MU_PSYC: ALGORITHMIC MUSIC COMPOSITION WITH A MUSIC-PSYCHOLOGY ENRICHED GENETIC ALGORITHM

by

Brae Stoltz

B.Sc. (Honours) in Computer Science, University of Northern British Columbia, 2017

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER SCIENCE

UNIVERSITY OF NORTHERN BRITISH COLUMBIA

February 2020

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

Recent advancement of artificial intelligence (AI) techniques have impacted the field of algorithmic music composition, and that has been evidenced by live concert performances wherein the audience reportedly often could not tell whether music was composed by machine or by human. Among the various AI techniques, genetic algorithms dominate the field due to their suitability for both creativity and optimization.

Many attempts have been made to incorporate rules from traditional music theory to design and automate genetic algorithms. Another popular approach is to incorporate statistical or mathematical measures of fitness. However, these rules and measures are rarely tested for their validity.

This thesis is aimed at addressing the above limitation and hence paving the way to advance the field towards composing human-quality music. The basic idea is to look beyond this constrained set of traditional music rules and statistical/mathematical methods towards a more concrete foundation. We look to a field at the intersection of musicology and psychology, referred to as music-psychology.

To demonstrate our proposed approach, we implemented a genetic algorithm exclusively using rules found in music-psychology. An online survey was conducted testing the quality of our algorithm's output compositions. Moreover, algorithm performance was analyzed by experimental study. The initial results are encouraging and warrant further research. The societal implications of our work and other research in the field are also discussed.

Chapter 1

Introduction

Algorithmic Composition (AC) is a field focused on composing music using algorithms. This is a multidisciplinary effort with contributions from musicians, composers, computer scientists, mathematicians, and psychologists. The field has a rich history, dating back to at least 1757. A brief history is presented here to set the context for the work presented in this thesis.

1.1 History

The field of AC can be traced back to at least the 18th century, where games were devised to create music by an algorithmic process. Currently, state-of-the-art Artificial Intelligence (AI) algorithms are dominating the field, leading to more autonomous composition systems.

1.1.1 Early Days (1757-1990)

While virtually all contemporary research in this field is aided by a computer, history has shown that a computer is not a requisite to the definition of AC [20]. The act of composition by algorithm has been around for hundreds of years, with the earliest known example widely taken to be Kirnbergers's *Der allezeit fertige menuetten-und polonoisenkomponist*[1] in 1757 [54], a "musical dice game" wherein the player rolls a set of dice repeatedly in order to randomly select pre-existing bars of music. At this time musical games were popular, with many others releasing

[1]Mozart's musical dice game is often taken to be the first, however Kirnberger's was published nearly 40 years prior. This confusion is understandable given that Kirnberger was Mozart's publisher.

their own form of the musical dice game (see Hedges' work for a detailed account [39]), including Mozart in 1793 [3, 39]. In 1856, *The Quadrille Melodist* offered pianists a set of cards that supposedly allowed them to generate 428 million different quadrilles[2] [5, 26, 75]. The Quasi-algorithmic technique of breaking music into repeatable sections and using number ratios measured in the world or the Fibonacci sequence in determining the number of repeats dates back to the 13th and 14th century [26].

As the computer emerged, more sophisticated algorithms were explored; one of the earliest computers built, the Illinois Automatic Computer (ILLIAC), was used by Hiller and Isaacson in 1957 to generate music using mathematical techniques, compiling a set of generated musics into the *ILLIAC Suite* [44]. Before the suite was completed, Hiller conducted many experiments, one being a collaboration with many others, named *Push Button Bertha*, which aired on the television show *Adventure Tomorrow* [3].

In the period between 1955 and 1965 there were at least 20 independent projects in AC [2, 3]. These include Hiller and Baker's MUsic SImulator-Interpreter for COMpositional Procedures (MUSICOMP) [4, 41, 42], König's Project 1 [56] (and later Project 2 in 1970 [55]) and Xenakis' Stochastic Music Program (SMP) [90]. The next notable composition system was David Cope's Experiments in Musical Intelligence (EMI), first started in 1981[3], which evolved into a book in 1996 [17, 18].

The motivation behind Cope's EMI was later admitted to be "selfish" [19], as it was intended to overcome a writer's block. Similar to Cope, composer John Adams created his own software program to use as a tool for composing music, although he "certainly does not let the computer make decisions for [him]"[4]. For a comprehensive listing of AC systems between 1961 and 2007, see Ariza's algorithmic.net at `http://algorithmic.net/` [2].

[2]A quadrille is a type of dance popular in late 18th- and 19th-century Europe.

[3]This date is lifted from David Cope's web page on EMI: `http://artsites.ucsc.edu/faculty/cope/experiments.htm`.

[4]This quote is from an interview conducted by Robert Davidson, accessible courtesy of the Wayback Machine [84]: `https://web.archive.org/web/20070205051022/http://www.topologymusic.com/articles/adams.htm`.

1.1.2 Contemporary (2000 onward)

As processing speed increased, the door opened for more computationally complex algorithms to be used, particularly AI algorithms such as Deep Neural Networks (DNNs) and Evolutionary Algorithms (EAs). Considering the increasing complexity of implementing such algorithms, most of the research in the field is being done by computer scientists. This comes in stark contrast to the early days of AC where several of the high profile individuals involved, particularly König, Xenakis, and Cope, were composers rather than scientists. A modern example of these composers however is Arne Eigenfeldt who created and iterated on his music generation software, the Kinetic Engine (KE), for many years [28–31].

The involvement of composers in the process of creating AC systems has been called for [15, 81], however it is exceedingly rare. An excellent example of contemporary AC research collaboration is the work of Chun-yien Chang and Ying-ping Chen, a composer and computer scientist respectively, whom created a generative composition system [15]. Their research also includes a performance of an output composition by a world-renowned cellist.

Machine Learning (ML) has been massively successful in many areas and also in AC. There is no shortage of music generation systems that utilize ML [7, 14, 47, 50, 51, 59, 66, 67, 71, 76, 89, 93]. In particular, there has been a plethora of experiments and research done by Google, such as AI-Duet, a system that attempts to perform a duet as the user plays the piano in real-time [66]. Also popular is Google's Bach Harmonizer Doodle[5], which attempts to harmonize[6] a user-inputted melody line in the style of Bach.

1.2 Directions

Our research uses a Genetic Algorithm (GA) to compose music. GAs, which constitute a subfield of AI, are explored within the context of AC in chapter 3. The design of our GA is similar to many of those in the field, however the domain knowledge present in the system is unique.

[5]https://www.google.com/doodles/celebrating-johann-sebastian-bach

[6]What is meant by harmonize is that other parts are generated that are played concurrently with the original input melody.

Insights from various research in the field of music-psychology were adapted into rules which were placed throughout the components of our GA. A presentation of the various related research in music-psychology is given in chapter 2, while our music generation system, MU_PSYC, is in chapter 4. The implications of this thesis and other related work in this field are laid out in chapter 6. Conclusions and future directions are described in chapter 7.

Chapter 2

Music-Psychology

A relatively recent branch of psychology and musicology, referred to as music-psychology, aims to investigate music scientifically. Strides have been made in determining musical universals, providing scientific explanations to current musical theory, and discovering new rules or frameworks have been made. This chapter explores various empirical research in this field.

2.1 Musical Theory and Understanding

"Music Theory" is a ubiquitous term often used in passing and rarely defined. One of the most trusted sources on music [88], *The Oxford Companion To Music* [32], does define the term, although in three separate ways:

1. "The first is what is otherwise called "rudiments", currently taught as the elements of notation, of key signatures, of time signatures, of rhythmic notation, and so on ...",

2. "The second is the study of writings about music from ancient times onward ...",

3. "The third is an area of current musicological study that seeks to define processes and general principles in music—a sphere of research that can be distinguished from analysis in that it takes as its starting-point not the individual work or performance but the fundamental materials from which it is built."

For the purposes of our research, we focus on its meaning in the context of composition, which corresponds to this third definition.

The goal of music theory is then to identify patterns of music and provide us with basic rules of composition. However, composers often violate these rules to much praise, such as the popularity of John Cage's "4'33" where the performer is instructed to not play their instrument for the entire piece. Movements such as Schoenburg's school of atonal music also broke many of the established rules of harmony. Within AC, the famed Xenakis explained that a teacher, Arthur Honegger, scolded his music for using parallel 5ths and octaves, which he ignored [91]. Entire genres of music seem to consistently violate them as well, such as the pervasiveness of power chords[1] in Rock music which again goes against parallel fifth and octave rules in music theory.

In an early survey of AC, Papadopoulos and Wiggins state that "what is missing [from the field of AC] is a thorough-going account of musical cognition from the early stages of perception to the complex stage of creation, and composition" [73]. Such sentiments are rare in this field as most work does not investigate the validity of their music theoretic [22, 24, 60, 74, 83] or mathematical/statistical methods [33, 63, 68] rules.

Of course, science aims to move towards more concrete understanding in general, with the term *folk-psychology* often used to describe older, unscientific ways of understanding. Folk-psychology, also known as common-sense psychology, is defined by "Goldman as the naive understanding of mental state concepts" [37], and is often referred to as a 'primitive' understanding. A relatively recent branch of psychology, referred to as music-psychology, aims to further musical understanding by moving beyond its current folk-psychological distinction.

In his book, *Sweet Anticipation*, music-psychologist David Huron explains that "…it is inevitable that we must move beyond folk-psychology to psychology proper" [49]. More recently, Martin Rohrmeier echos Huron, again urging musical understanding away from folk-psychology and towards psychology, neuro-cognition, mathematics, and computation [77]. The folk-psychology Huron, Rohrmeier, and others are referring to is the current understanding of music, namely music theory.

[1]Power chords are triads with the 3rd taken out, often accompanied with an octave.

2.2 Literature

There have been various studies relating to and explaining various phenomena present in the human auditory system and contextualizing in relation to music. Moreover, conjectures from musicological study have been the subject of empirical study. In this section we present various research in relation to expectancy theory (section 2.2.1), Auditory Scene Analysis (section 2.2.2), voice-leading (section 2.2.3), musical universals (section 2.2.4), and empirical study pertaining to musicological conjectures (section 2.2.5).

2.2.1 Expectancy Theory

A number of music-psychologists have been involved in developing a theory that describes music as a way of taking advantage of "evolutionarily ancient physiological and cognitive mechanisms for detecting and responding to unexpected events" [10]. Music thus repeatedly meets or violates a listener's expectations in ways that brings pleasure. Leonard B. Meyer was probably the first to introduce expectation as a central element in the understanding and emotional response to music [69]. Daniel Levitin and David Huron further contributed to this theory with their respective books titled *This is Your Brain of Music* [58] and *Sweet Anticipation* [49]. In *Sweet Anticipation*, David Huron proposed one of the most concrete theories of expectancy and music, the Imagination-Tension-Prediction-Reaction-Appraisal (ITPRA) theory.

Huron's work lays out four different types of expectation corresponding to different types of auditory memory: *Veridical*, *Schematic*, *Dynamic*, and *Conscious* [49]. Derived from episodic memory, *Veridical* expectation pertains to the knowledge of how a piece progresses in time. Being exposed to different musical styles forms ones *Schematic* expectations, which are related to semantic memory. *Dynamic* expectations are those of short-term memory, containing information about a piece in real-time. Finally, *conscious* expectations are those relating to a listener's conscious thoughts about how a piece of music will sound.

