UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA "RENATO M. CAPOCELLI"

CORSO DI DOTTORATO IN INFORMATICA
XIV CICLO – NUOVA SERIE

ANNO ACCADEMICO 2014-2015

TESI DI DOTTORATO IN INFORMATICA

Computer Music Algorithms

Bio-inspired and Artificial Intelligence Applications

Tutor
prof. **Roberto De Prisco**

Candidato
Gianluca Zaccagnino

Coordinatore
prof. **Gennaro Costagliola**

Contents

List of Figures

Introduction

Music is one of the arts that have most benefited from the invention of computers. Originally, the term Computer Music was used in the scientific community to identify the application of information technology in music composition. It began over time to include the theory and application of new or existing technologies in music, such as sound synthesis, sound design, acoustic, psychoacoustic. Thanks to its interdisciplinary nature, Computer Music can be seen as the encounter of different disciplines, including music, computer science, etc. The Computer Music has its origin in Electronic Music. With the advent of personal computers has become a discipline used to describe any kind of music created with the help of computer tools. The first attempts to play music with the computer were CSIRAC, designed and built by Pearcey and Beard, and later a system to play popular musical melodies, developed by Hill in 1950s. Digital sound synthesis and algorithmic composition were originated thanks to two major developments dating back to the 1950s: Music I, developed by Max Mathews at Bell Laboratories, and the Illiac suite, developed by the musicians and chemists Hiller and Isaacson to compose music which resulted in a computer composed suite, the Illiac Suite for string quartet. Early computer music programs typically did not run in real time. Programs would run for hours or days, on multi-million-dollar computers, to generate a few minutes of music. This was due to: (1) the enormous amount of data needed to specify a function of sound pressure and therefore the need for a very fast program, and (2) the need for a simple but powerful language with which it is possible to describe the sequence of sounds. The first problem was solved with the introduction of faster microprocessors. Software tools were created to solve this second problem, and the use of these tools originated computer music. The real-time generation of computer music began with the work on FM synthesis of John Chowning from the 1960s to the 1970s and with and the advent of inexpensive digital chips and microcomputers. Music III by Max Mathews and Joan Miller introduced the concept of unit generator (units of sound generation) in the synthesis of sound. A unit generator is used to construct various algorithms for sound. Music V has had many descendants in the next 30 years among which Csound developed at the Massachusetts Institute of Technology (MIT) by Barry Vercoe. Computer music genera-

tion and performance has become possible thanks to advances in computing power and software for manipulation of digital media. Current-generation micro-computers are powerful enough to perform very sophisticated audio synthesis using a wide variety of algorithms and approaches. Nowadays the creation of music heavily depends. In the last years technology has redefined the way individuals can work, communicate, share experiences, constructively debate, and actively participate to any aspect of the daily life, ranging from business to education, from political and intellectual to social, and also in music activity, such as play music, compose music and so on. In this new context, Computer Music has become an emerging research area for the application of Computational Intelligence techniques, such as machine learning, pattern recognition, bio-inspired algorithms and so on. This thesis is concerned with the Bio-inspired and Artificial Intelligence Applications in the Computer Music. Specifically, I face several problems: (1) automatic composition of background music for games, films and other human activities, (2) definition of new interaction modalities during music performances by using hands without the support of a real musical instrument, (3) definition of a new bio-inspired approach for automatic music composition in a specific style, and (4) definition of new approaches for the learning of harmonic and melodic rules of classical music, by using visualization techniques. This thesis consists of five chapters. Chapter 1 describes the Music Algorithmic Composition problem, in Chapter 2 we introduce a new bio-inspired approach for automatic music composition, named Music Splicing Systems. Chapter 3 describes our solution to the problem of automatic music composition of background music, in Chapter 4 we propose a system to define a personalizable virtual music instrument. Finally, Chapter 5 describes some work in progress and provides some conclusions.

Chapter 1

Algorithmic composition

1.1 Introduction

The term *Algorithmic Composition* refers to any process that involves the use of a set of formal rules to compose music, without any human intervention.

Over the centuries, methods for algorithmic composition have been frequently used; for example, the compositions characterizing the entire contrapuntal Baroque music are in fact reducible to real processes of this type. In an attempt to make a classification of methods used in this context, one must take into account the radical distinction between the compositions made by means of nondeterministic procedures, such as stochastic methods, and those implemented by deterministic processes, such as formal grammars.

If one now poses the legitimate question about which musical approaches actually may be subsumed under the term Algorithmic Composition, an answer can be found in the investigation of some general definitions of *algorithm*.

The term *algorithm* may be derived from the Greek word *arithmos* (number) and also from the name of the Persian mathematician Abu Jafar Muhammad ibn Musa al-Khwarizmi[1] in two different ways. Al-Khwarizmi wrote a treatise on the calculation with Indian numerals,Around AD 820, which was translated into latin around 1120 AD as *Algorismi de numero Indorum*. In this translation, the author was given the latinized name *Algorismus*. In *Carmen de Algorismo*, a mathematical treatise in rhymes by French scholar Alexander de Villa Dei from AD 1220, calculating in the new numeral systems is also referred to with the term *Algorismus*[2]. Later, it was Grecized and became *Algorithmus*, being used as a general indication for a controlled procedure. Now, this term can be defined as:

[1] "Mohammed the father of Jafar and the son of Musa, the Khwarizmian," also referred to as "al Khowarizmi," lived ca. 780-850.

[2] In Latin also "alchorismus" and "algoarismus"; in Old French "algorisme" and "argorisme"; in Middle English "augrim" and "augrym."

- A set of mathematical instructions that must be followed in a fixed order, and that, especially if given to a computer, will help to calculate an answer to a mathematical problem [77].
- A systematic procedure that produces, in a finite number of steps, the answer to a question or the solution of a problem [105].
- A set of rules that must be followed when solving a particular problem [180].

A classical algorithm in general must terminate, i.e. produce solutions after running for a finite time of steps, and in general it must be deterministic, i.e. given a particular input, it will always produce the same correct output. However, some classes of algorithms and systems of algorithmic composition may produce output indefinitely, or involve probability-based decisions in the problem solving process. For example, if new material is to be made continuously audible for the arrangement of a sound installationan, an algorithm for the generation of musical structure will not terminate.

Approaches based on stochastic algorithms that produce different results with the same initial values, can be found in most systems designed for Algorithmic Composition.

An algorithm may be very generally described as a formalizable and abstract procedure which, applied to the generation of musical structure, determines the field of application of algorithmic composition. Accordingly, models for generating musical structure may be obtained from nearly every scientific discipline. Apart from that, interesting musical results may also be reached through simple but innovative compositional strategies or an appropriate mapping of data onto musical parameters. Procedures used in algorithmic composition are very well suited to the generation of musical structure, and on the other hand each represents a class of algorithms that can process or generate musical information in a specific way.

1.2 Early Approaches

Greek music. The idea of using formal processes and instructions for creating music historically dates back to ancient Greece. *Pythagoras* claimed the existence of a direct relationship between the laws of nature and the expression of sounds in music. This was thought to be inseparable from the *numbers*, that are considered the key elements of access to the entire universe, physical and spiritual. *Tolomeo*, the leading astronomer of his time, believed in mathematical correspondences between musical intervals and stars, and that certain notes and certain modes correspond to particular planets, their mutual distances and movements. This idea is also present in the works of *Plato*, namely the myth of the *music of the spheres*, which

speaks of the music produced by the motion of the planets. The theoretical applications of the various numbers and mathematical properties resulting from the nature are in fact the formalism on which the musicians of ancient Greece founded their music systems.

Micrologus. Algorithmic Composition has a century old tradition not only in occidental music history. Guido D'Arezzo is today mainly known for inventing *solfeggio* as well as due to his essential contributions to the development of musical notation, but he also worked in the field of algorithmic composition. Indeed, in one of his most significant works, *Micrologus*, D'Arezzo invented a method for the automatic conversion of text into melodic phrases.

He outlines a system for the automatic generation of melodies out of text material. Letters, syllables and components of a verse are mapped on tone pitches and melodic phrases (*neumes*), whereas groups of neumes are separated by caesurae. On the level of groups of neumes, the caesurae correspond to breathing pauses and can also be found in smaller groups in the form of pauses or held notes. The vowels in the text can be mapped on different tone pitches. The concrete design of the melody is subject to musical limitations that are treated by Arezzo in his theory on *Motus*.

Ars Magna. In the time of Raimundus Lullus, the *motet* became the dominant form of *polyphonic vocal music* in occidental music history. One of the principles of this genre, the *isorhythm*, invented by Philippe de Vitry and reaching its peak with Guillaume de Machaut consists of multiple repeating melodic (color) and rhythmic (talea) models that also interfere with each other and can occur in different proportions.

In *Ars Magna*, Lullus realizes the concept of a *computer (music) system*. The analogies to hardware and software, data memory, program, etc. are evident in the components, definitions and rules of the Ars Magna. Lullus creates a system that due to its *underlying structure* (hardware, corresponding to the diagrams), a *knowledge base* (data, corresponding to the definitions) as well as *application instructions* (software, corresponding to the interpretation rules) independently generates statements. Because of the given combination possibilities, some degree of chance is involved; however, due to the interpretation rules, the system provides coherent statements in the given context, whose exact interpretation is left to the user.

The circular statements inherent in the Ars Magna can actually also be found in a system of Algorithmic Composition, because any compositional premises can hardly be compared to axioms. When compositional work is considered under this point of view, the circular statement is as inherent in it as it is in the Ars Magna.

Combinatorics systems of Kircher. Kircher worked in the fields of astronomy, mathematics, medicine, music, mineralogy and physics as well as in linguistics, where he tried to decipher the Egyptian hieroglyphs with combinatorial methods.

In his exhaustive musicologist work *Musurgia Universalis* from 1650, Kircher developed a system of algorithmic composition. This system consists of three categories of labeled wooden sticks (syntagmas) on which both numbers and rhythmic values are engraved. Kirchers system allows for the automatic generation of contrapuntal compositions in the style of the contrapunctus simplex and floridus. In an advanced form, style-typical material of particular musical genres can be produced.

In another work proposed by Kircher, the *Arca Musarithmica*, four-lined number columns can be combined with four-voice rhythmic patterns by means of the syntagmas. The number of columns represent levels of different modes and are arranged in groups of 2 to 12 units. These units serve to correctly transfer text passages and represent one syllable each. Each class of tone pitch symbols of a particular size can be combined with a class of rhythmic patterns of the same size, finally producing four-voice movements in the style of the contrapunctus simplex. Because the number of voices differs in a movement of the contrapunctus flores, in this form of syntagma the voices are only combined with a selection of appropriate values.

Combinatorics is also used in another system of Kircher: the *mathematical organ* (Figure 1.1).

