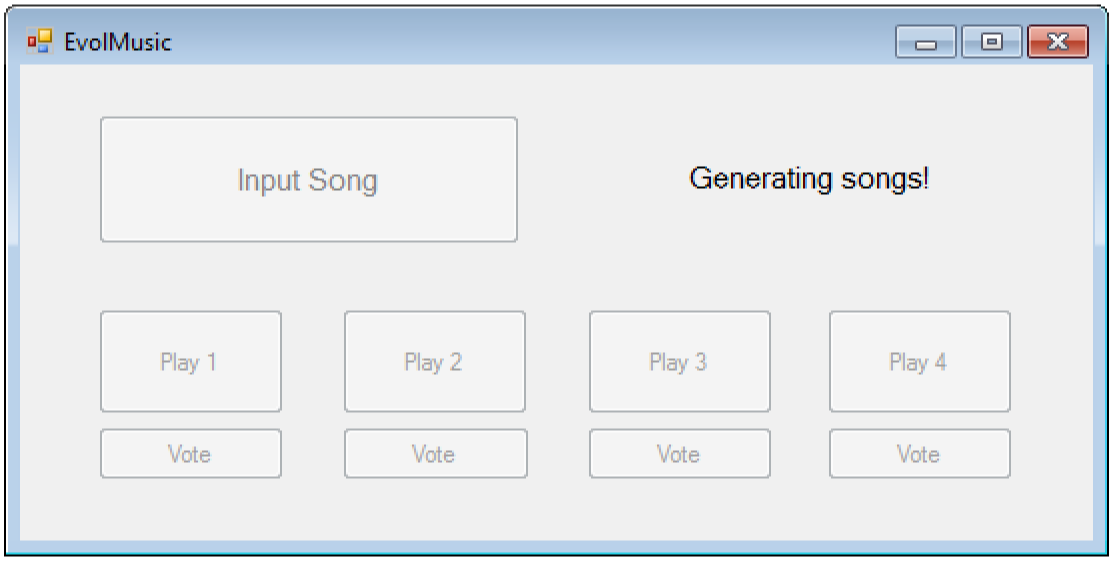


Figure 7: EvolMusic interface when new accompaniments being evolved

Once EvolMusic has completed generating all accompaniments, all the buttons become clickable to indicate the accompaniments, along with recorded user's input music, are ready to play and be voted on (as shown in figure 4).

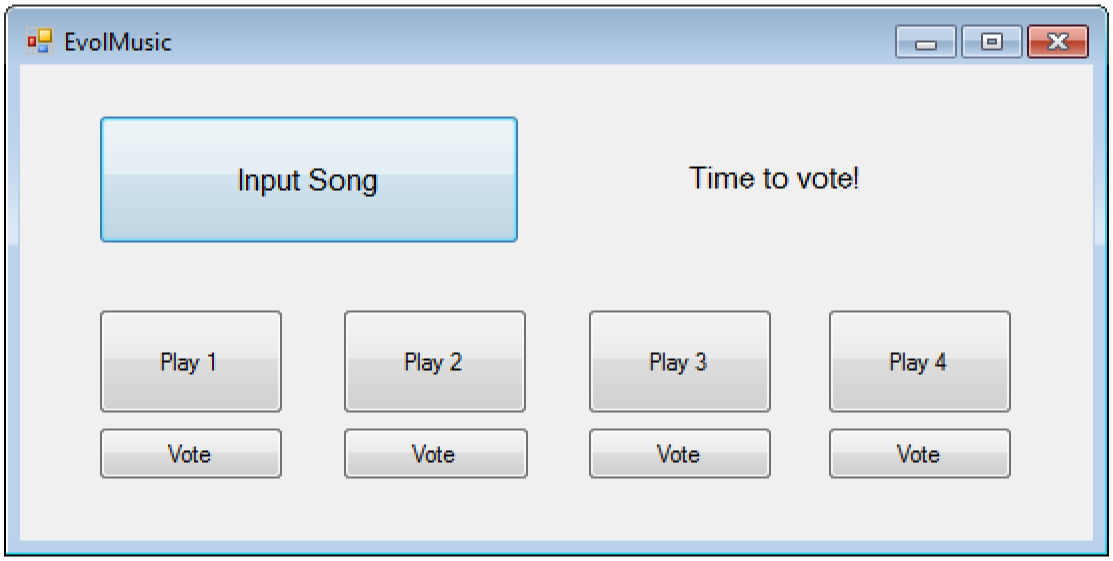


Figure 8: EvolMusic interface when it's ready to let user play the generated accompaniments and vote

3.4 SYSTEM DESIGN

EvolMusic was developed based on Evac (Shu & Bailey, 2013), and the main frame of the GA remains unchanged. The main part that has been changed is the fitness function. In the following two sections, first we will discuss in short about the overall configuration for the GA being used here (for more detail descriptions please refer to the paper "Evac: An Evolutionary Accompanist"); then we will go into more detail about the hill climbing technique used in fitness function learning.