Egermann et al. provide evidence for this theory by way of studying listener's emotions in a live concert setting [27]. In the same year, Rohrmeier attempted to bridge theories of expectancy with musical theory, providing many concrete examples [77]. Despite its popularity, expectancy theory is meant to explain only a fraction of emotional responses to music [27, 53].

2.2.2 Auditory Scene Analysis

Another evolved aspect of the auditory system is Auditory Scene Analysis (ASA), which is the ability to locate and identify sources of sound. In exploring the origins of music, Trainor finds that musical pitch and time developed according to auditory principles evolved for ASA [85]. ASA is subject to empirical study by Bonin et al., whom experiment with pitch, timbre and spatial ways to change the uncertainty in the number, identity or location of sounding objects [10]. They found that auditory cues that promote a single interpretation of a melody made unpleasant melodies more pleasant. The inverse was also found to be true; conflicting auditory cues made a pleasant melody sound less pleasant.

2.2.3 Voice-leading

In 2001, David Huron derived the rules of voice-leading from well known auditory principles established with cross-cultural samples [48]. Expectancy theory and ASA were leveraged to provide empirical evidence for these principles. Huron categorizes these principles as either core—principles that apply to all music—or auxiliary (genre specific), then derives rules using them. The goal of these rules is to optimize voice-leading, otherwise referred to as auditory streaming in psychology.

Notably, Huron's derivation found not only existing voice-leading rules but also those which do not appear in music theory teachings [48]. Below is a list of rules Huron derives that are familiar in that they exist in music theoretic teachings [48].

1. Registral Compass Rule; *Voice-leading is best practiced in the region between F2 and G5, roughly centered near D4.*

2. Chord Spacing Rule; *In general, chordal tones should be spaced with wider intervals between the lower voices.*

3. Avoid Unisons Rule; *Avoid shared pitches between voices.*

4. Common Tone Rule; *Pitch-classes common to successive sonorities are best retained as a single pitch that remains in the same voice.*

5. Conjunct Movement Rule; *If a voice cannot retain the same pitch, it should preferably move by step.*

6. Nearest Chordal Tone Rule; *Parts should connect to the nearest chordal tone in the next sonority.*

7. Part-Crossing Rule; *Avoid the crossing of parts with respect to pitch.*

8. Pitch Overlapping Rule; *Avoid "overlapped" parts in which a pitch in an ostensibly lower voice is higher than the subsequent pitch in an ostensibly higher voice.*

9. Parallel Unisons, Octaves, and Fifths Rule; *Avoid parallel unisons, octaves, or fifths.*

10. Exposed Intervals Rule; *When approaching unisons, octaves, or fifths, by similar motion, at least one of the voices should move by step.*

These range from simply avoiding unisons to more complex rules such as the exposed intervals rule in Palestrina-style counterpoint. Below is a list of unfamiliar rules that Huron derives [48].

1. Toneness Rule; *Voice-leading should employ tones that evoke strong, unique pitch sensations. This is best achieved using harmonic complex tones.*

2. Sustained Tones Rule; *In general, effective voice-leading is best assured by employing sustained tones in close succession, with few silent gaps or interruptions.*

3. Tessitura-Sensitive Spacing Rule; *It is more important to have large intervals separating the lower voices in the case of sonorities that are lower in overall pitch.*

4. Avoid Octaves Rule; *Avoid the interval of an octave between two concurrent voices.*

5. Avoid Perfect Fifths Rule; *Avoid the interval of a perfect fifth between two concurrent voices.*

6. Avoid Tonal Fusion Rule; *Avoid unisons more than octaves, and octaves more than perfect fifths, and perfect fifths more than other intervals.*

7. Leap-Lengthening Rule; *Where wide leaps are unavoidable, use long durations for either one or both of the tones forming the leap.*

8. Semblant Motion Rule; *Avoid similar or parallel pitch motion between concurrent voices.*

9. Parallel Motion Rule; *Avoid parallel motion more than similar motion.*

10. Oblique Approach to Fused Intervals Rule; *When approaching unisons, octaves, or fifths, it is best to retain the same pitch in one of the voices (i.e., approach by oblique motion).*

11. Avoid Disjunct Approach to Fused Intervals Rule; *If it is not possible to approach unisons, octaves and fifths by retaining the same pitch (oblique motion), step motion should be used.*

12. Avoid Semblant Approach Between Fused Intervals Rule; *Avoid similar pitch motion in which the voices employ unisons, octaves, or perfect fifths. (For example, when both parts ascend beginning an octave apart, and end a fifth apart.)*

The auxiliary principles Huron [48] describes are listed below.

1. Onset Synchrony – sounds that are coordinated in time are likely to fuse into a single sound.

2. Limited Density – humans can only keep track of a few number of distinct auditory streams.

3. Timbral Differentiation – sounds with similar timbres are more likely to be perceived as a single sound source.

4. Source Location – differentiation of sound sources is more difficult when sounds occur close to one another.

Closely related to onset synchrony is offset synchrony, where streams are fused even further when their cessation is coordinated [11, 25]. Duane offers an empirical study of onset synchrony, pitch co-modulation, and spectral overlap in 18th and 19th century string quartets [25]. Pitch co-modulation is similar to Huron's parallel motion rule [48], where sounds are tonally fused when moving in parallel. Spectral overlap also relates to a rule from Huron, the avoid tonal fusion rule, where sounds are tonally fused when separated by a perfect interval.

2.2.4 Musical Universals

The study of music universals[2] uncovers features of music that exist across many cultures. There have been many universals introduced both in psychology and music theory, which are summarized well by Brown and Jordania, whom compiled a list of over 60 [12]. However, only until recently has the field received empirical study through Savage et al.'s statistical analysis of 304 music recordings spanning nine geographic regions [78]. Their work found only 21 of the proposed 32 universals to be statistically significant, which we list here:

1. 2- or 3- beat subdivisions
2. Non-equidistant scales
3. Less than seven scale degrees
4. Chest voice
5. Discrete pitches
6. Motivic patterns
7. Descending/arched contours
8. Word use
9. Small intervals
10. Isochronous beat
11. 2-beat subdivisions
12. Short phrases
13. Instrument use
14. Male performers

[2]Music universals may be a misleading term; they are, in essence, generalizations [78].

15. Metrical hierarchy
16. Group performance
17. Voice use
18. Few durational values
19. Sex segregation
20. Phrase repetition
21. Percussion use

An interesting finding in Savage et al.'s research was the preponderance of descending/arched contours in the world's musics [78]. Other work on this specific universal has shown it to also exist in bird song [82][79], with Tierney et al. providing biological explanations for it and for the ubiquity of small intervals [82]. Only a handful of these empirically established music universals relate to composition, at least given the limited representations of music in AC.

2.2.5 Supporting Musicological Conjectures

Broze and Huron test the musicological conjecture that music which is higher in pitch is played faster [13]. Both compositional and performance practice were found to fit this trend. Interestingly, performers playing speeds were highly correlated with their tessitura[3]. Ornamentation was also found to occur most often in the uppermost voice. Another study relating to the perception of higher pitches is done by Trainor et al., whom explain the established high voice superiority effect in polyphonic music [86]. This is the human tendency to focus on the voice with the highest pitch in music.

Ammirante and Russo explore a phenomenon where skips[4] are frequently small, which they call the *low-skip bias* [1]. They find evidence that (i) skips occur more frequently in instrumental music rather than vocal, (ii) fewer skips are preferred over higher skips, and (iii) the more vocal

[3]Tessitura refers to a specific range of pitches.
[4]A skip, also known as a leap, is a large jump in pitch from one note to another.

music that a composer is involved in correlates with less skips. From this they conclude that composers often construct instrumental melodies with a vocal template.

Chapter 3

Music Composition with Genetic Algorithms

Alongside Machine Learning (ML) algorithms, Genetic Algorithms (GAs) are very popular within the field of Algorithmic Composition (AC). This chapter provides a background on what a GA is and how it finds optimized solutions within massive search spaces in section 3.1. Its suitability for music generation and various related literature is reviewed and discussed in section 3.2.

3.1 Background

Goldberg and Holland define GAs as "probabilistic search procedures designed to work on large spaces involving states that can be represented by strings" [36]. The main components of a GA are the genome representation which defines the search space, the fitness function which provides a means to judge an individual's quality, and the genetic operators. Selection, mutation, and crossover are the standard operators, with mutation and crossover providing a means for changing individuals and selection being the strategy of choosing which individuals survive to the next generation.

Each generation is a successive application of the fitness function, selection, crossover, and then mutation. The result from this repeated application is a constant, gradual increase in quality of the population. Often the purpose of running a GA is to find a single optimal solution which corresponds to the best individual in the final population. A basic GA is shown as algorithm 1 below.

Algorithm 1: Basic Genetic Algorithm

```
Output: Most Fit Individual Found
Populations: $P_{cur}, P_{new}$;

initialize $P_{cur}$; //random population
for some number of generations do
    calculate fitness for each individual in $P_{cur}$;
    select and move most fit individuals from $P_{cur}$ to $P_{new}$;
    randomly apply crossover to create children and add them to $P_{new}$;
    randomly copy and apply mutation to create new individuals and add them to $P_{new}$;
    $P_{cur} = P_{new}$;
end
return Most fit individual in $P_{cur}$;
```

3.2 Genetic Algorithms For Music Composition

Given the generic nature of GAs, they can be applied to a multitude of problems. Horner was one of the earliest researchers to hypothesize that GAs may lend themselves to generating music [46]. Early in the field, Hiller expressed that "computer-assisted composition is difficult to define, difficult to limit, and difficult to systematize" [43], thus we will focus more on the presentation of various research rather than a comprehensive taxonomy. This section is divided into the main components of a GA, presenting the approaches of various researchers.

3.2.1 Representation of Compositions

Western staff notation is used to visualize music in nearly all AC systems. Even when generating non-Western music, it is visualized with staff notation [94]. Outputs are auralized (aurally visualized) most often with the Musical Instrument Digital Interface (MIDI), a standard for playing, editing, and recording digital music. Because of the pervasiveness of staff notation and MIDI, they inspire representations of music in AC systems. Particularly, many systems adopt a MIDI-esque representation for its low spatial complexity; a note's pitch, for example, is represented in one byte rather than capturing pitch class, octave, or accidental as in staff notation.

Overly simplified representations of music in music generation systems is a widely accepted problem in AC [22, 40, 62]. For instance, single-part melody generation is common not only in the early works of Biles [7–9], but also in more recent research [21, 83]. Only a few attempts have been made to encompass non-fundamental musical features such as dynamics, key modulation, time modulation, accentuation, etc.

In GAs specifically, music is represented as an individual in the population. Most often, entire compositions are coded as individuals and thus the output of the GA is simply the most fit individual at the end of the run [7–9, 24, 60, 83, 94]. Other work, however, encodes individuals as phrases which are combined to form a full composition [30]. In other words, the population, or a proper subset of it, is combined to create the output composition.

3.2.2 Fitness Functions

At least within AC, fitness evaluation is widely considered the most difficult to design and the most crucial component of a GA [9, 21, 62, 65, 70]. Two main types of fitness functions emerge from the literature; (1) human-assisted; and (2) autonomous. Autonomous fitness functions are further split into two types, mentioned by de Freitas and Guimarães [21]; (2i) those that measure the individual's similarity to a target music; and (2ii) those that measure how well the individual follows compositional rules. Type (2i) fitness functions are typically hybrid systems in that the GA is usually combined with a NN, Markov Chains, some other Machine Learning (ML) algorithm, or statistical methods. However, ML algorithms are more often used by themselves in a generative fashion.