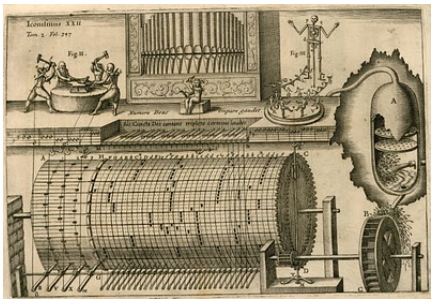


Figure 1.1: The mathematical organ by Kircher

This device is similar to a card index system and treats arithmetic, geometry, the building of forts, ecclesiastic time calculations, sundials, astronomy, astrology, cryptographs as well as music according to the above mentioned principles. For each special field there is a collection of labeled discs whose use (the possible combinations for solving the task), are explained in an enclosed booklet. Not all possible procedures are created by Kircher; the arithmetic part, for example, uses Napiers bones, for geometrical tasks the construction of the geometrical square (a common instrument of surveyors made up of a square frame with two scales and a rod to measure distances)

is applied. The importance of Kircher for the development of algorithmic thinking and finally the computer can be seen amongst other things in the fact that as an advancement of Lullus concept, a mechanical arrangement generates concrete outputs that may directly be used for solving a problem. Because all solutions possible in the system are also coded by the combination possibilities of the system, Kircher also paves the way for a comprehensive representation of knowledge in a chosen field of discourse.

1.3 Language and Music Generators

In the 18th century a game called *Musical Dice Game* became popular. In this game the outputs are completely independent from the evaluation of the user by exploring several combinatorial possibilities.

In this game, the user chose for every temporal unit a bar out of a particular table by rolling a dice, until a short piece of music of a musical genre had been produced. The principle of this game is simple, but effective: for each temporal position (mostly bar positions), a number of musical constellations (mostly bars) must be available out of which one can be chosen deliberately without running the danger of producing musical clashes by doing so. The application of the musical dice games is trivial; however, its design requires some talent in composing music, because for all musical possibilities attached one to another, not only the harmonic but also technical aspects of voice-leading have to be taken into consideration.

The first dice game is Johann Philipp Kirnbergers *Der allezeit fertige Menuettenund Polonaisencomponist* (The ever-ready minuet and polonaise composer), designed in 1757; up to the year 1812, at least 20 other creations of this type were built.

Even some of the most famous composers proposed their own versions of Dice games. For example, the *Einfall einen doppelten Contrapunct in der Octave von sechs Tacten zu machen ohne die Regeln davon zu wissen* (A method for making six bars of double counterpoint at the octave without knowing the rules) by Carl Philipp Emanuel Bach in 1758 or *Table pour composer des minuets et des Trios a linfinie; avec deux dez a jouer* (A table for composing minuets and trios to infinity, by playing with two dice), created around 1780 by Maximilian Stadler.

From 1793 on, also musical dice games under the names of Haydn and Mozart appear.

1.4 The Computer in Algorithmic Composition

The first completely computer-generated composition on a symbolic level was produced by Lejaren Hiller and Leonard Isaacson from 1955 to 1956 with the *Illiac Suite* on the ILLIAC computer at the University of Illinois

[113, 114]. The symbolic level refers to the fact that the output of the system represents note values that must be interpreted by the musician. The *Illiac Suite* marked the beginning of computer-assisted algorithmic composition whose advancement was influenced by other compositional tendencies as well as developments in other fields of art.

With the development of higher programming languages, the first computer music systems for Algorithmic Composition were also generated. Their concepts are based either on general programming languages or have been designed especially for the particular application. In 1963, Hiller and Robert Baker developed *Musicomp*, the first computer-assisted composition environment.

The system *Groove* proposed by Max Mathews and Richard F. Moore marked the beginning of the developments in the 1970s. *Midi Lisp*, *Patch Work* and *Bol Processor* were the new programming environments of the 1980s. *Common Music*, *Symbolic Composer*, *Open Music* and many other systems followed at the beginning of the 1990s. A number of these systems are still in use today and are constantly being advanced.

Beginning with the languages of the *MusicN* family of Mathews from the 1960s on, Barry Vercoe's *Csound* (from 1986 on), Bill Schottstaedt's *Common Lisp Music* (from 1991 on) up to current developments such as *PureData* (PD) and *SuperCollider*, there is a wide range of powerful and flexible computer music systems.

In the following sections we present specific classes of algorithms in the context of Algorithmic Composition, first providing a general introduction to their development and their theoretical basics, and then describing different musical applications. Each of these class of algorithms can be seen as a paradigm of Algorithmic Composition because it enable a specific approach to musical structure generation. Although there is no universal approach to place the different categories in the algorithms used for automatic composition. Some of the main methods used are based on: *generative grammars*, *evolutionay methods* and *neural networks*.

1.5 Methods based on generative grammars

Generative grammars are powerful methods for Algorithmic Composition and musical analysis. The basic linguistic model [80] developed by Noam Chomsky in 1957, is the initial point for the application of this and other, more extended generative principles in musical tasks.

Fields that often use generative grammars are traditional European art music, jazz, as well as music ethnology. Related formalisms, such as augmented transition networks, are used for example in David Cope's approaches [84] for automatically generating compositions conforming with a given musical style.

The rewriting formalism of a generative grammar finds its parallels in different types of automata, Lindenmayer systems and Markov models. A specialized form of generative grammar in the field of linguistics is the categorical grammar or C-Grammar, a predecessor of the PS-Grammar, created by the Polish logicians Stanislaw Lesniewski in 1929 and Kasimierz Adjukiewicz in 1935. In algorithmic composition this type of grammar is not very common, but it is used, for example, by Mark Steedman [199] as a tool for analyzing jazz chord sequences.

1.5.1 The Chomsky Hierarchy

Chomsky distinguishes between four types of generative grammars that show different levels of restriction. These four types generate formal languages and correspond to different types of automata which may check whether a certain symbol string is part of the respective formal language and can thus be produced by the particular grammar.

The type of grammar is related to the level of its generative capacity. This means that a grammars generative capacity is high when it is able to generate several expressions and to prevent incorrect productions at the same time. However, a grammars generative capacity is low if the rules only allow limited control over the expressions to be generated. Consequently, a grammar of lower order has a higher generative capacity, since in this case there are fewer limitations regarding the formulation of production rules.

Type-0 Grammar (unrestricted grammar)

- Restrictions: No restrictions
- Respective formal language: *Recursively enumerable language* or *partially decidable language*
- Respective automaton: *Non-deterministic Turing machine*. In a non-deterministic Turing machine, the same inputs can produce different possibilities for resulting state transitions
- Generative capacity: Very high
- Complexity: Undecidable (up to infinite)

Type-1 Grammar (context-sensitive grammar)

- Restrictions: On both sides of the production rules, an arbitrary number of sequences of terminal or non-terminal symbols is possible, but the number of symbols on the right-hand side must not be smaller than the number on the left-hand side
- Formal language: *Context-sensitive language*

- Respective automaton: *Linear-bounded automaton*
- Generative capacity: High
- Complexity: exponential

Type-2 grammar (context-free grammar)

- Restrictions: The left-hand side of the production rules consists of one single nonterminal variable, the right-hand side of an arbitrary number of terminal or nonterminal symbols
- Formal language: *Context-free language*
- Respective automaton: *Pushdown automaton*
- Generative capacity: Middle
- Complexity: Polynomial

Type-3 grammar (regular grammar)

- Restrictions: The left-hand side of the production rules consists of only one variable (non-terminal of V); on the right-hand side there is a terminal, followed by one nonterminal at most. This form of production rules is also referred to as *right-linear*. If there is a terminal on the right-hand side that is preceded by a non-terminal, these production rules are also called *left-linear*
- Formal language: *Regular language*
- Respective automaton: *Deterministic finite automaton (DFA)* or *non-deterministic finite automaton (NFA)*
- Generative capacity: Low
- Complexity: Linear

1.5.2 Generative grammars as a model of the Theory of Syntax

The Theory of Syntax is an area of linguistics that deals with the formal structure of sentences composed. In particular, it is proposed to represent the *principles* and *structures* related to the formation of possible sentences in a language and will determine whether a given expression is consistent with the rules of the language or not, or whether it is syntactically correct.

The generation of new sequences in a generative grammar is by means of *production rules*, through which the symbols of an expression are rewritten

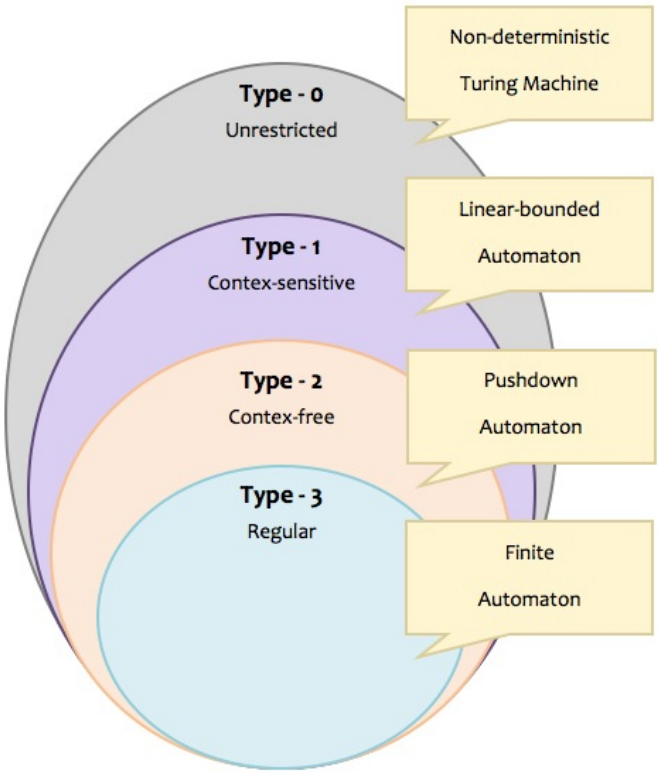


Figure 1.2: Chomsky Hierarchy

with additional symbols. We have to distinguish between symbols which can be rewritten further (*non-terminal symbols*) and symbols for which it is not possible (*terminal symbols*).

The process starts from a *start symbol* (nonterminal), continues with the rewriting of all non-terminal symbols and ends when the string consists entirely of terminal symbols.

A grammar can be denoted by a quadruple ($\mathcal{V}$, $\mathcal{V}_t$, $\mathcal{S}$, $\mathcal{P}$):

- $\mathcal{V}$ finite set of nonterminals
- $\mathcal{V}_t$ finite set of terminal symbols
- $\mathcal{S}$ initial non-terminal symbol
- $\mathcal{P}$ a set of production rules of the form: $\alpha \to \beta$, dove $\alpha \in \mathcal{V}^+$ e $\beta \in \mathcal{V}^*$

1.5.3 Generative Grammars in Algorithmic Composition

Music ethnology aims at describing different genuine music styles by grammatical models; in European art music, hierarchical grammatical structures are often used in analysis and generation; in jazz, this formalism is frequently used to create chord progressions on the basis of musical corpora

and rules of jazz harmony.