3.4.1 Genetic Algorithms

As with Evac, the GA individual representation used in EvolMusic was an array of 16 single notes. Each single note was an 11-bit integer encoding of several values (as shown in figure 5). Tournament selection was used as the parent selection strategy. A generational survival strategy with elitism was used. Uniform crossover with 90% probability and uniform mutation with mutation rate of 20% per gene were used. The population size was chosen to be 100, which was small enough to prevent excessive computing time, and large enough to maintain a certain level of diversity in the GA individuals.

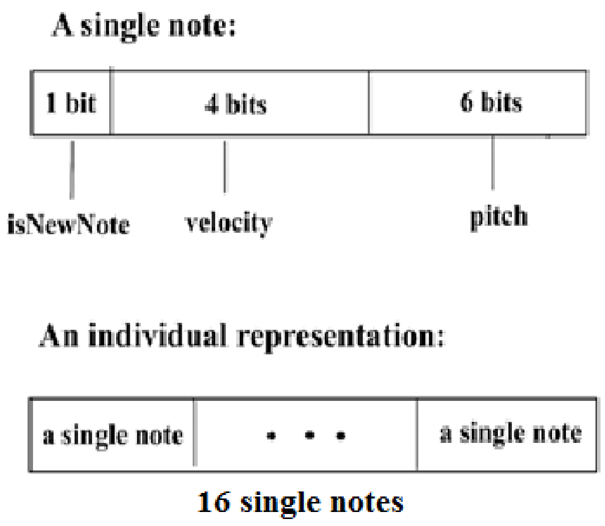


Figure 9: Representation for a single music note and an individual

3.4.2 Fitness Function Learning

In Evac, the fitness of an individual in the GA was achieved by transforming a similarity function. The similarity function was computed by comparing each individual against the user's input for the previous 16 single notes. Depending on the harmonic "distance", a mathematical value was assigned to each of the 12 different musical intervals that Evac allows the user to input. The similarity value of an individual was the sum of the similarity values calculated for each of the 16 single notes within that individual, and the fitness value for that individual was determined by passing its similarity value through a quadratic function. The quadratic function was set so that individuals with excessively low or high similarities received lower fitness values, whereas those with mid-to-high range similarity values received higher fitness values.

However, with EvolMusic, the quadratic weighting function was omitted, and the "similarity pairing values" were adjustable based on human evaluation (user voting) instead of being hardcoded. It should be pointed out that here since the similarity pairing values directly reflected the user's preference, the sum of the similarity paring values does not need to be passed through any weighting function; rather, the sum of the similarity pairing values can serve directly as fitness values for an individual. To reduce the size of the search space for the fitness function, instead of dealing with similarity values for all 12 intervals, the intervals were hand-grouped into four sets: high, medium-high, medium-low, and low similarity (as shown in table 1).

Table 4: Interval categories

Interval : Musical Name	Similarity value
0 : Unison	High
1 : Minor second	Low
2 : Major second	Low
3 : Minor third	Medium-high

4 : Major third	Medium-high
5 : Perfect fourth	High
6 : Tritone	Low
7 : Perfect fifth	High
8 : Minor sixth	Medium-high
9 : Major sixth	Medium-high
10 : Minor seventh	Medium-low
11 : Major seventh	Medium-low
12 : Octave	High

During the hill climbing technique to learn similarity values, there were four alternate model manipulations: higher similarity values between notes, lower similarity values between notes, increased relative distance between notes, and similarity values between notes that were unchanged. Specifically, the way the first three models tune the similarity values are shown in table 2. γ was set to 0.9 originally, and tuned over time, to accumulatively slow down the manipulation speed. The similarity values were controlled so that they were always between 0 and 1. As can been seen from the mathematical functions, "Model_up" and "Model_down" do not adjust similarity values linearly – the size of the adjustments gets smaller as the similarity values grow (Model_up) or shrink (Model_down). While Model_up and Model_down "squash" the similarity values, the third model spreads all the similarity values, where $\min_d$ in the equation was either the difference between 1 and the maximum similarity value, or the minimum similarity value, depending on which one is smaller.