3.2.2.1 Human-assisted Fitness Functions

Biles' early work on his AC system, *GenJam*, used a human-based fitness function (type 1), wherein musical excerpts where judged (assigned a fitness value) by the user [8]. This approach is used in many Interactive Genetic Algorithms (IGAs), where the goal of the algorithm is to interact with the user. However, work by Gong et al. proposes that human evaluation is not needed in an IGA, and that it leads to the *human fatigue bottleneck* [38]. This bottleneck is also found in the work of Biles [8], and Jordanous whom describes states that "any reliance on human intervention introduces a fitness bottleneck" [52], and acknowledged by Matić [68] and Gong et al. [38].

3.2.2.2 Autonomous Fitness Functions

The compositional rules used in type (2ii) fitness functions are most often taken from traditional music theory practices [94], particularly Western polyphonic music [22, 24, 60], and even personal experience [62]. The rules of the fitness function proposed by Donnelly and Sheppard closely resemble traditional music theory teachings, with an emphasis placed on counterpoint for later work [24]. Liu et al.'s research is quite similar, although they impose weights on rules [60]. More recent work by de Vega focuses on the problem of 4-part harmonization, a problem often given to music theory students [22]. Although the rules are not explicitly stated in his work, there are impressive results.

Matić's work shows type (2i) fitness functions in perhaps the simplest form; he uses arithmetic means and variances to determine the similarity to an input individual [68]. Similarly, Fox and Crawford created a hybrid system, *MAGE*, which used Markov chains as their fitness function [33]. Later, Ting et al. used Matić's metrics [68] for their fitness function, however they also included a rule-based evaluation akin to type (2ii) fitness functions [83]. Their work is then both determining an individual's similarity to input music (type 2i) and measuring how well it conforms to music theoretic rules (type 2ii).

After Biles' initial work on *GenJam* [8], he, along with Anderson and Loggi, experimented with the possibility of using a Neural Network (NN) as a fitness function [7]. Their work produced a negative result, stating that "the clear and unsurprising conclusion from this study is that humans listen to music in complex and subtle ways that are not captured well by simple statistical models." [7]. A couple of years later Burton and Vladimirova published a similar system with more positive result, however their system only generated rhythms [14].

Over 15 years later, more complicated and computationally complex statistical models, most often different types of NNs, are able to be used. In Manaris et al.'s GA, they employ a prototypical Artificial Neural Network (ANN) for a fitness function [65]. Mitrano et al. proposed an IGA based on Biles' *GenJam* [8], but used a Recurrent Neural Network (RNN) to judge fitness [70]. However, NNs are more commonly designed for music generation not fitness evaluation, apparent by Mitrano et al.'s adaptation of a generative RNN from Google [70].

3.2.2.3 Fitness-less Genetic Algorithms

Later work on *GenJam* involved changing the system to be autonomous, which Biles achieved by eliminating the need for a fitness function altogether [9]. He describes fitness itself as a bottleneck, as it is difficult to implement and is inherently subjective[1]. Eigenfeldt followed similarly, by modifying his longstanding music generation system, Kinetic Engine (KE) [28–31], to eliminate the fitness function [30]. To accomplish this, the mutation and crossover operators are more intricately designed in order to avoid generating low quality individuals. A middle ground exists in the work of de Freitas and Guimarães, whom implement "guided" genetic operators to avoid generating "absurd" solutions, making fitness-free evolution feasible, but also include a minimal fitness function where individuals are only judged on their conformity to the C major scale [21].

Biles' [9] and de Freitas and Guimarães' [21] work are both limited to generating single part melodies, whereas Eigenfeldt's system produced multi-part works, one of which was performed in concert [30]. However, his system is closed source and designed to be extremely personal [30]. Despite the ability to avoid Biles' described *fitness bottleneck* and the success of Eigenfeldt's KE [30], fitness-less GAs have not seen widespread adoption in AC.

3.2.3 Genetic Operators

We present various research on the three standard genetic operators: selection (section 4.3.3.1), mutation (section 4.3.3.3), and crossover (section 4.3.3.2). Other genetic operators that have been devised by AC researchers are grouped together with mutation, as often they are more complex mutations (they create a new individual from another).

[1]Biles' *fitness bottleneck* [9] is not to be confused with Jordanous' [52]. The former states that fitness itself is the bottleneck, while the latter alludes to the *human fatigue bottleneck* stated by Gong et al. [38].

3.2.3.1 Selection

The simplest form of selection is elitist selection, where the most fit individuals move on to the next generation. Donnelly and Sheppard's work uses this type of selection, where the top 10% individuals continue onto the next generation unchanged [24]. Similarly, Matić removes the lowest fit individuals in his work, however potential duplicates are removed beforehand [68]. [94][33]

These simplistic selection schemes result in *premature convergence*, essentially exploring only a subset in the solution space. The algorithm then attempts to find the most fit individual within this subset, when there is likely many other more fit solutions within the entire solution space. To combat this, many other selection schemes have been proposed that increase genetic diversity (increase the area explored), with the most popular being subject to comparison in Goldberg's early research [35]. These selection schemes are: (i) fitness-proportionate (roulette wheel), (ii) rank, (iii) tournament, and (iv) Genitor.

Most research within the field of AC does not provide an explanation of their choice of a selection scheme, most likely due to the fact that the fitness function is the main focus. Tournament selection is often chosen here, perhaps due to its similarity in performance to the other selection schemes and its better time complexity ($O(n)$ where n is the number of selected individuals). The development of new selection schemes has occurred in more recent research, such as the blending of fitness-proportionate and rank selection [57], however these have not been used in AC.

Most notable work in AC uses tournament selection, specifically binary tournament selection[2] [60, 61, 63, 83]. Roulette wheel selection has seldom been used, featuring in the work of de Vega [22] and Bernardes et al. [6]. Eigenfeldt is one of the only researchers to use rank selection, presented in his fitness-less version of his Kinetic Engine [30]. However, most research tends to not list their selection method [64, 65, 70, 80].

[2]Binary tournament selection is also known as 2-tournament selection, or tournament selection with a tournament size of 2.

3.2.3.2 Mutation

The purpose of mutation is to avoid stagnation, as "once combinations of variants in the current population have been explored [(crossover)], no new variations are possible" [45]. Mutation is often split into many sub-operators, each of which randomly alter different musical element(s) [7–9, 22, 24, 33, 60, 68, 70, 94]. These include randomly altering pitches and changing durations of notes primarily, but some research includes multi-note mutations inspired by century-old compositional techniques such as inversions and retrogrades[3] [21, 24, 33].

Once an individual is selected for mutation, each of the sub-mutations is assigned a probability of being selected which is arbitrary, experimentally set, or dynamic. Dynamic mutation and crossover probabilities have been introduced early in the study of GAs [87], although AC research tends to employ arbitrary or experimentally set probabilities. In their GA for music composition, Donnelly and Sheppard mutate the selection probabilities of mutation and crossover themselves [24]. The best individuals will approximately have the best set of mutation and crossover probabilities. An odd finding of their work was that the best individuals had used crossover approximately 99% of the time. This indicates their mutations often did not result in increases in fitness bigger than those offered by crossover. These findings may generalize to other GA based approaches in AC, as their mutations are common within the field.

Another approach to improving mutation is to control which individuals it is applied to, and how many times it is applied within one generation. For example, Matić devised a mutation operator that was applied more often to more fit individuals in later generations [68]. The result was that later generations stagnate less.

The creation of a more informed mutation operator, one with better knowledge of where in the individual to mutate, is mainly the goal of fitness-less GAs. However, de Vega wishes to employ it in a more standard GA (one with a rule-based fitness function) [22]. As Biles describes, this effectively decentralizes domain knowledge, distributing it elsewhere in the GA [9]. In the case of fitness-less GAs, domain knowledge is also distributed in the crossover and population initialization components of the algorithm.

[3]The technique of inversion is not to be confused with an inversion operator; research involving these techniques [24, 33] often perform retrogradation in their inversion operator.

3.2.3.3 Crossover

In many musical GAs [7, 8, 21, 24, 33, 60, 63, 65, 83, 94], crossover is implemented by splicing together different sections of music from two parents to form a child. These sections are often random windows of notes (or time) in the music [7–9, 24, 65, 83], although more current work often limits these sections to bars of music [21, 33, 60, 63, 94]. Eigenfeldt proposed single parent crossover in his early versions of the Kinetic Engine (KE), using a first-order Markov chain to generate the child [28, 31].

In a fitness-less version of Eigenfeldt's KE, crossover was omitted as it "was felt to produce unmusical results" [30]. Crossover is also omitted from Matić's work using a standard fitness function [68]. In Biles' fitness-less version of GenJam, specific points of crossover are selected over others such that the resulting child closely preserves horizontal intervals in the parent [9]. This essentially includes domain knowledge in the crossover operator.

3.2.4 Evaluation

Researchers often present their AC systems with little to no evaluation of its efficacy. Moreover, no attempt has been made to empirically compare the efficacy of separate AC systems, as it is not possible without surveying. With this, the field is little more than scattered attempts, however some work has taken the initiative to effectively study their own systems via survey. The largest being Scirea et al.'s work, where data collected from 298 participants was gathered and studied [80]. A non-scientific approach is to expose the algorithm's music in a concert setting, as done with the ILLIAC Suite [44] and *One, Previously* by Eigenfeldt's Kinetic Engine [30], or through a popular platforms such as with Google's many musical experiments [66, 71]. This simply allows for a conversation to be started and subjective judgements to be made, which is more effective than presenting notated snippets of generated music within a publication.

Chapter 4

A Psychology Enriched Genetic Algorithm for Music Composition

Given the success of GAs in AC, the criticisms of algorithms using traditional music-theoretic rules, and the many music-psychological rules which have gone unused in AC, a new GA is proposed here. This GA directly utilizes insights gained from music-psychology by incorporating found patterns or rules into the genome representation and the fitness function. Our proposed GA can be executed in parallel to reduce run-time. An online survey is conducted to measure the effectiveness of non-traditional rules.

The rest of the chapter is organized as follows. Motivations for our work are laid out in section 4.1. In section 4.2 we list the various compositional rules which our algorithm aims to satisfy. The details of genome representation are shown in section 4.3.1, followed by the population initialization in section 4.3.2. Section 4.3.3 contains the selection, mutation, and crossover operators – sections 4.3.3.1, 4.3.3.2, and 4.3.3.3 respectively. The fitness function is described in section 4.3.4. Section 4.3.5 discusses the algorithm from a high level, followed by section 4.3.6 which describes the implementation of the GA.

4.1 Motivations

Despite Google's massive infrastructure, ML methods still struggle to generate music well, as does most of the field. Moreover, ML algorithms are a black box in that it is unknown what exactly is happening inside them, making any insightful analysis difficult. Data sets are often purely of classical music, usually composed hundreds of years ago since those containing newer music are rare given that they are not often accompanied by notation that can be easily input. Furthermore, work by Sturm et al. explores the idea that data sets which include copyright protected work may cause a ML algorithm's output to be in violation of copyright, although few countries have created laws related to music generation systems [81]. These issues will likely impede the application of ML algorithms to newer genres.

For many musicians and composers, the definitive understanding of music is taken to be music theory. Similarly, AC research often looks towards music theoretic rules and guidelines when crafting a generative system. However, a different field of study rooted in musicology and psychology, referred to as music-psychology, aims to move beyond current music theoretical understanding to provide scientific explanations of music. Papadopoulos and Wiggins were among the first with the aim of using this field's domain knowledge, stating in their 1999 survey of AC that "what is missing [from this field] is a thorough-going account of musical cognition from the early stages of perception to the complex stage of creation and composition" [73].