The basic idea of the algorithmic composition based on generative grammars is that concepts that allow a hierarchical division of the musical material and work with the substitution of symbols can naturally be best formulated by a generative grammar. If explicitly formulated rules are assumed, a knowledge-based approach is involved that is also applied in expert systems and comparable procedures. If a system automatically generates rewriting rules out of a corpus, it is a non-knowledge-based approach and can also be referred to as grammatical inference. Here, a comparison to further procedures may be drawn. For example, the expressiveness of Markov models equals to type-3 grammars. Due to the fact that they only allow for the treatment of a context of successive symbols, they are inferior to grammars of lower order. Another disadvantage of Markov models results from their fixed order, which is set before the model is generated and, in most cases, is not able to describe the context sufficiently. Artificial neural networks or genetic algorithms may also be used for the treatment of tasks where no domain-specific knowledge exists.

Systems that can independently find regularities in given data can also be found in the field of unsupervised learning of AI. These and similar algorithms can represent useful tools for the analysis and modeling of musical styles, on which there is insufficient domain-specific knowledge, but which may, however, be performed due to some kind of implicit rule system. Here, the generation of a terminal alphabet is necessary for the analysis of a corpus and the further generation of new musical material. In this case, terminals can be for example chords, harmonic movements, melodic fragments, rhythmic figures or also playing techniques of a particular instrument.

Musical Analysis by Generative Models

One important predecessor of generative grammar for application in music is Heinrich Schenkers musical analysis methods [193]. According to Schenker, the components of a tonal structure may be referred to as an imaginary fundamental pattern he calls the *Ursatz*, whose further structuring creates the different levels of a composition.

Inspired by Heinrich Schenkers ideas, in *A Generative Theory of Tonal Music*, Fred Lerdahl and Ray Jackendoff described an exhaustive model for the representation of tonal music by a generative formalism. An examination of musical representation by generative grammars including extended formalisms is also provided by Curtis Roads [190, 189].

Bol Processor

An interesting result in the application of generative grammars in the field of *music ethnology* is reached by Bernard Bel and Jim Kippen in

the study of North Indian instrumental music (*music Tabla*). For musical analysis and also for improvisation, designed the Bel system *Bol Processor (BP)*[70].

Tabla music does not have a written notation, but is represented in a system of notation mouth. However, the rhythmic phrases are denoted using onomatopoeic syllables, such *bol*[3] (for example *dha*, *thi*, *trkt*, ecc.) , each of which refers to a stroke or a phrase on the instrument. Bel and Kippen's research is based on a particular style of music, tabla, said *qa'ida*, which provides a formal model that works with themes and variations. Here are the first ten lines of changes in an *qa'ida*.

dha tr kt dha	tr kt dha ge	dha ti dha ge	dhee na ge na
dha tr kt dha	tr kt dha dha	dha ti dha ge	dhee na ge na
dha ti dha tr	kt dha tr kt	dha ti dha ge	dhee na ge na
dha ti kt dha	ti-dha ti	dha ti dha ge	dhee na ge na
dha ti kt dha	ti dha tr kt	dha ti dha ge	dhee na ge na
ti-dha ti	dha dha tr kt	dha ti dha ge	dhee na ge na
ti dha tr kt	dha dha tr kt	dha ti dha ge	dhee na ge na
tr kt dha ti	dha dha tr kt	dha ti dha ge	dhee na ge na
tr kt tr kt	dha dha tr kt	dha ti dha ge	dhee na ge na
tr kt dha tr	dha dha tr kt	dha ti dha ge	dhee na ge na

A variation consists of sixteen bol of the same length and duration of each variation is between eight and twelve seconds. Bol of the valid sequences can be represented by the following finite-state automaton:

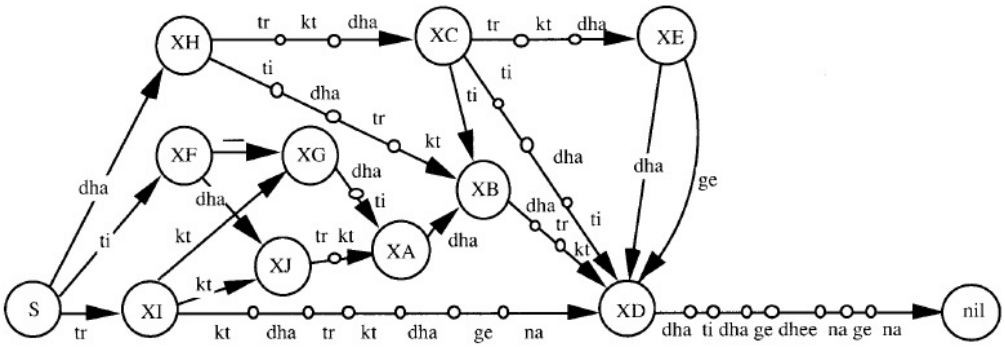


Figure 1.3: Finite automaton for representing sequences of bol

The processor BOL is currently a powerful tool for working with generative grammars in the field of algorithmic composition. In the selection of production rules to be applied, are regarded as likely to determine the order

[3]from Urdu/Hindi *bolna*, which means to speak

and preferences of these rules.

A particular aspect of BOL is the representation of temporal musical. In the representation of time, fragments of musical structures are put in relation by means of two key players that determine whether the fragments must be played sequentially or simultaneously.

Below is a part of generative grammar used by the system:

S	→	TA3	SA13
SA13	→	TA3	SA10
SA10	→	TC2	SA8
SA8	→	TD2	SA6
SA6	→	TA6	
SA10	→	TA2	SA8
. . .			
TB3	→	dhagena	
TF2	→	tidha	
TE2	→	ti-	
TC5	→	dhati-dhati	
TA6	→	dhagedheenagena	
TD2	→	dhati	
TC2	→	dhage	
TA3	→	dhatrkt	
TB2	→	trkt	
TA2	→	dhadha	

1.6 Methods based on evolutionary algorithms

Evolutionary algorithms (EAs) are population-based metaheuristic optimization algorithms that use biology-inspired mechanisms like mutation, crossover, natural selection, and survival of the fittest in order to refine a set of solution candidates iteratively [62, 68, 69].

The advantage of evolutionary algorithms compared to other optimization methods is their *black box* character that makes only few assumptions about the underlying objective functions. Furthermore, the definition of objective functions usually requires lesser insight to the structure of the problem space than the manual construction of an admissible heuristic. EAs therefore perform consistently well in many different problem categories.

1.6.1 The Basic Principles from Nature

In 1859, Darwin [89] published his book *On the Origin of Species* in which he identified the principles of natural selection and survival of the fittest as driving forces behind the biological evolution. His theory can be condensed into ten observations and deductions [89, 161]:

- The individuals of a species posses great fertility and produce more offspring than can grow into adulthood
- Under the absence of external influences (like natural disasters, human beings, etc.), the population size of a species roughly remains constant
- Again, if no external influences occur, the food resources are limited but stable over time
- Since the individuals compete for these limited resources, a struggle for survival ensues
- Especially in sexual reproducing species, no two individuals are equal
- Some of the variations between the individuals will affect their fitness and hence, their ability to survive
- A good fraction of these variations are inheritable
- Individuals less fit are less likely to reproduce, whereas the fittest individuals will survive and produce offspring more probably
- Individuals that survive and reproduce will likely pass on their traits to their offspring
- A species will slowly change and adapt more and more to a given environment during this process which may finally even result in new species

Evolutionary algorithms abstract from this biological process and also introduce a change in semantics by being *goal-driven*. The search space G in evolutionary algorithms is then an abstraction of the set of all possible DNA strings in nature and its elements $g \in G$ play the role of the natural genotypes. Therefore, we also often refer to G as the *genome* and to the elements $g \in G$ as *genotypes*. Like any creature is an instance of its genotype formed by embryogenesis3, the solution candidates (or *phenotypes*) $x \in X$ in the problem space X are instances of genotypes formed by the genotype-phenotype mapping: $x = gpm(g)$. Their fitness is rated according to objective functions which are subject to optimization and drive the evolution into specific directions.

1.6.2 The Basic Cycle of Evolutionary Algorithms

We can distinguish between single-objective and multi-objective evolutionary algorithms, where the latter means that we try to optimize multiple, possible conflicting criteria. Our following elaborations will be based on these MOEAs. The general area of Evolutionary Computation that deals with

multi-objective optimization is called EMOO, evolutionary multi-objective optimization.

A multi-objective evolutionary algorithm (MOEA) is able to perform an optimization of multiple criteria on the basis of artificial evolution. All evolutionary algorithms proceed in principle according to the scheme illustrated in Figure 1.4:

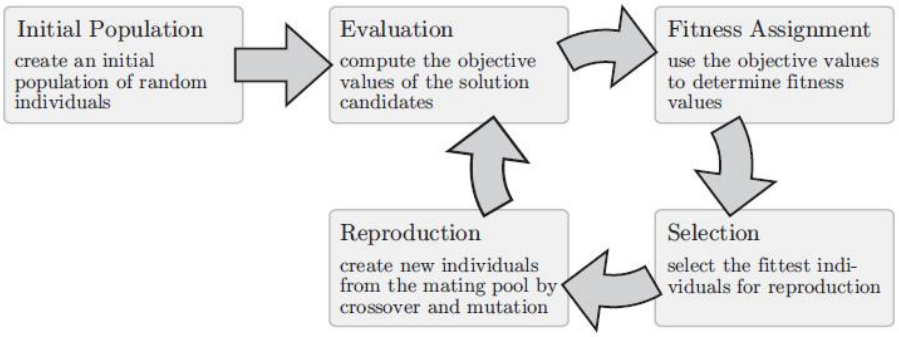


Figure 1.4: The basic cycle of evolutionary algorithms

1. Initially, a population Pop of individuals p with a random genome $p.g$ is created

2. The values of the objective functions $f \in F$ are computed for each solution candidate $p.x$ in Pop. This evaluation may incorporate complicated simulations and calculations

3. With the objective functions, the utility of the different features of the solution candi- dates have been determined and a fitness value $v(p.x)$ can now be assigned to each of them. This fitness assignment process can, for instance, incorporate a prevalence comparator function cmp_F which uses the objective values to create an order amongst the individuals.

4. A subsequent selection process filters out the solution candidates with bad fitness and allows those with good fitness to enter the mating pool with a higher probability. Since fitness is subject to minimization in the context of this book, the lower the $v(p.x)$-values are, the higher is the (relative) utility of the individual to whom they belong

5. In the reproduction phase, offspring is created by varying or combining the genotypes $p.g$ of the selected individuals $p \in Mate$ by applying the search operations searchOp $\in Op$ (which are called *reproduction operations* in the context of EAs). These offspring are then subsequently integrated into the population

6. If the terminationCriterion() is met, the evolution stops here. Otherwise, the algorithm continues at step 2

1.6.3 Classification of Evolutionary Algorithms

The family of evolutionary algorithms encompasses five members, as illustrated in Figure 1.5.