Table 5: Implementation of the three model manipulations

	Implementation functions
Model_up	$similarityValue_k = similarityValue_k + \gamma * 0.5 * (1 - similarityValue_k)$
Model_down	$similarityValue_k = similarityValue_k - \gamma * 0.5 * similarityValue_k$
Model_spread	$similarityValue_k = 0.5 * similarityValue_k \pm 0.5 * \gamma * min_d * (\frac{\lvert similarityValue_k - \overline{similarityValue}\rvert}{similarityValue_{max} - \overline{similarityValue}})$

3.5 RESULTS AND CONCLUSION

First of all, the UI design of EvolMusic made it very handy and intuitive to use. Second, the choice of input 12 notes at a time was successful – it allowed the user to input music that was long enough for a human being to sense and judge, and short enough to avoid human fatigue of listening and evaluating, as well as to prevent memory loss when the user compares and votes. Third, the computing time of generating new accompaniments were not unbearably long – it generally took less than 3 seconds.

More important, EvolMusic achieved its main goal successfully – to be able to generate different accompaniments and learn human preferences based on user feedback. The four accompaniments were all reasonably good – which was to some extent a merit inherited from Evac. The user was able to sense the difference between these accompaniments and make a choice, at least with multiple trials of listening and comparing, which was well supported by EvolMusic.

However, the fact that there were only four models of adjusting similarity values made the learning a bit crude. There were two reasons behind using only four models: first, more manipulating models means more computing resources will be consumed; second, too many accompaniment choices will add more burden and confusion to the user, since he or she would have to listen to and compare all the alternatives.

3.6 FUTURE WORK

Despite the positive results of EvolMusic, there are a few things that might be improved. First of all, EvolMusic was not able to incorporate the Evac's real time performance. After the user votes for his or her favorite accompaniment, and EvolMusic digests that information and generates new accompaniments, the user would have to repeat the same procedure again – listening to four alternative accompaniments and voting. In other words, EvolMusic was only able to learn the user's preference, but not to put that knowledge to use in accompaniment. This problem will have to be solved if we want EvolMusic to be practically useful.

Second, though with careful listening, a user can find the differences between the alternative accompaniments, maybe it is a good idea to make the differences more obvious.

Third, as with Evac, EvolMusic was also just a prototype to convey the author's initial idea. As a result, these two pieces of software are not to their highest potential. Many details need to be improved in order to make them more successful. For example, a button to tune the volume of the accompaniment relative to the user's input music seems necessary for the software UI; permitting the user to choose the number of notes in the input, as well as more timbre choices are desirable. Finally, more than one accompaniment instrument should be considered to enhance the performance of the software.

3.7 REFERENCES

[1] Roger B. Dannenberg 1984. An On-Line Algorithm for Real-Time Accompaniment. International Computer Conference, 193-198

[2] Roger B. Dannenberg, Thom, Belinda, & Watson, David 1997. A Machine Learning Approach to Musical Style Recognition. Computer Science Department. Paper 504 http://repository.cmu.edu/compsci/504

[3] Bishop, C. M. 1995. Neural Networks for Pattern Recognition. Clarendon Press.

[4] Rowe, R., 1993. Interactive Music Systems. MIT Press.

[5] Rubine, D. 1991. The automatic recognition of gestures. Tech. rep., School of Computer Science, Carnegie Mellon University

[6] Adam Linson, Chris Dobbyn, & Robin Laney, 2012. Critical Issues in Evaluating Freely Improvising Interactive Music Systems. International Conference on Computational Creativity.

[7] Clarke, E. 2005. Creativity in Performance. Musicae Scientiae 9:1, 157-182

[8] Clarke, E. 2005. Ways of Listening: An Ecological Approach to the Perception of Musical Meaning. Oxford University Press

[9] Collin, N. 2008. The Analysis of Generative Music Programs. Organised Sound, No. 3: 237-248

[10] Lehmann, A., R. Kopiez, 2010.The Difficulty of Discerning between Composed and Improvised Music. Musicae Scientiae 14: 113-129

[11] MacDonald, R., G. Wilson, & D. Miell. 2011. Improvisation as a Creative Process within Contemporary Music. Musical Imaginations: Multidisciplinary Perspectives on Creativity, Performance and Perception, D. Hargeaves, D. Miell, and R. Macdonald, Eds. Oxford University Press

[12] Pressing, J. 1987 The Micro- and Macrostructural Design of Improvised Music. Music Perception 5(2): 133-172

[13] Pearce, M. T., & G. A. Wiggins, 2001. Towards a Framework for the Evaluation of Machine Compositions. Proc. of the AISB'01 Sym. On Artificial Intelligence and Creativity in the Arts and Sciences. Brighton: 22-32

[14] Stowell, D., A. Robertson, N. Bryan-Kinns, &M. D. Plumbley. 2009. Evaluation of Live Human Computer Music Making: Quantitative and Qualitative Approaches. Int. J. of Human Computer Studies 67, No. 11: 960-975.