Despite this sensible direction, creating a system that incorporates music-psychological rules is relatively unexplored territory. As knowledge would need to be directly built into the system, a ML algorithm is not suitable. An EA, particularly a GA, is chosen for this work, given its success in the field.

4.2 Musical Rules

We use select universals from Savage et al.'s work on musical universals [78] and formulate them into rules. More rules are implemented based on Huron's work in deriving voice-leading rules [48], which are discussed in detail in chapter 2. These are split into three main types: Savage et al.'s musical universals [78] (type **R1**), Huron's traditional rules [48] (type **R2**), and Huron's non-traditional rules (type **R3**).

The following are rules of type **R1**, which are musical universals restated as rules.

1. Discrete Pitches Rule – *Pitches should be organized discretely.*

2. Beat Subdivisions Rule – *Rhythmic beats should be able to be subdivided by 2.*

3. Scale Length Rule – *Scales should be less than or equal to 7 scale degrees.*

4. Non-equidistant Scales Rule – *The differences between adjacent scale degrees should vary (no whole-tone nor chromatic scale).*

5. Descending/Arched Contour Rule – *Penalizes melodic contours that are not arched or purely descending.*

6. Few Durational Values Rule – *Penalizes compositions that use a wide variety of duration values. We arbitrarily place the limit as five, since the exact limit is not known.*

Next are the traditional voice-leading rules (type **R2**).

7. Registral Compass Rule – *Avoid notes that lay outside of the register bounds (*`F2` *to* `G5`*).*

8. Part Crossing Rule – *Penalizes individuals if, at any given point, parts cross with respect to pitch.*

9. Avoid Unisons Rule – *Avoid unisons between the voices.*

10. Exposed Intervals Rule – *When approaching unisons, octaves, or fifths by similar motion, penalizes the situation where neither voice moves by step.*

11. Chord Spacing Rule – *Chordal tones should have more separation in the lower parts. Specifically, the soprano and alto voices should be separated by no more than an octave as should be the alto and tenor voices. The bass voice and tenor voice have no such restriction.*

Lastly, rules of type **R3** (the non-traditional voice-leading rules).

12. Leap Lengthening Rule – *Leaps (movement by more than 7 semitones) should be accompanied by longer durations, and stepwise motion with shorter durations. Penalizes outliers of this trend.*

13. Semblant Motion Rule – *Avoid similar and parallel motion between concurrent voices.*

14. Parallel Motion Rule – *Avoid parallel motion more than similar motion.*

15. Avoid Semblant Approach To Fused Intervals Rule – *Avoid similar and parallel motion when moving to and from unisons, octaves, and fifths.*

16. Parallel Fused Intervals Rule - *Avoid parallel unisons, octaves, and fifths.*

17. Avoid Tonal Fusion Rule – *Avoid unisons more than octaves, octaves more than fifths, and fifths more than other intervals.*

18. Oblique Approach To Fused Intervals Rule – *When approaching unisons, octaves, or fifths, rewards oblique motion more than stepwise motion and penalizes other types of motion.*

19. Toneness Rule – *Use only harmonic complex tones.*

20. Sustained Tones Rule – *Minimize silent gaps or interruptions.*

4.3 Genetic Algorithm

Our GA has four main components, namely genome representation, population initialization, genetic operators, and fitness function. We describe them next before presenting our algorithm.

4.3.1 Genome Representation

Similar to the work of Zheng et al. [94], we represent genomes on the level of musical components rather than a character or binary level encoding. Genomes are stored in a custom data structure resembling Western Staff Notation and then converted to MIDI upon output. This helps maintain the boundary between different musical elements. Furthermore, rules 1-4, 19, and 20 in section 4.2 are followed in this representation.

The population PP is a list of n chromosomes:

$$\mathsf{PP} = \langle C_1, C_2, \ldots, C_i, \ldots, C_n \rangle.$$

A chromosome or 'composition' is a list of p parts:

$$C_i = \langle P^i_1, P^i_2, \dots, P^i_j, \dots, P^i_p \rangle.$$

P^i_j represents the j-th part in the i-th composition.

Next are the parts, which are lists of measures each of size m:

$$P^i_j = \langle M^{i,j}_1, M^{i,j}_2, \dots, M^{i,j}_k, \dots, M^{i,j}_m \rangle,$$

where $M^{i,j}_k$ is the k-th measure of the j-th part in the i-th composition. Each measure consists of a variable number of notes:

$$M^{i,j}_k = \langle N^{i,j,k}_1, N^{i,j,k}_2, \dots, N^{i,j,k}_l \rangle.$$

Notes consist of a pitch Pit, which refers to how high or low the note sounds[1], and a Duration Dur, meaning how long the note is played:

$$N^{i,j,k}_l = \langle Pit^{i,j,k}_l, Dur^{i,j,k}_l \rangle.$$

Rules 1 and 19 are satisfied by the use of discrete pitches, which symbolize harmonic complex tones. Rules 2 and 20 are satisfied since duration values, listed in table 4.1, are subdivided by 2, and there are silent gaps given that there are no rest notes.

Instead of representing a pitch as a single character as in most research, we define it as having a pitch class PC and an octave O:

$$Pit^{i,j,k}_l = \langle PC^{i,j,k}_l, O^{i,j,k}_l \rangle.$$

There are twelve pitch classes with the familiar letters A, B, C, D, E, F, G and their flats/sharps. A mapping of numbers to pitch classes which we use in our system is shown in table 4.2. Octaves correspond to specific pitch ranges which, when combined with a pitch class, specify a pitch.

The upper bound of the amount of notes allowed in a measure, l, is determined by the global

[1]This corresponds to the specific frequency of the fundamental tone.

Duration Value	1	2	4	8	16	32	64
Duration	Whole	Half	Quarter	Eighth	Sixteenth	32nd	64th

Table 4.1: Duration Values

Pitch class ID	0	1	2	3	4	5	6	7	8	9	10	11
Pitch class	C	C♯/D♭	D	D♯/E♭	E	F	F♯/G♭	G	G♯/A♭	A	A♯/B♭	B

Table 4.2: Pitch class IDs

time signature T and the shortest allowed note duration. The time signature is defined as

$$T = \langle T_n, T_d \rangle,$$

where T_d is the *delineation* of the note, which refers to a note duration (4 for quarter note, 8 for eighth note, etc.), and T_n is the number of those delineations allowed in a measure. The shortest allowed note duration, U, provides a bound on how many notes can be in a measure:

$$1 \leqslant l \leqslant T_n \times \frac{U}{T_d}.$$

The scale or key K defines which notes are allowed to be played. Similar to the time signature, the key cannot change over time as it is global for a specific population. A list of pitch classes is used to define the key. The default key, C Major, is thus

$$K = \{0, 2, 4, 5, 7, 9, 11\}.$$

To satisfy rules 3 and 4, the members of K cannot be all equally spaced, and $|K| \leqslant 7$.

4.3.2 Population Initialization

Each individual is initialized according to the parameters, listed in section 4.3.6.2. Measures are populated with T_n notes of random pitch and durations all equal to T_d. For instance, if the time signature is $T = \langle 3, 4 \rangle$, then each measure will have 3 quarter notes. Parts have pitch bounds,

$$B = \{\{lb_0, ub_0\}, \{lb_1, ub_1\}, \ldots, \{lb_p, ub_p\}\},$$

where p is the number of parts, lb is a lower bound, and ub is an upper bound. These bounds prevent notes from occurring outside of an instrument's range and massive part crossing.

4.3.3 Genetic Operators

Selection, crossover, and mutation operate on the composition space Φ:

- Mutation $\Pi : \Phi \to \Phi$.
- Crossover : $\Phi \times \Phi \to \Phi$.
- Selection $\Omega : \Psi \to \Phi$, where $\Psi = \mathcal{P}(\Phi) - \emptyset$, the set of all non-empty subsets.

4.3.3.1 Selection

The selection operator selects and removes a chromosome from the current generation and adds it to a new population. Two different selection operators are implemented, tournament selection and roulette wheel selection, however deterministic tournament selection is used in our experiments.

Tournament Selection

Tournament selection randomly chooses k individuals to be apart of a tournament τ and selects a winner of the tournament by probabilistic selection. Selecting the tournament is implemented in our algorithm by shuffling the population, choosing a random position h, and then taking the consecutive elements starting from that position:

$$\tau = \langle C_{1+h}, C_{2+h}, \ldots, C_{|\tau|+h} \rangle, \text{ where } (1 \leqslant h \leqslant n - |\tau|).$$

The tournament is then sorted by fitness in a descending fashion.

The probability of an individual being selected in this tournament is determined by the equation

$$p_i = \rho(1-\rho)^i, (0 \leqslant \rho \leqslant 1),$$

where i is the position of an individual in the sorted tournament, p_i is the probability of selection of that individual, and ρ is a constant. When $\rho = 1$, the best individual is always selected, which is known as *Deterministic Tournament Selection*.

Roulette Wheel

Roulette wheel selection, also known as fitness proportionate selection, selects individuals based on a probability that is based on their fitness compared to the rest of the population. The probability of a composition, C_i being selected is

$$p_i = \frac{F(C_i)}{\sum_{j=1}^{|PP|} F(C_j)}.$$

4.3.3.2 Crossover

Crossover is implemented on the measure level by randomly selecting two chromosomes (C_u and C_v) from the new population and creating a child C from them. The child is identical to one of the parents, except for one measure which is swapped out for the other parent's measure that is in the same position:

$$C_c = \otimes(C_u, C_v) = \langle P_1^u, P_2^u, \ldots, \langle M_1^{u,j}, M_2^{u,j}, \ldots, M_k^{v,j}, \ldots, M_m^{u,j} \rangle, \ldots, P_P^u \rangle, \text{ where}$$

$$(1 \leqslant j \leqslant p), \text{ and } (1 \leqslant k \leqslant m).$$

4.3.3.3 Mutation

When a chromosome is selected for mutation, one of four sub-operators are applied to create a new, mutated chromosome. These are

1. Random transposition (Π_{rt}),

2. Splitting (Π_s),

3. Merging (Π_m), and

4. Repetition (Π_r).

A probabilistic selection chooses which sub-operator will be used, with probabilities p_{rt}, p_s, p_m, and p_r respectively.

Mutation is defined as

$$\Pi(C_u) = C_c,$$

where C_u is the original composition and C_c is the mutation of it. Initially, $C_c = C_u$ except for one mutation, performed by one of the sub-operators described below.

Random Transposition Operator

A note is randomly chosen by indexes i, j, and k, and then shifted by a random scale degree, δ:

$$N_l^{c,j,k} = \langle Pit_l^{u,j,k} + \delta, Dur_l^{u,j,k} \rangle, (-6 \leqslant \delta \leqslant 6).$$

This is the only mutation operator that changes the pitch of notes.

Splitting

A note is selected and split into two notes by halving their duration:

$$M_k^{c,j} = \langle N_1^{u,j,k}, N_2^{u,j,k}, \dots, \langle Pit_l^{u,j,k}, \frac{Dur_l^{u,j,k}}{2} \rangle, \langle Pit_l^{u,j,k}, \frac{Dur_l^{u,j,k}}{2} \rangle, \dots \rangle.$$

Splitting allows the number of notes to increase during generation.