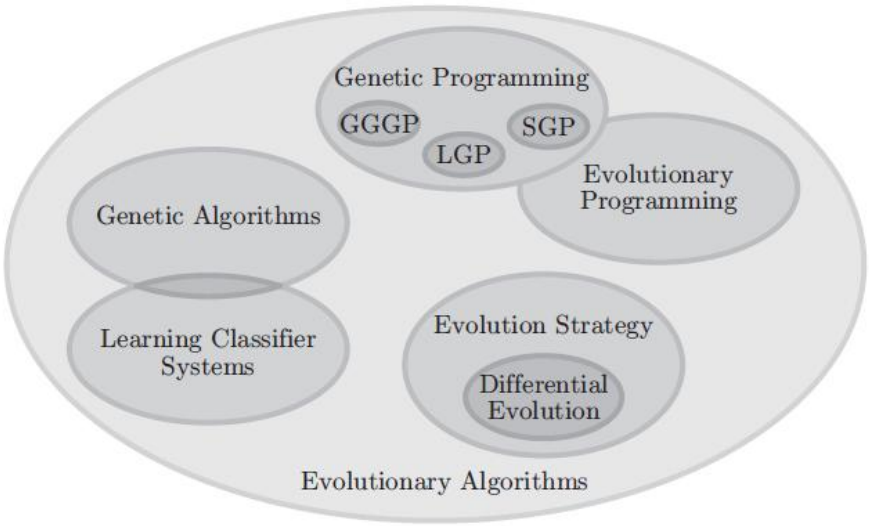


Figure 1.5: The family of evolutionary algorithms

We will only enumerate them here in short.

- **Genetic algorithms** (GAs). GAs subsume all evolutionary algorithms which have bit strings as search space G
- The set of evolutionary algorithms which explore the space of real vectors $X \subset \Re^n$ is called **Evolution Strategies**
- For **Genetic Programming** (GP), we can provide two definitions: On one hand, GP includes all evolutionary algorithms that grow programs, algorithms, and these alike. On the other hand, also all EAs that evolve tree-shaped individuals are instances of Genetic Programming
- **Learning Classifier Systems** (LCS), are online learning approaches that assign output values to given input values. They internally use a genetic algorithm to find new rules for this mapping
- **Evolutionary programming** (EP) is an evolutionary approach that treats the instances of the genome as different species rather than as individuals. Over the decades, it has more or less merged into Genetic Programming and the other evolutionary algorithms.

In the next section we will focus on a particular evolutionary method: *Genetic Algorithms*.

1.6.4 Genetic Algorithms

Genetic algorithms are a particular class of evolutionary algorithms. The terminology of genetic algorithms, including selection, mutation, survival of the fittest, illustrates the principles of these algorithms as well as their conceptual proximity to biological selection processes.

For the application of a genetic algorithm, domain-specific knowledge of the problem to be solved is not necessary. Therefore, this class of algorithms is especially suitable for tasks that are difficult to model mathematically or for problem domains that do not have an explicit superior rule system.

By analogy to the biological model, the respective computer program serves as the habitat that provides particular conditions for surviving and heredity. In this artificial living space, populations of individuals, or chromosomes, are produced whose adaptation to an objective, referred to as *objective score*, is examined by means of a *fitness function*. The fitness function may represent a mathematical function, a comparison set, or a rule-based system that examines the ability of a chromosome to fulfill the objective score. In Algorithmic Compositions, human fitness-raters are also frequently used; this approach, however, is subject to some restrictions.

In principle, the scheme of a genetic algorithm is structured as follows:

1. Generate random starting population of n chromosomes
2. Calculate the fitness of each chromosome If good enough, dump the result, END Else:
3. The fittest chromosomes are transferred unmodified or undergo crossover or mutation
4. Select a number of fittest chromosomes as starting individuals
5. Create next generation and repeat from step 2

1.6.5 Genetic Algorithms in Algorithmic Composition

As one of the first applications of genetic processes in Algorithmic Composition, Andrew Horner and David Goldberg [127] described the generation of melodic material by means of *thematic bridging*. This technique modifies a start pattern using a number of functions and compares the results with another pattern as fitness function. For the generation of a composition, Horner and Goldberg used six such cycles in order to produce melodic material that is then further structured by a five-voice canonical layering. The functions that in this case undertake the tasks of crossover and mutation are,

for example, with an initial pattern of Gb, Bb, F, Ab, Db and a reference pattern of F, Ab, Eb, as follows:

1. Start pattern: Gb Bb F Ab Db
2. Deletion of the last element: Gb Bb F Ab
3. Random swapping of the elements: Bb F Ab Gb
4. Deletion of the last element: Bb F Ab
5. Modification of the first element: Eb F Ab
6. Random swapping of the elements: F Ab Eb

Output: Gb Bb F Ab Bb F Ab Gb Bb F Ab Eb F Ab F Ab Eb.

In a two-stage process, the fitness function compares the conformity of the tone pitches of the output with the pitches in the reference pattern, as well as the length of the output with a desired objective. Although Horner and Goldbergs procedural method is very simple in this example, it describes the possibility of arbitrarily adjusting the principles of crossover, mutation and fitness evaluation with regard to the structuring of musical material.

Analogies to the Process of Composition.

In their applications of genetic algorithms to the generation of musical structure, Bruce L. Jacob [132] and Andrew Gartland-Jones [108, 109] emphasized this procedures similarity to a traditional compositional process. Jacob outlines the objective of *Variations*, his algorithmic composition system: The system was designed to reproduce very closely the creative process that this author uses when composing music, and Gartland-Jones refers to the general function principle of a genetic algorithm as follows: A commonly used compositional process may be described as taking an existing musical idea and changing it in some way. Musicians in various styles and genres may follow the process differently, some through improvisation, others through pencil and paper, but what is most often practiced is taking an existing idea and mutating it to provide a new idea. In fact, mutation is closely related to notions of development, which lie at the heart of western musical concepts of form and structure. It may even be possible to see development as directed mutation. With the core elements of GAs being mutation (including crossover) and fitness assessment, there appears to be a strong correlation with at least some aspects of the human process of generating musical material.

According to Gartland-Jones, populations of phrases of two bars obtain their fitness after the application of structure-modifying operations by a simple comparison with phrases of a given corpus. This principle finds further application in the generation of obbligatos, interactive installations and a software system. Distinctive parameters for notes are pitch, duration and

velocity; the populations are generated considering both the key and mode of the reference patterns. Ten functions are selected to serve as genetic operations that, among other things, mirror, invert, rearrange and mutate the material. The comparison of the patterns is applied on every note: If a note corresponds to a note of the reference pattern, it obtains fitness 1. Consequently, the fitness of a pattern is the total of the fitness of all single notes, whereas the position of the notes in the pattern is not taken into account. For the application of this principle in an installation the space is divided into sixteen sectors that are assigned four cycles with different fitness. Sensors capture the spatial position of the visitors and display the corresponding patterns with speakers. Obbligatos are generated in another application of this principle in relation to a users input. Here, each of the last four bars of incoming MIDI data are subject to different genetic operations and made audible simultaneously. Implementations of this principle within a software program make use of virtual building blocks that interact musically. Each of these blocks has its own musical pattern, and is able to send this pattern to other blocks and also receive patterns from these. When a pattern is passed to another block, it serves as a reference pattern for the pattern of the receiving block. The latter generates a new musical pattern based on its own music and the music it has just been passed, which again may be sent.

In his software project Variations, Jacob used three program modules for the generation of musical structure by means of genetic algorithms: the *Composer*,*Ear* and *Arranger*. In the Composer, variations of existing motifs are produced with the help of different techniques. The Ear evaluates the correctness of the motifs by permitted interval combinations and produces musical phrases that are assembled by the Arranger into larger units, which will be finally evaluated by a user. At the beginning, the user initializes the Composer with a number of motifs to which the module adds further motifs through transposition or rhythmic changes, for example. The evaluation of the motifs in the Ear is performed automatically by applying a simple comparison of patterns in order to examine whether all newly generated motifs show the same intervallic relations as the entered motifs. Because a motif in this system may be composed of both single notes and chords, in the evaluation the horizontal as well as the vertical intervallic relations are equally important. A valid new motif is at least a subset of the intervallic relations of the entered motifs; similarly, the doubling of chord components is allowed, as is the transposition of valid new generations or their rhythmic changes.

The Ear arranges all valid motifs into phrases that are composed by the Arranger by means of combination to larger musical units which are finally rated by the user.

Limits of Genetic Algorithms.

Somnuk Phon-Amnuaisuk, Andrew Tuson and Geraint Wiggins used genetic algorithms to harmonize soprano voices. In this approach, genetic operators such as mutation of chord type and swapping voices are applied and the fitness functions are executed in compliance with basic rules of music theory. The authors work is interesting also due to the fact that they point out intrinsic weaknesses of a genetic algorithm regarding the performance of particular tasks. An optimal fitness for all categories cannot be achieved even within a very large number of productions; the generation of chord progressions turns out to be extremely difficult.

First, harmonic progressions are highly context-sensitive so that when changing a chord, the functional context of the entire environment must be recreated. In the context of a genetic algorithm this may mean that when improving a particular fitness dimension, the newly generated chromosome provides worse values for other fitness dimensions. Another problem is rooted in the fact that the genetic algorithm yields good results in small musical tasks without, however, being able to generate the musical part as a whole in a way that makes sense. Finally, consideration must be given to the fact that the search in the state space is heuristic and not complete, and this, as a result, is a reason for the impossibility of always reaching an optimal solution for the musical task.

Even if the rule-based system shows a clear advantage over the genetic algorithm in this study, the fact that the rule-based approach is also subject to strong restrictions must be considered. The system is determined by the implemented rules; the output is completely foreseeable so that in some cases the gain of any insight from musical resynthesis must also be questioned. On the other hand, the genetic algorithm is completely able to produce surprising and yet musically satisfying results. Most of the algorithmic fitness functions generalize musical information by means of knowledge-based or rule-based strategies in order to be able to represent a uniform set of criteria for the systems outputs. In the recognition of inventive solutions that are actually violations of musical convention these fitness functions reach their limits early. In addition to the abovementioned strategies, artificial neural networks may also be applied for fitness evaluations; some works on this subject were produced by Lee Spector and Adam Alpern, Brad Johanson and Al Biles. A neural network may produce surprising results, both as a producing and an evaluating entity. Regarding context-sensitive structure, however, neural networks are often subject to the same restrictions as genetic algorithms so that in these cases, rating by a user is recommended.