[15] Smith, H. &R. T. Dean. 1997. Improvisation Hypermedia and the Arts since 1945. Routledge.

[16] Shu Zhang & C. Thomas Bailey. 2013 Evac: An Evolutionary Accompanist

CHAPTER 4

SUMMARY AND CONCLUSIONS

The goal of this research was to apply evolutionary algorithm and machine learning techniques to the interactive music composition area. Two music accompaniment systems that address different issues were proposed in this research. Implementation details, achievements and drawbacks of these two systems were discussed.

The first system proposed, Evac, uses a novel, implicitly interactive genetic algorithm which allows the user's musical input to influence the generated accompaniment without the need for explicit rating of the individuals of the genetic algorithm. In contrast to many pieces of software in the world of evolutionary music and art, Evac runs in real time, allowing the user to experience the same kind of exploration that happens in real life improvisation scenarios with other musicians. Evac demonstrates satisfying performance on musical improvisation with the user, though it shows several disadvantages such as latency. Another drawback in Evac is the hardcoded fitness function in the genetic algorithm, which leads to the research in the second system – EvolMusic.

EvolMusic does not use hardcoded mathematical fitness function and implicit genetic algorithm; instead, it allows direct control from the user over the accompaniment using machine learning techniques to learn a fitness function from the user's preferences. EvolMusic does not run in real time; instead, it records a piece of the user's musical input and generates four different accompaniments, lets the user vote for his or her favorite, adjusts the GA's fitness function, and generates new accompaniments which can be further used to learn the user's preference.

In the future, s possible extension is to combine the functionality of the two systems, so that the resulting system will be able to both run in real time and learn the user’s preference. If a system is able to learn the user’s preference, and put that knowledge to use in accompaniment, then it will be practically more useful.

Another area worthy of exploration is to develop accompaniments of more obviously different music styles. Now though, with careful listening, a user can find the differences of music in both the changing patterns in the generated accompaniment in Evac and the alternative accompaniments offered to the user to choose in EvolMusic. It may be a good idea to make the changes and differences more obvious.

Furthermore, the system can be improved in several minor aspects to make it more practically desirable. For example, different accompaniment instruments can be considered to enhance the performance of the software, or several instruments playing together to form a more sophisticated accompaniment. If different instruments are incorporated, then volume control for each instrument is a reasonable design variable. More keys to allow the user to input music should be allowed in future system design. Finally, a better user interface may be needed.

CHAPTER 5

BIBLIOGRAPHY

Burton, A. R., & Vladimirova, T. 1999. Application of Genetic Techniques to Musical Composition. *Computer Music Journal , 23*.

Eiben, A. E., & Smith, J. E. 2003. *Introduction to Evolutionary Computing.*

Graca, F. d., & Machado, P. 2008. Evolving Assemblages of 3D Objects. In *Applications of Evolutionary Computing* (pp. 453-462).

Hornby, G. S., & Bongard, J. C. 2012. Accelerating Human-Computer Collaborative Search through Learning Comparative and Predictive User Models. *GECCO* .

Masson, D., Demeure, A., & Calvary, G. 2010. Magellan, an Evolutionary System to Foster User Interface Design Creativity. *EICS* .

Matic, D. 2010. A Genetic Algorithm for Composing Music . *Yugoslav Journal of Operations Research* , 157-177.

Tokui, N., & Iba, H. 2000. Music Composition with Interactive Evolutionary Computation. *Proceedings of the 3rd international conference on generative art* , 215-226.

Wannarumon, S., Bohez, E. L., & Annanon, K. 2008. Aesthetic Evolutionary Algorithm for Fractal-Based User-Centered Jewelry Design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* , 19-39.

Kamien, Roger 2008. *Music: An Appreciation, 6th Brief Edition,* p.41. ISBN 978-0-07-340134-8

Roger B. Dannenberg 1984. An On-Line Algorithm for Real-Time Accompaniment. *International Computer Conference,* 193-198

Roger B. Dannenberg, Thom, Belinda, & Watson, David 1997. A Machine Learning Approach to Musical Style Recognition. *Computer Science Department.* Paper 504

http://repository.cmu.edu/compsci/504

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition.* Clarendon Press.