Merge

The merge operator merges two adjacent notes of the same duration:

$$M_k^{c,j} = \langle N_1^{u,j,k}, N_2^{u,j,k}, \dots, N_{l-1}^{u,j,k}, \langle \frac{Pit_l^{u,j,k} + Pit_{l+1}^{u,j,k}}{2}, Dur_l^{u,j,k} \times 2 \rangle, N_{l+2}^{u,j,k}, \dots \rangle, \text{ such that}$$

$$\mathrm{Dur}_{l}^{u,j,k} = \mathrm{Dur}_{l+1}^{u,j,k}.$$

Merging counteracts splitting by allowing the number of notes in a composition to shrink.

Repetition

The chromosome is duplicated with no mutation applied ($C_c = C_u$).

4.3.4 Fitness Function

The fitness function and the rule satisfaction functions output a real number in the interval [0,1]:

$$F : \Phi \to [0,1], \text{ and } R : \Phi \to [0,1].$$

Fitness is defined as the average of all the rules involved:

$$F(C) = \frac{1}{u} \sum_{R_i \in S_R} R_i(C),$$

where R_i is the rule satisfaction function for rule i, S_R is the set of rule functions used and $u = |S_R|$. Rule satisfaction functions take a chromosome as input and output how well it conforms to a specific rule:

$$R_i(C) = \frac{NIF_i(C)}{NI_i(C)},$$

where $NI_i(C)$ is the number of instances where the i-th rule *can* be followed, $NIF_i(C)$ is the number of instances where the i-th rule *is* followed, and i corresponds to the rules listed in section 4.2.

As genome representation satisfied rules 1-4, 19 and 20 in section 4.2, the fitness function contains the rest. That is,

$$S_R \subseteq \{r_5, r_6, \ldots, r_{18}\}.$$

4.3.5 Algorithm

Putting the above described components together we obtain the final algorithm, given in algorithm 2. The first step of our algorithm is computing the fitness values for each individual. A fitness threshold, FT, is set to stop the evolution once any individual reaches a certain fitness value. In addition, a stagnate threshold is introduced to stop evolution when the top fitness value has not been improved for some number of generations. This effectively terminates the algorithm when it cannot escape a local minimum.

In the next step we introduce the concept of *invitations*, where the most fit individual(s) move to the next generation untouched. Selection then selects individuals, adding them to the new population. Finally, crossover and mutation are applied to ensure $|\mathsf{PP}_{new}| = |\mathsf{PP}_{cur}|$.

4.3.6 Implementation

The GA is implemented in C++ for performance reasons. OpenMP is used to implement a parallel version of our algorithm. As for output at the end of a run, both CSV and MIDI files are generated. The outputted CSV files contain the fitness of every individual, averages, and bests for each generation. MIDI files of each individual in the final population are also output, with the most fit individual found marked.

4.3.6.1 Parallelism

The fitness function and each of the genetic operators have parallel versions which utilize multiple CPU cores. The parallel version of the fitness function, mutation, and crossover operators is done in a data-parallel fashion with simple parallel for loops: each CPU core is delegated a similar or equal number of individuals to score, mutate, or crossover. These are implemented by OpenMP's parallel for loop.

Since an individual is removed from the selection pool each time the selection operator is applied, it cannot employ the same data-parallel design as the other components. However, both rank and tournament selection use sorts which can be parallelized. Tournament selection

Algorithm 2: GA for Music Composition

```
Input: $\mathsf{PP_{ini}}$, $\mathsf{NofG}$, $\mathsf{NofS}$, $\mathsf{NofC}$, $\mathsf{NofM}$, $\mathsf{NofP}$, $\mathsf{FT}$, $\mathsf{ST}$.
Output: Most Fit Composition Found
Data Structures: $\mathsf{PP_{cur}}, \mathsf{PP_{new}}$; //sets of compositions

$\mathsf{PP_{cur}} = \mathsf{PP_{ini}}$;
for NofG do
    $\mathsf{PP_{new}} = \emptyset$;
    for all $\mathsf{C_u} \in \mathsf{PP_{cur}}$ do
        Compute $\mathsf{F(C_u)}$;
        if $\mathsf{F(C_u)} \geqslant \mathsf{FT}$ then terminate by returning $\mathsf{C_u}$;
    end
    for NofI do
        $\mathsf{C_u}$ = most fit composition in $\mathsf{PP_{cur}}$;
        $\mathsf{PP_{cur}} \xRightarrow{\mathsf{C_u}} \mathsf{PP_{new}}$;
    end
    for NofS do
        $\mathsf{PP_{cur}} \xRightarrow{\Omega(\mathsf{PP_{cur}})} \mathsf{PP_{new}}$;
    end
    for NofC do
        Choose $\mathsf{C_u}$ and $\mathsf{C_v}$ from $\mathsf{PP_{new}}$;
        Add $\otimes(\mathsf{C_u}, \mathsf{C_v})$ to $\mathsf{PP_{new}}$;
    end
    for NofM do
        Choose $\mathsf{C_u}$ from $\mathsf{PP_{new}}$;
        Add $\Pi(\mathsf{C_u})$ to $\mathsf{PP_{new}}$;
    end
    $\mathsf{PP_{cur}} = \mathsf{PP_{new}}$;
    if Number of generations where top fitness has not been improved $\geqslant \mathsf{ST}$ then break;
end
return Most fit composition in $\mathsf{PP_{cur}}$;
```

$X \xRightarrow{z} Y$ - Move z from X to Y
$\mathsf{PP_{ini}}, \mathsf{PP_{cur}}, \mathsf{PP_{new}}$ - initial, current, new population sets
FT - fitness threshold
ST - stagnation threshold
NofI - number of invitations
NofG - number of generations
NofS - number of selections
NofC - number of crossovers
NofM - number of mutations
$\Omega(x)$ - subset of x obtained by selection operation Ω

also has a shuffle function that is also able to be run in parallel. We use the parallel functions available in C++'s standard library for shuffling and sorting.

4.3.6.2 Parameters

There are many parameters for our algorithm scattered across many of the different components. The parameters must meet certain criteria before the algorithm is run with them. A list of every parameter and criteria is consolidated here, separated by components of the GA.

General Parameters:

- $|PP|$; the population size.
- $NofI$; the number of invitations.
- $NofG$; the number of generations.
- $NofS$; the number of selections.
- $NofM$; the number of mutations.
- $NofC$; the number of crossovers.
- FT, where $0 \leqslant FT \leqslant 1$; the fitness threshold.
- ST, where $1 \leqslant ST \leqslant NofG$; the stagnation threshold.

With the following restriction:

$$NofS + NofM + NofC = |PP|.$$

Representation Related Parameters:

- p; the number of parts.
- l; the number of measures.
- U; the shortest possible duration.

- $K, \forall k \in K, 0 \leqslant k \leqslant 11$; the allowed pitch classes (key/scale).

- T; the time signature.

With the following restrictions:

$$T_n \geqslant 1, \text{ and } \forall z \in \mathbb{Z}_{\geqslant 0}, U = 2^z, \text{ and } T_d = 2^z.$$

Population Initialization Parameters:

- $B = \{\{lb_0, ub_0\}, \{lb_1, ub_1\}, \ldots, \{lb_p, ub_p\}\}$; the lower and upper bounds for each part.

Selection Related Parameters:

- $|\tau|$, where $|\tau| \leqslant |PP|$; the tournament size (tournament selection only).

- ρ, where $0 \leqslant \rho \leqslant 1$; the tournament constant (tournament selection only).

- SP; the selection pressure (rank selection only)

Mutation Related Parameters:

- p_{rt}; the probability of choosing the random transposition sub-operator. $0 \leqslant p_{rt} \leqslant 1$.

- p_s; the probability of choosing the split sub-operator. $0 \leqslant p_s \leqslant 1$.

- p_m; the probability of choosing the merge sub-operator. $0 \leqslant p_m \leqslant 1$.

- p_r; the probability of choosing the repeat sub-operator. $0 \leqslant p_r \leqslant 1$.

With the following restriction:

$$p_{rt} + p_s + p_m + p_r = 1.$$

Chapter 5

Experimental Study

To illustrate the performance of our algorithm and quality of music it is capable of generating, we conducted a performance study and an online survey, respectively. The performance study is described in section 5.1, and the online survey is presented in section 5.2.

5.1 Performance Study

An experiment was run on a 6-core processor clocked at 4GHz with the following parameters:

- $|PP| = 100$,
- $NofI = 2$,
- $NofG = 100$,
- $NofS = 25$,
- $NofM = 50$,
- $NofC = 25$,
- $FT = 1$,
- $ST = 100$,
- $p = 3$,

- $l = 4$,
- $U = 64$,
- $T = \langle 4, 4 \rangle$,
- $K = \{0, 2, 4, 5, 7, 9, 11\}$,
- $B = \{\{C2, C4\}, \{C3, G4\}, \{C4, D5\}\}$,
- Deterministic tournament selection ($\rho = 1$) with $|\tau| = 10$, and
- $p_{rt} = 0.45, p_s = 0.25, p_m = 0.25, p_r = 0.05$.

Figure 5.1 shows this experiment's run times for each component with both single threaded and parallel versions. When ran in parallel, the fitness function, mutation, and crossover components saw a 2.5x, 2.2x, and 2x reduction in run-time respectively. By far the largest bottleneck on overall run time is the fitness function, since it exhaustively iterates through every composition. Selection saw a negligible speedup, most likely due to its inability to run data-parallel.

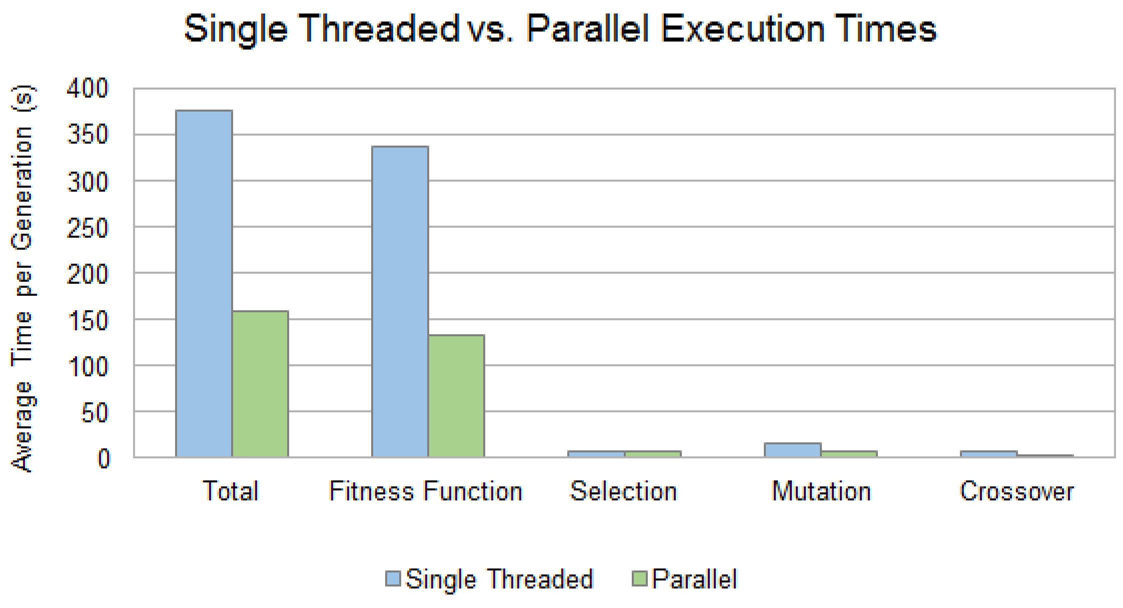


Figure 5.1: Graph containing average execution time per generation for each of the GA components with the single threaded and parallel versions of the algorithm.