In order to reduce the enormously increasing search space in these cases, Paul Pigg suggested a two-staged model for the improvement of these restrictions. In his approach, the user structures movements such as Intro, Chorus, Solo, etc. and initializes them by indicating bar and key. Two

genetic algorithms then generate the fine structure: The first genetic algorithm produces a genetic pool of bars that possess all characteristics of the corresponding movement; by this, a separate population is generated for each movement. The chromosome is represented by two separate symbol strings, the first referring to the pitch class and the other to the position of the octave. In place of a pitch class, symbols for rests and holds may be introduced in the symbol string as well. Crossover and mutation find application as genetic operators, whereas crossover is carried out in the usual way in the pitch classes and octaves. Mutation processes consist of shortening, extending and changing notes as well as random octave mutation. Fitness is rated by means of the simple principle that pitch classes are compared with the scale degrees that are, in turn, determined by the key of the respective movement. The evaluation of every single octave is based on the mean value of the chromosome; each derivation reduces an optimal fitness. The second genetic algorithm generates further variations on the basis of these chromosomes. In contrast to the first algorithm, this one generates chromosomes of greater length and its fitness function additionally includes triads belonging to the scale in the evaluation. Even when, due to the simple comparison of patterns, the fitness functions in Piggs model have difficulties in recognizing a coherent melodic structure, the pre-structuring of the musical material is an interesting approach to an efficient reduction of a large search space.

Rhythmic Generators.

Genetic algorithms also find application in the production of rhythm. Damon Horowitz developed a system [119] in which rhythmic patterns of differently instrumented structure are generated by means of the preferences and ratings of a user. Each chromosome represents a bar whose rhythmic values may comprise pauses as well as note values between 1/16 triplets and 1/4 notes. A number of rhythmic parameters including density, repetition, stressed beats and many more are created by the user in different ways according to their importance for the generation of the populations. Each of these criteria is furthermore assigned an optimal value for the fitness evaluation that must be reached, as well as a weighting in relation to the other criteria. This weighting controls how dominantly the respective criterion is represented in the musical appearance of the chromosome. As an additional option, another genetic algorithm produces the mentioned optimal values and weightings, and by doing this allows the generation of rhythmic *families*, groups of structures that differ in the occurrence of rhythmic characteristics. This Meta-GA facilitates evaluation by a user because the selection of particular rhythmic families leads more quickly to a rhythmic structure that is considered satisfying. Another function enables an efficient generation of rhythmic patterns by further structuring chromosomes regarding some parameters. The rhythmic structure of each chromosome

is produced by different percussion instruments whose sound characteristics are distinguished by their respective parameter ranges from each other. Crossover and mutation are also only applied within similar instruments so that sudden changes of sound color may be avoided by genetic operations.

For the generation of his genetic algorithms, Burton used ART (short for Adaptive Resonance Theory) networks as fitness evaluators which are a type of network applied to the forming of categories within unordered data. For the fitness evaluation, the network is trained with a corpus of drum patterns of different styles; the outputs of the genetic algorithm are compared by means of the resulting categories (clusters). An interesting aspect of this comprehensive work is also shown by a number of variants of genetic operations that are examined regarding their suitability for producing the rhythmic patterns.

Interactive Systems.

In numerous works Al Biles described the functioning of his program GenJam [72], developed to improvise jazz music based on genetic algorithms. On the basis of given harmonic and rhythmic structures, GenJam generates melodies that are rated by a user. The starting point for the melodies to be produced is information about tempo, rhythmic articulation, parts to be repeated and chord progressions. The chromosomes are represented as binary strings within two populations indicating bars and phrases. A bar in the genetic population consists of the assigned fitness as well as other values that represent notes, ties and pauses. The chromosomes on the phase level dispose of pointers on each four bar units. The population size in GenJam comprises 48 phrases and 64 bars. The scale degrees for the single chromosomes are selected on the bar level with respect to a number of possible chord types, as can be seen in Figure 1.6.

Chord	Scale	Chord	Scale
CMaj7, C6, C	C D E G A B	C7#9	C Eb E G A Bb
C7, C9, C13	C D E G A Bb	C7b9	C Db E F G Bb
Cm7, Cm9, Cm11	C D Eb F G Bb	CmMaj7	C D Eb F G A B
Cm7b5	C Eb F Gb Ab Bb	Cm6	C D Eb F G A
Cdim	C D Eb F Gb G# A B	Cm7b9	C Db Eb F G A Bb
C#5	C D E F# G# A B	CMaj7#11	C D E F# G A B
C7#5	C D E F# G# A#	C7sus	C D E F G A Bb
C7#11	C D E F# G A Bb	CMaj7sus	C D E F G A B
C7alt	C Db D# E Gb G# Bb	Â	Â

Figure 1.6: Allowed tone pitches regarding chord type in GenJam.

The genetic operation carried out on both chromosome types is a one-

point crossover, one half of the symbol string resting unmodified, the other half being subject to a number of mutations. Holds are represented here by 15, pauses by 0. Four chromosomes of the bar level are chosen at random, the two with the highest fitness being selected for mutation; their production replaces the two other chromosomes. This population forms the basis for the structuring of the phrases that in turn may be submitted to several mutations. The operator *Invert* additionally changes pauses to holds and vice versa. *Genetic repair* randomly chooses a new order for the elements and also replaces the chromosome with the worst fitness by a randomly selected element. *Super phrase* generates a completely new phrase by using bars of greatest fitness selected from four groups that consist of three consecutive bars each. *Lick thinner* substitutes randomly selected bars that are chosen most frequently in order to avoid the generation of material that only differs slightly from an optimal solution. *Orphan phrase* works similar to *super phrase* with the difference that it selects those bars that occur least frequently. By means of these genetic mutations operating on the bar and phrase level, structurally similar phrases may be generated whose genetic variety is guaranteed by functions that work against the production of monotonous musical material. The output of the system is rated binary by the user during playback. The fitness obtained through this is assigned to each particular combination of phrases and bars and serves as a guideline for the further generation of populations. In an extension of the software, the originally random selection of note values is controlled by algorithms that initialize the genetic populations by a structure similar to a reference corpus. Another improvement is the integration of user improvisations that are used as a basis for the genetic mutations.

1.7 Methods based on artificial neural networks

Artificial Neural Networks (ANN) were defined to reproduce some activities of the human brain, such as, for example, perception of images, pattern recognition and language understanding. The human nervous system is made up of billions of *neurons*. A neuron consists of a cell body and several branched extensions, called *dendrites*, through which the neuron receives electrical signals from other neurons. Each neuron has an *axon* with filamentous extensions which are used to transmit electrical signals to other neurons. The points of connection between the filamentous extensions and the dendrites are called *synapses*. A neuron receives electrical signals from the dendrites; when the overall received signal is bigger than a given *threshold* the neuron becomes active and transmits eletrical signals to other neurons through the filamentous extensions (see Figure 1.7). The interconnection of neurons in the human brain is very complicated and depends on the experience of the person. Such interconnection is the result of the life-

long learning experience that the person has been subject to. In the human brain there is no centralized control, in the sense that different brain areas work together, in uencing each other and contributing to the achievement of a specific task. Finally, the brain is fault tolerant. This means that if a neuron or one of its connections are damaged, the brain continues to function, albeit with degraded performance. In particular, the capabilities of the human brain degrades gradually as more and more neurons are destroyed (graceful degradation). This normally happens when people get old. Artificial neural networks try to simulate the behavior of the human brain. The activation function performs a weighted sum $a = \sum_{i=0}^{n} x_i w_i$ of the input and applies a function f to the resulting value a to produce an output $y = f(a)$. Usually, for the algorithmic music composition problems were used layered feedforward artificial neural networks. In these networks the neurons are arranged in *layers*. There is an input layer which receives the input signals and an output layer which produces the output signals. In between these two layers there might be several hidden layers each of which receives signals form the previous one and sends signals to the next one. An artificial neural network, as a human brain, needs first to "learn", before being able to solve problems. One of the most used learning techniques for an artificial neural network is the so called *supervised learning*. With such a technique we have to provide examples of problems together with the corresponding solutions. Analyzing the relations between the provided input and output the learning algorithm can set the synapses weights which determine the network behavior.

1.7.1 The Architecture of Neural Networks

In this section we describe several models of ANN.

The Perceptron. The perceptron is a model of a feedforward network that is constructed in regard to visual pattern recognition based on the concept of an artificial eye. The perceptron of a simple form has no hidden layer and the information from the image is passed to the input layer by non-trainable connections and fixed weights and from the input layer to the output layer with trainable connections and adjustable weights. In the binary model of the perceptron, inputs and outputs may only assume binary values and the weights are represented in real numbers. A simple threshold function is applied here and the output layer may consist of one or more neurons. As a training algorithm, the perceptron uses the delta rule which is a form of supervised learning in which the weightings of the neurons are updated corresponding to the error in the output of the perceptron. An extended model of a perceptron has a number of trainable layers and is known as a multi-level perceptron or also, corresponding to the learning algorithm that is applied in this type, a back-propagation net.

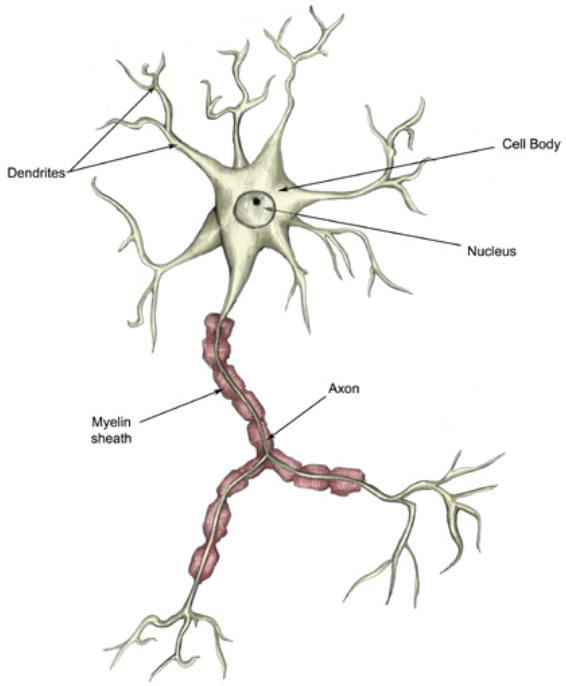


Figure 1.7: Model of a biological neuron

The Back-Propagation Network as an Extension of the Perceptron. A back-propagation network is a perceptron extended by additional layers and solves the class of non-linear separable functions. This type of ANN is trained with the backpropagation learning algorithm that also represents a method of supervised learning. The back-propagation algorithm changes the connection weights depending on the net error, which is the difference between the expected and the actual output of the ANN. The weights are changed beginning with the connections to the output layer and then continue backwards to the connections of the input layer.