Rowe, R , 1993. *Interactive Music Systems*. MIT Press.

Rubine, D. 1991. The automatic recognition of gestures. Tech. rep., School of Computer Science, Carnegie Mellon University

Adam Linson, Chris Dobbyn, & Robin Laney, 2012. Critical Issues in Evaluating Freely Improvising Interactive Music Systems. *International Conference on Computational Creativity.*

Clarke, E. 2005. Creativity in Performance. *Musicae Scientiae* 9:1, 157-182

Clarke, E. 2005. *Ways of Listening: An Ecological Approach to the Perception of Musical Meaning.* Oxford University Press

Collin, N. 2008. The Analysis of Generative Music Programs. *Organised Sound,* No. 3: 237-248

Lehmann, A., R. Kopiez, 2010.The Difficulty of Discerning between Composed and Improvised Music. *Musicae Scientiae* 14: 113-129

MacDonald, R., G. Wilson, & D. Miell. 2011. Improvisation as a Creative Process within Contemporary Music. *Musical Imaginations: Multidisciplinary Perspectives on Creativity, Performance and Perception,* D. Hargeaves, D. Miell, and R. Macdonald, Eds. Oxford University Press

Pressing, J. 1987 The Micro- and Macrostructural Design of Improvised Music. *Music Perception* 5(2): 133-172

Pearce, M. T., & G. A. Wiggins, 2001. Towards a Framework for the Evaluation of Machine Compositions. *Proc. of the AISB'01 Sym. On Artificial Intelligence and Creativity in the Arts and Sciences.* Brighton: 22-32

Stowell, D., A. Robertson, N. Bryan-Kinns, &M. D. Plumbley. 2009. Evaluation of Live Human Computer Music Making: Quantitative and Qualitative Approaches. *Int. J. of Human Computer Studies* 67, No. 11: 960-975.

Smith, H. &R. T. Dean. 1997. *Improvisation Hypermedia and the Arts since 1945.* Routledge.

Roger B. Dannenberg 1984. An On-Line Algorithm for Real-Time Accompaniment. International Computer Conference, 193-198

Roger B. Dannenberg, Thom, Belinda, & Watson, David 1997. A Machine Learning Approach to Musical Style Recognition. Computer Science Department. Paper 504 http://repository.cmu.edu/compsci/504

Bishop, C. M. 1995. Neural Networks for Pattern Recognition. Clarendon Press.

Rowe, R , 1993. Interactive Music Systems. MIT Press.

Rubine, D. 1991. The automatic recognition of gestures. Tech. rep., School of Computer Science, Carnegie Mellon University

Adam Linson, Chris Dobbyn, & Robin Laney, 2012. Critical Issues in Evaluating Freely Improvising Interactive Music Systems. International Conference on Computational Creativity.

Clarke, E. 2005. Creativity in Performance. Musicae Scientiae 9:1, 157-182

Clarke, E. 2005. Ways of Listening: An Ecological Approach to the Perception of Musical Meaning. Oxford University Press

Collin, N. 2008. The Analysis of Generative Music Programs. Organised Sound, No. 3: 237-248

Lehmann, A., R. Kopiez, 2010.The Difficulty of Discerning between Composed and Improvised Music. Musicae Scientiae 14: 113-129

MacDonald, R., G. Wilson, & D. Miell. 2011. Improvisation as a Creative Process within Contemporary Music. Musical Imaginations: Multidisciplinary Perspectives on Creativity, Performance and Perception, D. Hargeaves, D. Miell, and R. Macdonald, Eds. Oxford University Press

Pressing, J. 1987 The Micro- and Macrostructural Design of Improvised Music. Music Perception 5(2): 133-172

Pearce, M. T., & G. A. Wiggins, 2001. Towards a Framework for the Evaluation of Machine Compositions. Proc. of the AISB'01 Sym. On Artificial Intelligence and Creativity in the Arts and Sciences. Brighton: 22-32

Stowell, D., A. Robertson, N. Bryan-Kinns, &M. D. Plumbley. 2009. Evaluation of Live Human Computer Music Making: Quantitative and Qualitative Approaches. Int. J. of Human Computer Studies 67, No. 11: 960-975.

Smith, H. &R. T. Dean. 1997. Improvisation Hypermedia and the Arts since 1945. Routledge.