5.2 Online Survey

Two surveys were conducted to test whether Savage et al.'s musical universals [78] and Huron's non-traditional rules [48] are more effective than just Huron's traditional rules. Each participant was asked a demographic question, Q_0, and then given five comparisons between two compositions, $Q_1, Q_2, \ldots, Q_5$, where they decide on which they enjoyed more or choose a neutral position. For each comparison, the two compositions had different types which were both unknown to the participants and in random order. These composition types refer to how that particular composition was generated.

The sole difference in generation was the fitness function, specifically the set of rules which it used, S_R. The first type was generated using a fitness function with only the traditional rules from Huron's work [48] (type **R2**), and the second with all of the rules listed in section 4.2 (all of Huron's rules [48] as well as Savage et al.'s musical universals [78]; types **R1**, **R2**, and **R3**).

$$S_R^1 = \{r_7, r_8, \ldots, r_{11}\}, \text{ and } S_R^2 = \{r_5, r_6, \ldots, r_{18}\} \text{ respectively.}$$

5.2.1 Results

The first survey had compositions generated with the following parameters:

- $|PP| = 500$,
- $NofI = 5$,
- $NofG = 500$,
- $NofS = 75$,
- $NofM = 300$,
- $NofC = 125$,
- $FT = 1$,

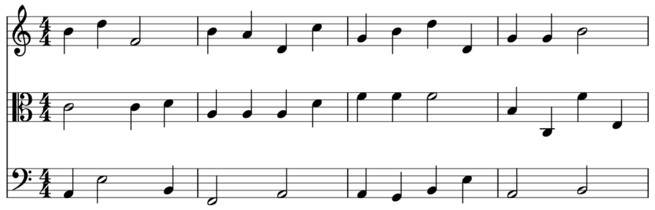
- $ST = 500$,
- $p = 3$,
- $l = 8$,
- $U = 64$,
- $T = \langle 4, 4 \rangle$,
- $K = \{0, 2, 4, 5, 7, 9, 11\}$,
- $B = \{\{C2, C4\}, \{C3, G4\}, \{C4, D5\}\}$,
- Deterministic tournament selection ($\rho = 1$) with $|\tau| = 10$, and
- $p_{rt} = 0.45, p_s = 0.25, p_m = 0.25, p_r = 0.05$.

As expected, there are big differences in fitness values for the two types of compositions; the first had near perfect fitness values (all above 0.995) whereas the second had a median of only 0.932. Overall, respondents preferred the composition generated using just the traditional rules.

Respondent fatigue was thought to confound the results and so a second survey was conducted, with compositions generated using the same parameters but shorter in length ($l = 4$). The first composition type again had near perfect fitness values, while the second had a median of 0.935. For this survey, respondents preferred compositions generated using all of the rules. The overall results for both surveys separated by the two most common regions are listed in table 5.1. Results for each comparison of the two surveys is shown in table 5.2.

Comparison 3 (Q3) of survey 2 is the most drastic difference in terms of composition preference, featuring a 63%/22.2% preference in favor of the type 2 composition. The compositions involved in this comparison are shown in figure 5.2. The case where type 1 is preferred most of the time is in comparison 2 of the same survey, where 55.6% of participants preferred the type 1 composition over type 2. These compositions are shown in figure 5.3.

(a) Type 1 composition with 1.0 fitness.



(b) Type 2 composition with 0.934 fitness.

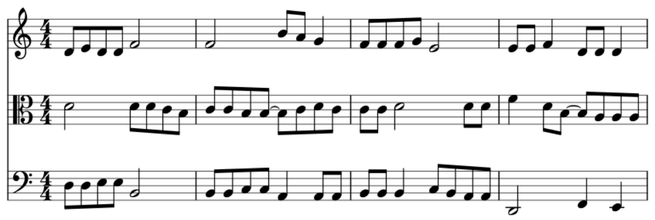
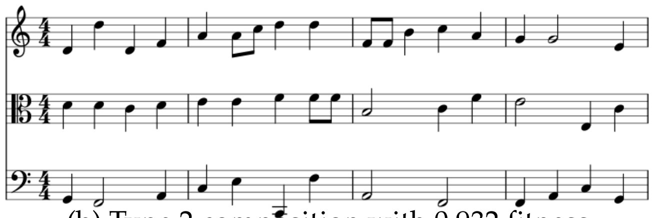


Figure 5.2: Compositions in Q3 of Survey 2.

(a) Type 1 composition with 0.995 fitness.



(b) Type 2 composition with 0.932 fitness.

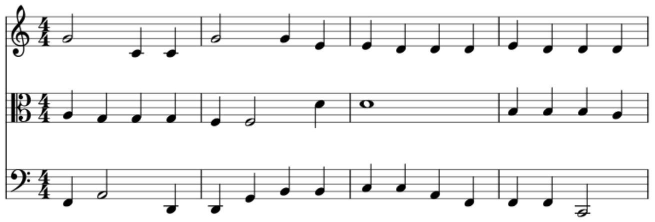


Figure 5.3: Compositions in Q3 of Survey 2.

Region	Composition Preference			Total Responses
	Type C2	Type C3	Neutral	
Survey 1				
North Americans	40.6%	41.7%	17.7%	175
Europeans	40%	24%	36%	75
Overall	38.9%	37.1%	23.9%	280
Survey 2				
North Americans	36%	52%	12%	100
Europeans	32%	24%	44%	25
Overall	35.6%	47.4%	17%	135

Table 5.1: Overall Survey Responses.

Question	*Survey 1*			*Survey 2*		
	Composition Preference			Composition Preference		
	Type C2	Type C3	Neutral	Type C2	Type C3	Neutral
Q1	51.8%	32.1%	16.1%	51.9%	25.9%	22.2%
Q2	37.5%	37.5%	25%	**55.6%**	**33.3%**	**11.1%**
Q3	23.2%	50%	26.8%	**22.2%**	**63%**	**14.8%**
Q4	39.3%	30.4%	30.4%	22.2%	59.3%	18.5%
Q5	42.9%	35.7%	21.4%	25.9%	55.6%	18.5%

Table 5.2: Survey Responses to the Five Comparisons (Q2-Q6).

5.2.2 Analysis

Changes in responses between the two surveys suggest that: (i) the length of these compositions effects respondent's perception of them; (ii) the participants are less fatigued in the second survey; (iii) the small differences in fitness values of the second type of composition (medians of 0.932 in the first survey and 0.935 in the second) make a difference; and/or (iv) results are confounded by uncontrolled aspects of music. As evolution is no different in the four and eight measure case, (i) is unlikely. A fitness value difference of 0.003 in the type 2 compositions (iii) is also unlikely to generate any measurable difference.

Fatigue (ii) is likely to happen in the first survey, as each composition is 32 seconds long, totalling over 5 minutes of listening, compared to the second survey's 16 second compositions. Moreover, these compositions were played by computer, which lacks the compelling expressiveness of a human performer.

Examination of the individual comparisons provides an explanation related to beats; compositions that were heavily preferred, type 1 in Q2 of survey 2 (shown in figure 5.3) and type 2 in Q3 of survey 2 (shown in figure 5.2), have offbeat notes, while their counterparts do not. No offbeat notes create an illusion of a chord sounding at every beat, having no melody stand above it. This suggests there is a musical rule related to beats and the need for off-beat notes that is not being accounted for.

Chapter 6

Implications of Automated Music Composition

In science, innovation takes front stage. The questions asked rarely pertain to why we are innovating and how such innovation will effect the world. There are many issues and ethical considerations with the continuation of this work and with automated composition as a whole. These are consolidated into three topics all revolving around automation: disruption of the job market, disruption of the legal system, and the goals of automated composition. Many of the topics here may call into question the purpose of the work we have presented. Nevertheless, discussion of them is very rare and they are extremely important.

6.1 Disruption of the Job Market

The ability to automate increasingly complex tasks with AI has cultivated a genuine fear for ones job and therefore livelihood in many industries. The fear of automation in the music industry is summarized well in a 1991 interview with world-renowned bassist Anthony Jackson [16]. Despite the primitive state of technology in music at that time, Anthony describes that it "effects everybody" and that "everyone's work has gone down."

There have been significant advancements in AC since then and opposition from both musicians and the public are still present [26]. In academia, concern has not been raised until recently.

For instance, Sturm et al. find that music AI not only negatively impacts job seekers, but also is “likely to have adverse effects on innovation in the music-ecosystem” [81]. They urge those involved in such research to “consider the longer term effects of their decisions.”

Sturm et al. also state that musicians and composers should be involved in the making of music AI [81]. Researchers involved in AC often have the same view, opting to create systems that aid composers and musicians rather than replacing them. These systems are usually highly specialized, tailored for a specific individual, such as the work of Xenakis [92], Cope [17, 18], and Eigenfeldt [28–31]. Moreover, researchers seldom collaborate with professional musicians and composers. A step in the right direction is taken by Chen and Chang [15], which brings together researcher, composer, and musician. Their generation system is developed with a professional composer and the generated results are played by a professional musician.

6.2 Disruption of the Legal System

Who owns a machine-generated artwork and whether it is protected by law are still open questions, with only a few countries enacting protection for computer-generated works [81]. For instance, the composition system AIVA[1], dubbed as the first ‘electronic composer’ whom is recognized by a music society (the SACEM[2]), it is not clear whether those whom created AIVA are entitled to the revenue made from its compositions or if they have copyright or neighbouring rights. In the case of a user composing with a music generation system, such as Computoser[3], AuralFractals[4], DeepBeat[5], Amper Music[6], The Uncanny MusicBox[7], and many others, it is unclear whether the designer of these systems, the user, both, or neither have copyright or neighbouring rights to the generated output.

For many scholars, autonomously generated music is not eligible for copyright protection [81]. However, many generation systems have varying degrees of autonomy, and it is unclear

[1]https://aiva.ai/
[2]SACEM is a professional association that handles the royalties, promotion, and supporting of artists.
[3]http://computoser.com/
[4]https://www.auralfractals.net/
[5]http://deepbeat.org/
[6]https://www.ampermusic.com/
[7]http://uncannymusicbox.com/

what amount of human intervention is necessary for a work to be protected. Algorithms that require data sets may also be found to breach copyright if the data set includes music not yet in public domain. The European Union (EU) has allowed scientific research to fully use such legally acquired data sets, with other uses prohibited only if rightholders have restricted use [81]. The research of Sturm et al. [81] and Deltorn and Macrez [23] offer detailed analyses of the legal implications on specific music generation systems.

6.3 End-Goal

The purpose of creating a music generation system is often said to be to aid composers, yet most of the work with this objective is tailored specifically to a single composer. Moreover, the act of creating a general music generation system to aid composers is extremely similar to creating a fully autonomous one, with the only difference being the number of decisions made by the computer and not by the composer. As the field improves, more and more decisions are able to be made adequately by the computer and researchers have largely focused on creating more autonomous systems.

Complete automation is another frequently re-occurring objective of AC, especially with research done by large companies such as Google. The benefits for large companies are clear: company sound logos [72], video game soundtracks [34], film soundtracks, and other associated music can be generated instead of hiring musicians, composers, music producers, etc. Companies offering enterprise-level music generation systems already exist: AIVA[2], Amper Music[7], and others. Depending on how countries issue protection for computer-generated music, large-scale investment into this technology will have an adverse effect on those involved in the music industry.