Recurrent Neural Networks. When information from a previous step is important for the actual one step, recurrent neural networks can be used, which implement feedback in their design, like by connections leading from units in a layer to the same layer or to previous ones. This ANN is trained with the back-propagation algorithm. One of the first attempts to use such type of network for algorithmic music composition was HARMONET, realized by Hermann Hild, Johannes Feulner andWolfram Menzel [50]. Such a system harmonizes melodies in the style of J.S. Bach based on neural networks and a rule-based system. For the generation of a harmonic skeleton, a recurrent net with a hidden layer is used. This architecture is consequently

extended to three parallel nets. An interesting representation form in HARMONET indicates each tone pitch as the set of all harmonic functions that may contain this pitch as a chord element. In the generation of the harmonic skeleton, each quarter note is harmonized, taking into account its local context that functions as the input of the net. The current harmony Ht consists of the harmonic (Ht-1, Ht-2, Ht-3) and the melodic (st-1, st , st+1) context; phrt contains information on the position of Ht relative to the beginning or the end of a musical phrase; strt as a Boolean value indicates whether the current harmony is a stressed quarter. For the encoding of this context, 106 neurons are used. 70 of these neurons are used within a hidden layer and 20 form the output of the net. Having been generated by the ANN, the harmonic progressions are controlled by constraints that examine the material in terms of voice distances, parallels and the like. The net is trained on two sets of Bach chorales, each containing 20 chorales in major and minor respectively, using the back-propagation algorithm as training method. In a final step, ornamenting eights are added to the chord skeleton by a further net structure which also works in a contextbased manner.

Boltzmann Machines and LSTMs . Matthew Bellgrad and Lawrence Peh Tsang [11] use a Boltzmann machine (BM) for the harmonization of given chorale melodies composed in the Baroque period. A Boltzmann machine is one variant of a Hopfield net. As in a Hopfield net, the BM consists in general of a single layer of completely connected neurons and serves to assign patterns as well. In order to prevent the net from being limited to local energy minima, in the processing of a cycle the BM may also take on states with a higher energy potential. This process is a form of simulated annealing, so called for the conceptual analogy of this algorithm to the hardening of metals. In a BM, the net energy is temporarily increased by means of a controllable temperature parameter that in each step may additionally modify the activation of a neuron through a probability-based value. For their model Bellgrad and Tsang use an extended version of a so-called Effective Boltzmann Machine (EBM) that is trained on the local contexts of a set of chorales. All chorales of the training set are transposed on a mutual key. The tone pitches are represented locally by 35 neurons that present pitches of an underlying scale. Three further neurons indicate formal segments. According to its use by Bellgrad and Tsang, an event is the interconnection of a number of neurons to form a chord in the shortest appearing duration; longer durations are produced by repetitions. The harmonic context is learned by means of nested Boltzmann machines that each examines the immediate harmonic neighborhood of the note currently sounding. In case the observed context is not contained in the same configuration in the training set, the model may also generate harmonizations with a differing number of voices. An extension of the system that com-

prises restricting rules avoids undesired voice crossings and guarantees the correct number of voices for each harmonization step. Jurgen Schmidhuber and Douglas Eck [40] use a long short-term memory recurrent neural network (LSTM) for the production of melodic material over a given chord progression. A problem occurring in traditional network architectures lies in their treatment of context-sensitive material. Although common recurrent ANNs may consider a certain number of previously produced data, in a larger context, however, this type of network also shows some weaknesses, as Mozer states for his system: While the local contours made sense, the pieces were not musically coherent, lacking thematic structure and having minimal phrase structure and rhythmic organization. It appears that the CONCERT architecture and training procedure do not scale well as the length of the pieces grows and as the amount of higherorder structure increases. An LSTM consists of an input and output layer as well as a number of interconnected memory blocks. Each block contains a differing number of recurrent memory cells containing the information of previous states. Information is exchanged through gates that, according to the input and type of threshold function, may enable either the admission or transfer of information, pass on information after a delay or delete the content of the memory block. In most cases a conventional recurrent ANN may use ten to twelve previous steps for the context treatment, while an LSTM can treat a context of over 1000 timesteps. Eck and Schmidhuber use a local representation of note values and rhythmic changes are produced by holds. For both the chord and melody material the tonal space of an octave is applied. In a first experiment, only chord progressions are learned using a form of twelve bar blues. Further, parallel to the chord sequence, melody lines are built based on a pentatonic scale. An evaluation carried out by a jazz musician finds clearly better results for these generations than for passages produced with a random walk method. Although a considerably larger context may be treated by LSTMs in contrast to simple recurrent architectures, this model also reveals certain limitations in terms of the generation of larger musical sections. In the context of algorithmic composition neural networks are, in most cases, not exclusively used for the generation of structure, but can be found, however, often as an extension of other procedures (for example, in the form of fitness raters in the field of genetic algorithms) or for the examination of single musical aspects. Alejandro Pazos, A. Santos del Riego, Julian Dorado and J.J. Romero-Cardalda [107] developed a system treating agogics in Western classical music. The musical information used is provided by means of rhythmic pulsations of a MIDI pedal made by the musician. These data make up the training material of the ANN that is used for the prediction of agogic variation. Artificial neural networks may also be well applied in the field of classification where musical material is, among other things, examined in terms of the tonality of individual musical segments [99, 127, 128] or stylistic distinctions [28, 69]. For these analyses,

it is mainly Kohonens self-organizing maps that are applied.

1.8 Methods based on splicing systems

Splicing systems are a formal model of a generative mechanism of words (strings of characters), inspired by a recombinant behavior of DNA. They are defined by a finite alphabet $\mathcal{A}$, an initial set $\mathcal{I}$ of words and a set $\mathcal{R}$ of rules. Many of the studies about splicing systems focused on the properties of the generated language and their theoretical computational power.

Splicing is a language-theoretic word operation, introduced by Head [], which models a DNA recombination process, namely the action of two compatible restriction enzymes and a ligase enzyme on two DNA strands. Abstracting from the physical phenomenon, splicing is formalized as an operation on two words that generates new words. It concatenates a prefix of one string with a suffix of another string, under some conditions, represented as a (splicing) rule. A splicing system consists of a set $\mathcal{I}$ of words, the initial language, defined over an alphabet $\mathcal{A}$, and a set $\mathcal{R}$ of splicing rules. The language generated by a splicing system contains every word that can be obtained by repeated applications of rules to pair of words in the initial language and in the set of generated words. Splicing systems are a relatively old research topic in computer science and much early research effort has been devoted to the study of the computational power of such formal systems. The computational power mainly depends on the level of the Chomsky hierarchy that $\mathcal{I}$ and $\mathcal{R}$ belong to. For instance, the class

Chapter 2

Music Splicing Systems

2.1 Introduction

Modern computers are powerful means for creating, enjoying, and sharing art, music, film, and much more. Beside mere computations, computers are able to produce "creative" results, especially when the creation process is driven by clever algorithms. In the field of music, theory and application of new and existing technologies have been successfully employed in many aspects, such as sound synthesis, digital signal processing, sound design, and so on. A specific music problem which has been tackled with the use of computers is that of *Algorithmic Music Composition*. We can formalize such problem as the definition of a formal process whose goal is that of producing music with no (or minimal) human intervention.

In this chapter we describe the use of biological environment features, specifically those that characterize *splicing systems*, to develop a system for algorithmic composition([23],[24]).

2.2 Background

In this section we briefly recall the needed background to understand the rest of this chapter.

2.2.1 Music background

We consider the tempered music system used in western countries. Such a system, starting from a reference sound (a given frequency, normally 440Hz, that defines a reference note) defines octaves to be those sounds with frequencies which can be obtained by repeatedly doubling or halving the reference frequency. Each octave is divided into 12 equally spaced[1] notes, denoted by the letters A, A# or Bb, B, C, C# or Db, D, D# or Eb, E, F, F# or Gb,

[1] To be precise if a note has frequency f the "next" note has frequency $f \cdot 2^{\frac{1}{12}}$.

G and G# or Ab. This system has been developed mainly to easy transposition. Indeed tempered music is based on the notion of tonality. Roughly speaking, a tonality is a group of notes which form a scale. Starting from each of the 12 notes in an octave one can have a tonality (there are various kinds for each note, like major, minor, etc.). For example, the major scale of C is C, D, E, F, G, A, B, while the major scale of D is D, E, F#, G, A, B, C#. The notes of a scale are often denoted also by I, II, III, IV, V, VI, VII especially when one wants to emphasize only the degree of the scale and not the particular note, which depends on the tonality. A piece written in a given tonality can be easily transposed into another tonality by simply shifting all the notes; this is possible because all the notes are equally spaced. A good explanation of the tempered music system can be found in Chapter 3 of [157].

Usually one tonality is considered the main tonality of the piece, and therefore notes of the corresponding scale are considered more important than notes outside the scale. Western music, starting from the common practice period, is based upon harmonic and melodic rules for the tempered system, which are very well-established. We refer the reader to any good book on harmony, like [185], for a discussion about such rules. Here we will only recall some important concepts relevant to our work.

We focus our attention on music composed in 4 voices; unreachable examples of such type of compositions are J.S. Bach's chorales. A chorale consists of 4 independent voices, called *bass, tenor, alto* and *soprano*. A piece of music consists of a sequence of *measures*, each consisting of a given number of *beats*. In each beat the 4 voices play (or sing) a note[2]. The vertical set of notes in a beat is a *chord*. The notion of chord is fundamental. There are numerous kinds of chords. In this work we consider only 3- and 4-note chords. Chords are built upon each degree of the scale, that is I, II, III, IV, V, VI, VII. Often the note on which the chord is built, called the *root note* of the chord, is given to the bass; however the bass can also play any other note (chord inversions). Chord inversions are usually denoted with a superscript that indicates the inversion (for example, III^6, V^{46}, I^{357}). The chords we consider are described in Table 2.1 (for each we give one example in the tonality of C) .

One of the fundamental rules of the tempered music system is that a vertical set of notes must be one of the allowed chords. Other rules concern sequences of chords. Some sequences are "better" than others, where better is hard to define since it is a subjective evaluation. Anyway it is largely accepted that particular sequences of chords work better than others. Some chords are "more important" than others because they suggest, prepare, enforce or device tonal centers. The art of tonal music consists precisely in

[2]This a simplification of what really happens since a composition may have also passing notes which are not part of the harmony.

Chord	An example root note	set of notes
Major triad	C	C, E, G
Minor triad	E	E, G, B
Major seventh	C	C, E, G, B
Minor seventh	D	D, F, A, C
Dominant seventh	G	G, B, D, F
Half-diminished seventh	B	B, D, F, A

Table 2.1: Chords (with examples in the tonality of C)

arranging chords in such a way that their interplay is pleasant and meaningful. On the practical side, this translates into simple rules which state things like (the following is taken from [185]):

"I is followed by IV or V, sometimes VI less often II or III."