[2]https://aiva.ai/
[7]https://www.ampermusic.com/

Chapter 7

Conclusions and Future Prospects

7.1 Approach

In summary, our approach to the problem of generating music via algorithm was to use domain knowledge rooted in music-psychology rather than traditional music theory alone. The novelty of this approach comes from the fact that virtually all research in the field uses music theory for their domain knowledge. This direction was taken to provide the algorithm with a stronger foundation, as music-psychological findings are empirically studied whereas the majority in music theory are not.

Music-psychological domain knowledge was converted into a set of rules, which were then built into a GA. These rules were distributed throughout the algorithm; the genome representation was defined to satisfy some of the rules, with most being built into the fitness function. In the later addition of chord progressions, domain knowledge was built in the population initialization phase as well by ensuring the basic structure of chords before the running of the GA.

Genome representation was designed to closely resemble staff notation, offering a solid border between musical elements. Moreover, this representation avoids the over-simplification problems of pitch-only and rhythm-only research through inclusion of the fundamental aspects of music. More extraneous elements, such as dynamics, key-modulations, etc. are not present in our system.

In addressing the fitness computation bottleneck, a parallel version of our algorithm was created. Parallelization was done with a simple data-parallel model, allowing the workload to be evenly distributed across each CPU core. All genetic operators as well as population initialization were implemented in parallel.

7.2 Findings

Two surveys were conducted to test the efficacy of adding non-traditional rules to the fitness function. The first survey suggested that adding non-traditional rules did not increase the quality of generated output, while the second suggest that it did. With no differences between the two surveys other than the length of the generated compositions, it is thought that user fatigue could have effected the first survey's results. Diving deeper, however, we found that preference in compositions was heavily determined by the number of eighth notes; compositions with little to no eighth notes were not preferred. These findings indicate that there are additional rule(s) that need to be identified and incorporated into our system.

To test the efficacy of the parallelization, both sequential and CPU-parallel implementations were tested. Echoing previous work, computing fitness was still found to be the bottleneck of the system, however its run-time was significantly reduced by adopting a simple data-parallel model. Mutation and crossover computations saw similar speedups, although their run-time pales in comparison to that of the fitness function.

7.3 Additions

There have been a number additions to the algorithm since this survey was conducted. The algorithm tended produce compositions with a high number of dissonant intervals, particularly sharp dissonances, such as minor 2nds, major 7ths, and tritones, since there was no rules involving harmony besides avoiding tonally fused intervals. To remedy this, the ability for the user to input a chord progression has been added, with there being one chord per measure. Chord detection was done in the fitness function by checking whether the tones present are those in the chord. In the case of seventh chords, fifths are not checked as it is often the tone left out of more complex chords in Western harmony. Population initialization was also tweaked to ensure chords were correct before being mutated.

Research involving GAs have called for a new type of mutation operator, one that is informed about where to mutate a particular individual instead mutating purely at random [22]. Attempts were made at implementing this type of mutation operator in naïve way with little success. This was implemented by keeping track of positions in the composition where rules were violated. In this way, the fitness function was informing the mutation operator of the locations to mutate. It is likely that mutations produce a ripple effect throughout the composition and therefore must be more sophisticated to achieve positive results.

7.4 Future Directions

To further tackle the fitness computation bottleneck, GPU-based parallelization should be used. However, due to the complexity of the representation and the fitness function, such parallelization may be considerably difficult given the limitations of current GPU processing. Another way to approach the problem is to use a more efficient genome representation. This, in turn, may further complicate the fitness function and fog the boundary between musical elements.

In terms of domain knowledge, there is a plethora of usable music-psychology research available that can readily be implemented in our algorithm and other AC research. Chapter 2 reviews a handful of many promising research in music-psychology. Rules involving harmony are particularly pertinent to our system as there are none besides the later addition of user-defined chord progressions. However, implementing more rules into the fitness function will likely slow evolution further, as evidenced by the fitness discrepancy in using only traditional rules versus using all rules in our experiments (1 vs. 0.94 respectively). A more successful approach may be to distribute rules throughout the GA more, such as in mutation, crossover, representation, or population initialization, or to avoid a rule-based approach altogether.

Recent critiques of musical universals offer a view that there are no true universals in music. In the context of AC, this implies that a truly objective composition system is not possible, at least without considering the culture one grows up in. Considering this, we hope to explore culture- or genre-specific additions to our system and avoid claims of objectivity.

The final and perhaps most important direction for our work is to seriously consider the implications of AC research on society. AC is improving faster than ever, with musicians and composers already losing work due to these systems. We must ensure that true autonomy is not the goal of AC, instead focusing on aiding and collaborating with musicians and composers.

ACRONYMS

AC Algorithmic Composition. 7–9, 12, 18, 20–22, 24–29, 49–51, 54, 55

AI Artificial Intelligence. 7, 9, 49, 50

ANN Artificial Neural Network. 24

ASA Auditory Scene Analysis. 13, 14

DNNs Deep Neural Networks. 9

EA Evolutionary Algorithm. 29

EAs Evolutionary Algorithms. 9

GA Genetic Algorithm. 5, 9, 10, 20–23, 26, 28, 29, 31, 38, 40, 43, 52, 54

GAs Genetic Algorithms. 20, 22, 24, 27, 28, 54

IGA Interactive Genetic Algorithm. 23, 24

IGAs Interactive Genetic Algorithms. 22

ILLIAC Illinois Automatic Computer. 8, 27

ITPRA Imagination-Tension-Prediction-Reaction-Appraisal. 13

KE Kinetic Engine. 9, 24, 25, 27

MIDI Musical Instrument Digital Interface. 21

ML Machine Learning. 9, 20, 22, 29

NN Neural Network. 22, 23

NNs Neural Networks. 23, 24

RNN Recurrent Neural Network. 24

Bibliography

[1] Paolo Ammirante and Frank A Russo, *Low-skip bias*, Music Perception: An Interdisciplinary Journal **32** (2015), no. 4, 355–363.

[2] Christopher Ariza, *An open design for computer-aided algorithmic music composition*, Universal-Publishers, 2005.

[3] Christopher Ariza, *Two pioneering projects from the early history of computer-aided algorithmic composition*, Computer Music Journal **35** (2011), no. 3, 40–56.

[4] Robert Baker, *Musicomp: Music simulator-interpreter for compositional procedures for the ibm 7090*, no. 9, Experimental Music Studio, 1963.

[5] Konstantinos Bakogiannis and George Cambourakis, *Semiotics and memetics in algorithmic music composition*, Technoetic Arts **15** (2017), no. 2, 151–161.

[6] Gilberto Bernardes, Caros Guedes, and Bruce Pennycook, *Style emulation of drum patterns by means of evolutionary methods and statistical analysis*, Proceedings of the Sound and Music Conference, 2010, pp. 1–4.

[7] John Biles, Peter Anderson, and Laura Loggi, *Neural network fitness functions for a musical iga*, (1996).

[8] John A Biles, *Genjam: A genetic algorithm for generating jazz solos*, ICMC, vol. 94, 1994, pp. 131–137.

[9] John A Biles, *Autonomous genjam: eliminating the fitness bottleneck by eliminating fitness*, Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop Program, San Francisco, 2001.

[10] Tanor L Bonin, Laurel J Trainor, Michel Belyk, and Paul W Andrews, *The source dilemma hypothesis: Perceptual uncertainty contributes to musical emotion*, Cognition **154** (2016), 174–181.

[11] Albert S Bregman, *Auditory scene analysis: The perceptual organization of sound*, MIT press, 1994.

[12] Steven Brown and Joseph Jordania, *Universals in the world's musics*, Psychology of Music **41** (2013), no. 2, 229–248.

[13] Yuri Broze and David Huron, *Is higher music faster? pitch–speed relationships in western compositions*, Music Perception: An Interdisciplinary Journal **31** (2013), no. 1, 19–31.

[14] Anthony R Burton and Tanya Vladimirova, *Genetic algorithm utilising neural network fitness evaluation for musical composition*, Artificial Neural Nets and Genetic Algorithms, Springer, 1998, pp. 219–223.

[15] Chun-yien Chang and Ying-ping Chen, *Fusing creative operations into evolutionary computation for composition: From a composer's perspective*, 2019 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2019, pp. 2113–2120.

[16] Jazz In Concert, *Jazz in concert 1991 with tony lakatos, peter o´mara, david witham, anthony jackson terri lyne carrington*, 1991.

[17] David Cope, *Experiments in musical intelligence*, Proceedings of the International Computer Music Conference. San Francisco, 1987.

[18] David Cope and Melanie J Mayer, *Experiments in musical intelligence*, vol. 12, AR editions Madison, WI, 1996.

[19] David Cope, *Experiments in musical intelligence (emi): Non-linear linguistic-based composition*, Journal of New Music Research **18** (1989), no. 1-2, 117–139.

[20] David Cope, *Algorithmic composition [re] defined*, Proceedings of the International Computer Music Conference, 1993, pp. 23–25.

[21] Alan RR de Freitas and Frederico Gadelha Guimarães, *Originality and diversity in the artificial evolution of melodies*, Proceedings of the 13th annual conference on Genetic and evolutionary computation, ACM, 2011, pp. 419–426.

[22] Francisco Fernandez de Vega, *Revisiting the 4-part harmonization problem with gas: A critical review and proposals for improving*, Evolutionary Computation (CEC), 2017 IEEE Congress on, IEEE, 2017, pp. 1271–1278.

[23] Jean-Marc Deltorn and Franck Macrez, *Authorship in the age of machine learning and artificial intelligence*, (2019).

[24] Patrick Donnelly and John Sheppard, *Evolving four-part harmony using genetic algorithms*, European Conference on the Applications of Evolutionary Computation, Springer, 2011, pp. 273–282.

[25] Ben Duane, *Auditory streaming cues in eighteenth-and early nineteenth-century string quartets*, Music Perception: An Interdisciplinary Journal **31** (2013), no. 1, 46–58.

[26] Michael Edwards, *Algorithmic composition: computational thinking in music*, Communications of the ACM **54** (2011), no. 7, 58–67.

[27] Hauke Egermann, Marcus T Pearce, Geraint A Wiggins, and Stephen McAdams, *Probabilistic models of expectation violation predict psychophysiological emotional responses to live concert music*, Cognitive, Affective, & Behavioral Neuroscience **13** (2013), no. 3, 533–553.

[28] Arne Eigenfeldt, *Kinetic engine: toward an intelligent improvising instrument*, Proceedings of the Sound and Music Computing Conference, 2006, pp. 97–100.

[29] Arne Eigenfeldt, *Drum circle: Intelligent agents in max/msp.*, ICMC, 2007.

[30] Arne Eigenfeldt, *Corpus-based recombinant composition using a genetic algorithm*, Soft Computing **16** (2012), no. 12, 2049–2056.

[31] Arne Eigenfeldt, *The evolution of evolutionary software: intelligent rhythm generation in kinetic engine*, Workshops on Applications of Evolutionary Computation, Springer, 2009, pp. 498–507.

[32] David Fallows, *The oxford companion to music*, Oxford University Press, 2016.

[33] Richard Fox and Robert Crawford, *A hybrid approach to automated music composition*, Artificial Intelligence Perspectives in Intelligent Systems, Springer, 2016, pp. 213–223.

[34] Mikael Fridenfalk, *Algorithmic music composition for computer games based on l-system*, Games Entertainment Media Conference (GEM), 2015 IEEE, IEEE, 2015, pp. 1–6.

[35] David E Goldberg and Kalyanmoy Deb, *A comparative analysis of selection schemes used in genetic algorithms*, Foundations of genetic algorithms, vol. 1, Elsevier, 1991, pp. 69–93.

[36] David E Goldberg and John H Holland, *Genetic algorithms and machine learning*, Machine learning **3** (1988), no. 2, 95–99.

[37] Alvin I Goldman, *The psychology of folk psychology*, Behavioral and Brain sciences **16** (1993), no. 1, 15–28.

[38] Dunwei Gong, Xin Yao, and Jie Yuan, *Interactive genetic algorithms with individual fitness not assigned by human.*, J. UCS **15** (2009), no. 13, 2446–2462.

[39] Stephen A Hedges, *Dice music in the eighteenth century*, Music & Letters **59** (1978), no. 2, 180–187.

[40] Dorien Herremans, Ching-Hua Chuan, and Elaine Chew, *A functional taxonomy of music generation systems*, ACM Computing Surveys (CSUR) **50** (2017), no. 5, 69.

[41] Lejaren Hiller, *Programming a computer for music composition*, Computer Applications in Music. Morgantown: West Virginia University Library (1967), 65–88.

[42] Lejaren Hiller, *Some compositional techniques involving the use of computers*, 1969, pp. 71–83.

[43] Lejaren Hiller, *Composing with computers: A progress report*, Computer Music Journal **5** (1981), no. 4, 7–21.

[44] Lejaren A Hiller Jr and Leonard M Isaacson, *Musical composition with a high speed digital computer*, Audio Engineering Society Convention 9, Audio Engineering Society, 1957.

[45] Robert Hinterding, Harry Gielewski, and Thomas C Peachey, *The nature of mutation in genetic algorithms.*, ICGA, 1995, pp. 65–72.

[46] Andrew Horner and David Goldberg, *Genetic algorithms and computer-assisted music composition*, Proceedings of the Fourth International Conference on Genetic Algorithms, 1991, pp. 479–482.

[47] Yu-Lun Hsu, Chi-Po Lin, Bo-Chen Lin, Hsu-Chan Kuo, Wen-Huang Cheng, and Min-Chun Hu, *Deepsheet: A sheet music generator based on deep learning*, Multimedia & Expo Workshops (ICMEW), 2017 IEEE International Conference on, IEEE, 2017, pp. 285–290.

[48] David Huron, *Tone and voice: A derivation of the rules of voice-leading from perceptual principles*, Music Perception: An Interdisciplinary Journal **19** (2001), no. 1, 1–64.

[49] David Brian Huron, *Sweet anticipation: Music and the psychology of expectation*, MIT press, 2006.

[50] Natasha Jaques, Shixiang Gu, Richard E. Turner, and Douglas Eck, *Generating music by fine-tuning recurrent neural networks with reinforcement learning*, Deep Reinforcement Learning Workshop, NIPS, 2016.

[51] Daniel D Johnson, *Generating polyphonic music using tied parallel networks*, International Conference on Evolutionary and Biologically Inspired Music and Art, Springer, 2017, pp. 128–143.

[52] Anna Jordanous, *A fitness function for creativity in jazz improvisation and beyond.*, ICCC, 2010, pp. 223–227.

[53] Patrik N Juslin and Daniel Västfjäll, *Emotional responses to music: The need to consider underlying mechanisms*, Behavioral and brain sciences **31** (2008), no. 5, 559–575.

[54] Johann Philipp Kirnberger, *Der allezeit fertige menuetten-und polonoisenkomponist*, Berlin: Germany Winter (1757).[doi¿ 10.1093/acprof: oso/9780195148367.001. 0001] (1757).

[55] GM Koenig, *Project two*, Electronic Music Reports **3** (1970).

[56] Gottfried M Koenig, *Project one: A programme for musical composition*, Electronic Music Reports of the Institute of Sonology **2** (1969).

[57] Rakesh Kumar and Jyotishree, *Blending roulette wheel selection rank selection in genetic algorithms*, International Journal of Machine Learning and Computing (2012), 365–370.

[58] Daniel J Levitin, *This is your brain on music: The science of a human obsession*, Penguin, 2006.

[59] Ryan Lichtenwalter, Katerina Lichtenwalter, and Nitesh V Chawla, *Applying learning algorithms to music generation.*, IICAI, 2009, pp. 483–502.

[60] Chien-Hung Liu and Chuan-Kang Ting, *Polyphonic accompaniment using genetic algorithm with music theory*, Evolutionary Computation (CEC), 2012 IEEE Congress on, IEEE, 2012, pp. 1271–1278.

[61] Chien-Hung Liu and Chuan-Kang Ting, *Fusing flamenco and argentine tango by evolutionary composition*, 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2017, pp. 2645–2652.

[62] Chien-Hung Liu and Chuan-Kang Ting, *Computational intelligence in music composition: A survey*, IEEE Transactions on Emerging Topics in Computational Intelligence **1** (2017), no. 1, 2–15.

[63] Henrique Barros Lopes, Flávio Vinícius Cruzeiro Martins, Rodrigo TN Cardoso, and Vinícius Fernandes dos Santos, *Combining rules and proportions: A multiobjective approach to algorithmic composition*, Evolutionary Computation (CEC), 2017 IEEE Congress on, IEEE, 2017, pp. 2282–2289.

[64] Róisín Loughran, James McDermott, and Michael O'Neill, *Grammatical music composition with dissimilarity driven hill climbing*, International Conference on Evolutionary and Biologically Inspired Music and Art, Springer, 2016, pp. 110–125.

[65] Bill Manaris, Patrick Roos, Penousal Machado, Dwight Krehbiel, Luca Pellicoro, and Juan Romero, *A corpus-based hybrid approach to music analysis and composition*, Proceedings of the National Conference on Artificial Intelligence.

[66] Yotam Mann, *Ai duet*, Experiments with Google. See, https://experiments.withgoogle.com/ai/ai-duet (2016).

[67] Rachel Manzelli, Vijay Thakkar, Ali Siahkamari, and Brian Kulis, *An end to end model for automatic music generation: Combining deep raw and symbolic audio networks*, Proceedings of the Musical Metacreation Workshop at 9th International Conference on Computational Creativity, Salamanca, Spain, 2018.

[68] Dragan Matić, *A genetic algorithm for composing music*, Yugoslav Journal of Operations Research **20** (2010), no. 1, 157–177.

[69] Leonard B Meyer, *Emotion and meaning in music*, Ph.D. thesis, University of Chicago, Department of Music, 1954.

[70] Peter Mitrano, Arthur Lockman, James Honicker, and Scott Barton, *Using recurrent neural networks to judge fitness in musical genetic algorithms*, Proceedings of the 5th International Workshop on Musical Metacreation (MUME) at the 8th International Conference on Computational Creativity (ICCC), 2017.

[71] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, *Wavenet: A generative model for raw audio*, arXiv preprint arXiv:1609.03499 (2016).

[72] Noriko Otani, *Generation of a corporate sound logo based on symbiotic evolution*, 2019 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2019, pp. 2106–2112.

[73] George Papadopoulos and Geraint Wiggins, *Ai methods for algorithmic composition: A survey, a critical view and future prospects*, AISB Symposium on Musical Creativity, Edinburgh, UK, 1999, pp. 110–117.

[74] John Polito, Jason M Daida, and Tommaso F Bersano-Begey, *Musica ex machina: Composing 16th-century counterpoint with genetic programming and symbiosis*, International Conference on Evolutionary Programming, Springer, 1997, pp. 113–123.

[75] Curtis Roads, John Strawn, et al., *The computer music tutorial*, MIT press, 1996.

[76] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck, *A hierarchical latent vector model for learning long-term structure in music*, arXiv Preprint (2018).

[77] Martin Rohrmeier, *Musical expectancy: Bridging music theory, cognitive and computational approaches*, Zeitschrift der Gesellschaft für Musiktheorie [Journal of the German-speaking Society of Music Theory] **10** (2013), no. 2.

[78] Patrick E Savage, Steven Brown, Emi Sakai, and Thomas E Currie, *Statistical universals reveal the structures and functions of human music*, Proceedings of the National Academy of Sciences **112** (2015), no. 29, 8987–8992.

[79] Patrick E Savage, Adam T Tierney, and Aniruddh D Patel, *Global music recordings support the motor constraint hypothesis for human and avian song contour*, Music Perception: An Interdisciplinary Journal **34** (2017), no. 3, 327–334.

[80] Marco Scirea, Julian Togelius, Peter Eklund, and Sebastian Risi, *Metacompose: A compositional evolutionary music composer*, International Conference on Evolutionary and Biologically Inspired Music and Art, Springer, 2016, pp. 202–217.

[81] Bob LT Sturm, Maria Iglesias, Oded Ben-Tal, Marius Miron, and Emilia Gómez, *Artificial intelligence and music: Open questions of copyright law and engineering praxis*, Arts, vol. 8, Multidisciplinary Digital Publishing Institute, 2019, p. 115.

[82] Adam T Tierney, Frank A Russo, and Aniruddh D Patel, *The motor origins of human and avian song structure*, Proceedings of the National Academy of Sciences (2011), 201103882.

[83] Chuan-Kang Ting, Wu Chia-Lin, and Chien-Hung Liu, *A novel automatic composition system using evolutionary algorithm and phrase imitation*, IEEE Systems Journal **11** (2017), no. 3, 1284–1295.

[84] Brad Tofel, *Wayback'for accessing web archives*, Proceedings of the 7th International Web Archiving Workshop, 2007, pp. 27–37.

[85] Laurel J Trainor, *The origins of music in auditory scene analysis and the roles of evolution and culture in musical creation*, Philosophical Transactions of the Royal Society B: Biological Sciences **370** (2015), no. 1664, 20140089.

[86] Laurel J Trainor, Céline Marie, Ian C Bruce, and Gavin M Bidelman, *Explaining the high voice superiority effect in polyphonic music: Evidence from cortical evoked potentials and peripheral auditory models*, Hearing Research **308** (2014), 60–70.

[87] JA Vasconcelos, Jaime Arturo Ramirez, RHC Takahashi, and RR Saldanha, *Improvements in genetic algorithms*, IEEE Transactions on magnetics **37** (2001), no. 5, 3414–3417.

[88] S Wright, *Oxford university press and music publishing: A 75th anniversary retrospective*, BRIO-INTERNATIONAL ASSOCIATION OF MUSIC LIBRARIES- **35** (1998), 90–100.

[89] Jian Wu, Changran Hu, Yulong Wang, Xiaolin Hu, and Jun Zhu, *A hierarchical recurrent neural network for symbolic melody generation*, arXiv preprint arXiv:1712.05274 (2017).

[90] Iannis Xenakis, *Free stochastic music from the computer. programme of stochastic music in fortran*, Gravesaner Blätter **26** (1965), 54–92.

[91] Iannis Xenakis, Roberta Brown, and John Rahn, *Xenakis on xenakis*, Perspectives of New Music (1987), 16–63.

[92] Iannis Xenakis, *Formalized music: thought and mathematics in composition*, no. 6, Pendragon Press, 1992.

[93] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang, *Midinet: A convolutional generative adversarial network for symbolic-domain music generation*, Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR'2017), Suzhou, China, 2017.

[94] Xiaomei Zheng, Dongyang Li, Lei Wang, Yuanyuan Zhu, Lin Shen, and Yanyuan Gao, *Chinese folk music composition based on genetic algorithm*, Computational Intelligence & Communication Technology (CICT), 2017 3rd International Conference on, IEEE, 2017, pp. 1–6.