"II is followed by V, sometimes VI, less often I, III or IV."

Beside rules about chord sequences, we also have rules about melodic lines. A melodic line is simply the sequence of notes played by each single voice. Rules about melodic lines can refer to the movement of a single voice (for example, normally a jump bigger of an octave is not allowed; jumps within the notes of the scale are preferred to jumps to notes not part of the scale) or also to the movements of two voices (for example, two voices that proceed by parallel fifth are not allowed).

We refer the interested reader to a standard textbook on harmony, like [185], for a better and more detailed explanation.

2.2.2 Splicing Systems

Splicing is a language-theoretic word operation, introduced by Head [4], which models a DNA recombination process, namely the action of two compatible restriction enzymes and a ligase enzyme on two DNA strands. Abstracting from the physical phenomenon, splicing is formalized as an operation on two words (string) that generates new words. It concatenates a prefix of one string with a suffix of another string, under some conditions, represented as a (splicing) rule. A splicing system consists of a set $\mathcal{I}$ of words, the initial language, defined over an alphabet $\mathcal{A}$, and a set $\mathcal{R}$ of splicing rules. The language generated by a splicing system contains every word that can be obtained by repeated applications of rules to pair of words in the initial language and in the set of generated words. Splicing systems are a relatively old research topic in computer science and much early research effort has been devoted to the study of the computational power of such formal systems. The computational power mainly depends on the level of the Chomsky hierarchy that $\mathcal{I}$ and $\mathcal{R}$ belong to. For instance, the class of languages generated by splicing systems with a finite initial language and

a finite set of rules, often referred to as finite splicing systems, contains all finite languages and it is strictly contained in the class of regular languages. This result has been proved in several works by using different approaches (see [65, 64]). Splicing systems theory is still an interesting field of research, with complex open problems, as shown by more recent literature on this topic, such as [66].

In [4] Head has formalized the biochemical operation of splicing as an operation on strings. Following Head's work there have been further development of such an abstraction, in particular Păun [54] and Pixton [55] have introduced alternative operations. Each of these operations take as input two words and can generate either one new word, 1-splicing, or 2 new words, 2-splicing.

In this work we will use the 2-splicing operation introduced by Păun.

A Păun's 2-splicing rule r is explicited in the form $r = u_1\#u_2\$u_3\#u_4$, where u_1, u_2, u_3, u_4 are strings over a given alphabet $\mathcal{A}$ such that $\#, \$ \notin \mathcal{A}$. The words u_1u_2 (concatenation of u_1 and u_2), and u_3u_4 (concatenation of u_3 and u_4) are called *sites* of r. Roughly speaking, a site is a substring of the input words where we can "cut" the input words. Hence a rule identifies two points where we can cut the input strings.

Given such a rule r, and two words x and y belonging to the current language, if x contains the first site u_1u_2, that is $x = x_1u_1u_2x_2$, and y contains the second site u_3u_4, that is $y = y_1u_3u_4y_2$, rule r produces the strings $z = x_1u_1u_4y_2$, $w = y_1u_3u_2x_2$. We denote this operation by $(x, y) \vdash_r (z, w)$. Notice that if an input words contains a site multiple times, then the operation can be applied multiple times so that 2 new words are generated for each pair of 2 sites in the input words.

Let $r = a\#b\$c\#d$ and consider $x = eabf$ and $y = gcdh$. Then $eadh$ and $gcbf$ are generated by splicing. If we consider $x = eabfiabl$ and $y = gcdh$, then $eadh$, $gcbfiabl$, $aebfiadh$ and $gcbl$ are generated.

Splicing systems are models for generating languages based on the splicing operation. In order to generate a language using a splicing system we start from an initial set of words (often called the initial language) and we apply the rules of the splicing system to produce new words which are added to the initial set.

More formally a *splicing system* is a triple $\mathcal{S} = (\mathcal{A}, \mathcal{I}, \mathcal{R})$, where $\mathcal{A}$ is a finite alphabet such that $\#, \$ \notin \mathcal{A}$, $\mathcal{I} \subseteq \mathcal{A}^*$ is the initial language and $\mathcal{R} \subseteq \mathcal{A}^*\#\mathcal{A}^*\$\mathcal{A}^*\#\mathcal{A}^*$ is the set of rules. A splicing system $\mathcal{S}$ is finite if $\mathcal{I}$ and $\mathcal{R}$ are both finite sets. Let $L \subseteq \mathcal{A}^*$. We set $\sigma'(L) = \{w', w'' \in \mathcal{A}^* \mid (x, y) \vdash_r (w', w''),\ x, y \in L, r \in \mathcal{R}\}$. The splicing operation on languages

is defined as follows:

$$\begin{aligned} \sigma^0(L) &= L, \\ \sigma^{i+1}(L) &= \sigma^i(L) \cup \sigma'(\sigma^i(L)),\ i \geq 0, \\ \sigma^*(L) &= \bigcup_{i \geq 0} \sigma^i(L). \end{aligned}$$

Given a splicing system $\mathcal{S} = (\mathcal{A}, \mathcal{I}, \mathcal{R})$, the language generated by $\mathcal{S}$ is $L(\mathcal{S}) = \sigma^*(\mathcal{I})$. A language L is *$\mathcal{S}$-generated* (or is a *Păun splicing language*) if there exists a splicing system $\mathcal{S}$ such that $L = L(\mathcal{S})$.

2.3 Music Splicing System

In this section we describe two representations adopted to define the music splicing systems for the problem considered: *Note representation* and *Tonality-degree representation.*

The basic idea for both of them is the following. We start from a ground data set of Bach's chorales, representing them as words; such words are the initial set of the splicing system. Then we apply the splicing system to produce a language from the initial set. Such a language will contain many words; the actual number depends on the initial set and the set of rules of the splicing system. We remark that the process is deterministic. In order to generate the final language the splicing process has to be applied many times. This process might take a very long time, so for practical applications, such as the one we are proposing, we may stop the generation at some point and proceed with the language generated so far. Once we stop the splicing process we need to choose one single word from the generated language as the output of the system.

2.3.1 Note representation

We define a music splicing system that uses the Note representation as a *Note Music Splicing System* $\mathcal{S}_{\text{NMSS}}$. Such a system is defined by an initial ground data set of Bach's chorales, represented as words (initial set of words $\mathcal{I}_{\text{NMSS}}$ on the alphabet $\mathcal{A}_{\text{NMSS}}$), and a set of well-established rules in classical music composition (set of splicing rules $\mathcal{R}_{\text{NMSS}}$) obtained by extracting information about the notes. The language generated contains words that represent pieces of "new" chorales in the Bach style. Formally, a Note Music Splicing System is a triple $\mathcal{S}_{\text{NMSS}} = (\mathcal{A}_{\text{NMSS}}, \mathcal{I}_{\text{NMSS}}, \mathcal{R}_{\text{NMSS}})$.

The alphabet $\mathcal{A}_{\text{NMSS}}$

The alphabet we need has to allow us to specify the notes for each voice. Let $\mathcal{A}_V = \{\beta, \tau, \alpha, \sigma\}$ be the voice alphabet, where β stands for bass, τ

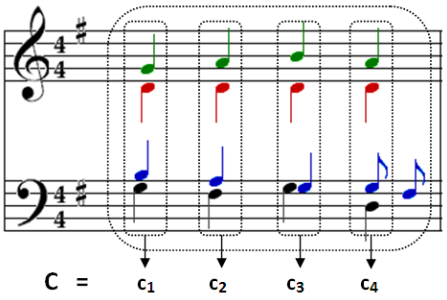


Figure 2.1: A fragment of BWV 32.6

for tenor, α for alto and σ for soprano. Let $\mathcal{A}_N = \{C, C\#, Db, D, D\#, Eb, E, F, F\#, Gb, G, G\#, Ab, A, A\#, Bb, B\}$ be the notes alphabet and $\mathcal{A}_O = \{3, 4, 5, 6\}$ be the octaves alphabet. Then let $\mathcal{A}_{\text{NMSS}} = \mathcal{A}_V \cup \mathcal{A}_N \cup \mathcal{A}_O$.

Using $\mathcal{A}_{\text{NMSS}}$ we can represent 4-voice chorale-like music as words. Notice that the "composition" that we will represent are sequences of chords. They might represent an entire chorale but also a fragment of it consisting of a few measures or even a single chord. A chord contains the notes sung (or played) in a specific beat by the bass, the tenor, the alto and the soprano voice (or instrument). Hence, a 4-voice composition is a sequence $C = (c_1, \ldots, c_n)$ where each c_i is a chord, for each $1 \leq i \leq n$.

We can represent each chord c_i as a word over $\mathcal{A}_{\text{NMSS}}$, more precisely, for each $1 \leq i \leq n$, c_i can be represented as a word $w_i = \beta x_i \tau y_i \alpha v_i \sigma z_i$, where $x_i, y_i, v_i, z_i \in \mathcal{A}_N \mathcal{A}_O$.

An entire composition $C = (c_1, \ldots, c_n)$ is represented as $w(C) = w_1 w_2 \ldots w_n$. Let $\mathcal{C} = C_1, \ldots, C_k$ be a set of 4-voice compositions (entire compositions, fragments or even single chords). The set of words associated to $\mathcal{C}$ is $\mathcal{W}(\mathcal{C}) = w(C_1), \ldots, w(C_k)$. We also say that $\mathcal{C}$ is the set of 4-voice compositions associated to $\mathcal{W}$.

Let us consider for example the 4-voice music fragment C in Figure 2.1 (a fragment of Chorale BWV 32.6), $C = (c_1, c_2, c_3, c_4)$. We have $w_1 = \beta G4 \tau B4 \alpha D5 \sigma G5$, $w_2 = \beta F\#4 \tau A4 \alpha D5 \sigma A5$, $w_3 = \beta G4 \tau G4 \alpha D5 \sigma B5$, $w_4 = \beta D4 \tau G4 \alpha D5 \sigma A5$, so
$w = \beta G4 \tau B4 \alpha D5 \sigma G5 \beta F\#4 \tau A4 \alpha D5 \sigma A5 \beta G4 \tau G4 \alpha D5 \sigma B5 \beta D4 \tau G4 \alpha D5 \sigma A5$.

The initial set of words $\mathcal{I}_{\text{NMSS}}$

To define $\mathcal{I}_{\text{NMSS}}$ a set, called the ground set $\mathcal{G}$, of 10 Bach's chorales[3] has been considered. Each one of these 10 chorales was transposed in every tonality. In addition other words associated to single chords have been inserted into $\mathcal{I}_{\text{NMSS}}$ to define the splicing rules.

[3]The 10 chorales in the ground set $\mathcal{G}$ are BWV 3.6, BWV 10.7, BWV 11.6, BWV 12.7, BWV 13.6, BWV 14.5, BWV 20.7, BWV 20.11, BWV 31.9 and BWV 32.6.

The objective is to define rules that generate good music patterns. So the chorales in $\mathcal{G}$ have been first analyzed, and then single chords have been extracted so that we can generate music that is closer to that in $\mathcal{G}$. During the chord extraction we attached to the extracted chord also information about the degree of the scale on which the chord is built. This information will be crucial in re-arranging the chords, by means of splicing rules, so that specific sequences of chords (e.g., cadences) will be produced. As done for the initial set, we have transposed each extracted chord in all 12 tonalities.

Rules definition

The definition of the rules is very important because the rules determine the language being generated. We would like to provide rules that produce good patterns of music. In order to do so we analyze and extract single chords from the chorales in the ground data set $\mathcal{G}$ so that we can (sort of) reproduce music that is similar to that in the ground data set.

In more details, we have extracted $13 * 12 = 156$ chords from the BWV 3.6, $18 * 12 = 216$ chords from the BWV 10.7, $9 * 12 = 108$ chords from the BWV 11.6, $17 * 12 = 204$ chords from the BWV 12.7, $19 * 12 = 588$ chords from the BWV 13.6, $7 * 12 = 228$ chords from the BWV 14.5, $5 * 12 = 60$ chords from the BWV 20.7, $16 * 12 = 192$ chords from the BWV 20.11, $16 * 12 = 192$ chords from the BWV 31.9, $18 * 12 = 216$ chords from the BWV 32.6, for a total of 2160 chords. We call $\mathit{Chords}(\mathcal{G})$ the set of these chords. The reason why we selected only some chords in each chorale and not every one was to keep the cardinality of $\mathit{Chords}(\mathcal{G})$ to a reasonable size.

For each extracted chord $c \in \mathit{Chords}(\mathcal{G})$, we store the information, provided by the harmonic analysis, about the degree on which the chord is built. We denote with $\mathit{Degree}(c)$ the degree of c. The set of words associated to $\mathit{Chords}(\mathcal{G})$ is $\mathcal{W}(\mathit{Chords}(\mathcal{G}))$.

We model the set of splicing rules on the basis of the classical harmonic rules (see Section 2.2.1). In particular we have decided to model as splicing rules the following musical cadences:

1. V $\rightarrow$ I
2. II $\rightarrow$ V
3. VI $\rightarrow$ II
4. V $\rightarrow$ VI
5. IV $\rightarrow$ V
6. IV $\rightarrow$ I
7. III $\rightarrow$ VI.

Moreover we also want that a composition starts and ends with a chord built on the first (I) degree of the scale, since this is what happens normally. We remark that for each of these situations (each cadence, and the starting and ending of the composition) we will define several rules of the splicing system.

Thus we can group all the rules in three sets of rules:

Rules for starting with I: For each pair of chords $c_1, c_4 \in \mathit{Chords}(\mathcal{G})$, such that $\mathit{Degree}(c_1) = I$, we define a rule $r = w_1\#\epsilon\$\epsilon\#w_4$ where ϵ is the empty word, w_1 is the word associated to c_1 and w_4 is the word associated to c_4. Moreover we also insert w_1 in $\mathcal{I}_{\text{NMSS}}$.

Rules for cadences: For each quadruple of chords $c_1, c_2, c_3, c_4 \in \mathit{Chords}(\mathcal{G})$, such that both $\mathit{Degree}(c_1) \to \mathit{Degree}(c_4)$ and $\mathit{Degree}(c_3) \to \mathit{Degree}(c_2)$ are cadences, we define the rule $r = w_1\#w_2\$w_3\#w_4$ where w_i is the word associated to c_i, for $i = 1, 2, 3, 4$.

Rules for ending with I: For each pair of chords $c_1, c_4 \in \mathit{Chords}(\mathcal{G})$, such that $\mathit{Degree}(c_4) = I$ we define a rule $r = w_1\#\epsilon\$\epsilon\#w_4$ where w_1 is the word associated to c_1 and w_4 is the word associated to c_4. Word w_4 is also inserted into $\mathcal{I}_{\text{NMSS}}$.

The above set of rules, applied to the ground set $\mathcal{G}$ yields a total of $1,658,880$ in $\mathcal{R}_{\text{NMSS}}$.

Time and space complexity

One of the most significant limitations of the Note representation is its efficiency in terms of both space and time complexity. Indeed an empirical evaluation has shown that the system based on that representation is quite slow and uses a lot of memory. At an analytical level, this can be justified by the fact that there is no information about the tonality and degree of the chords. This means that, in order to define the rules, we have first to transpose all the chords extracted in all the tonalities, and then we have to build the rules by considering all possible combinations. This calls for a lot of memory usage and consequently time consumption.

In Section 2.4.1, we report the time and memory consumption which, for very large initial sets and executions with many iterations, can become very large.

2.3.2 Tonality-degree Representation

In order to address the issues described in Section 2.3.1, in this section we introduce the *Tonality-degree representation*. We define a music splicing system that uses this representation as a *Tonality-degree Music Splicing*

System. The basic idea is to increase the musical information in the Note representation by also considering music degree information for each beat of the composition.

Similarly to what done in Section 2.3.1, we start from a ground data set of Bach's chorales, representing them as words and proceed as done Section 2.3.1, that is we apply the splicing system to the set of initial words repeating it many times; to produce the output we then choose one of the words.

As we have said in the previous section, a splicing system $\mathcal{S}$ consists of three components: an alphabet $\mathcal{A}$ of the symbols used, an initial set $\mathcal{I}$ of words and a set $\mathcal{R}$ of rules. In the following we will give these components for the tonality music splicing system $\mathcal{S}_{\text{TDMSS}} = (\mathcal{A}_{\text{TDMSS}}, \mathcal{I}_{\text{TDMSS}}, \mathcal{R}_{\text{TDMSS}})$.

The alphabet $\mathcal{A}_{\text{TDMSS}}$

For the Tonality-degree representation we use the voice alphabet $\mathcal{A}_V$, the notes alphabet $\mathcal{A}_N$ and the octaves alphabet $\mathcal{A}_O$, as defined in Section 2.3.1. Additionally, we introduce the tonality alphabet $\mathcal{A}_T = \{C,\ C\#,\ Db,\ D,\ D\#,\ Eb,\ E,\ F,\ F\#,\ Gb,\ G,\ G\#,\ Ab,\ A,\ A\#,\ Bb,\ B\}$, the quality alphabet $\mathcal{A}_Q = \{M, m\}$ (where M stands for major tonality and m for minor tonality), and the degree alphabet $\mathcal{A}_D = \{1, 2, 3, 4, 5, 6, 7\}$.

As explained in Section 2.3.1 an entire composition $C = (c_1, ..., c_n)$ is represented as $w(C) = w_1 w_2 \cdots w_n$. Given a set of 4-voice compositions $\mathcal{C} = \{C_1, \ldots, C_k\}$, the set of words associated to $\mathcal{C}$ is $\mathcal{W}(\mathcal{C}) = \{w(C_1), \ldots, w(C_k)\}$. We also say that $\mathcal{C}$ is the set of 4-voice compositions associated to $\mathcal{W}(\mathcal{C})$.

Let us consider the 4-voice music fragment C in Fig. 2.1 (a fragment of Chorale BWV 32.6), $C = (c_1, c_2, c_3, c_4)$. We have $w_1 = GM1\beta G4\tau B4\alpha D5\sigma G5GM1$, $w_2 = GM5\beta F\#4\tau A4\alpha D5\sigma A5GM5$, $w_3 = GM1\beta G4\tau G4\alpha D5\sigma B5GM1$, $w_4 = GM5\beta D4\tau G4\alpha D5\sigma A5GM5$, so $w = w_1 w_2 w_3 w_4$.

Initial set and rules definition

The initial set is defined as described in Section 2.3.1. As for the Note representation, we start again from the ground data set $\mathcal{G}$ obtained by analyzing a set of Chorales by Bach.

Unlike to what done for the previous representation, we don't attach the degree of the scale on which the chord is built during the chord extraction. Instead, we directly integrate the degree and the tonality of the chord in the word representation of each chord. So, we do not need to transpose each chord in all 12 tonalities, and we use only the real chords that appear in the the ground data set.

As for the Note representation, in this representation we model the set of splicing rules on the basis of classical harmonic rules.

However, for the Tonality-degree representation system, we don't build the rules by checking all the combinations, but we use directly the chords extracted. This guarantees a considerably smaller number of rules. Furthermore, in this representation we directly extract the rules for the modulations.

Thus we can group all the rules in four sets of rules:

Rules for starting with I: For each $c_1, c_4 \in \mathit{Chords}(\mathcal{G})$), such that $\mathit{Degree}(c_1) = I$ and $\mathit{Tonality}(c_1) = \mathit{Tonality}(c_4)$, we define a rule $r = w_1\#\epsilon\$\epsilon\#w_4$ where ϵ is the empty word, w_1 is the word associated to c_1 and w_4 is the word associated to c_4. Moreover we also insert w_1 in $\mathcal{I}_{\text{TDMSS}}$.

Rules for cadences: For each $c_1, c_2, c_3, c_4 \in \mathit{Chords}(\mathcal{G})$, such that both $\mathit{Degree}(c_1) \to \mathit{Degree}(c_4)$, with $\mathit{Tonality}(c_1) = \mathit{Tonality}(c_4)$, and $\mathit{Degree}(c_3) \to \mathit{Degree}(c_2)$, $\mathit{Tonality}(c_3) = \mathit{Tonality}(c_2)$, are cadences, we define the rule $r = w_1\#w_2\$w_3\#w_4$ where w_i is the word associated to c_i, for $i = 1, 2, 3, 4$.

Rules for ending with I: For each $c_1, c_4 \in \mathit{Chords}(\mathcal{G})$, such that $\mathit{Degree}(c_4) = I$ and $\mathit{Tonality}(c_1) = \mathit{Tonality}(c_4)$ we define a rule $r = w_1\#\epsilon\$\epsilon\#w_4$ where w_1 is the word associated to c_1 and w_4 is the word associated to c_4. Word w_4 is also inserted into $\mathcal{I}_{\text{TDMSS}}$.

Rules for modulations: For each $c_i, c_{j+1}, c_j, c_{i+1} \in \mathit{Chords}(\mathcal{G})$, such that $\mathit{Tonality}(c_i) \neq \mathit{Tonality}(c_{i+1})$, and c_i and c_{i+1} are consecutive chords, and $\mathit{Tonality}(c_j) \neq \mathit{Tonality}(c_{j+1})$, c_j and c_{j+1} are consecutive chords, we define the rule $r = w_1\#w_2\$w_3\#w_4$ where w_i is the word associated to c_i, for $i = 1, 2, 3, 4$.

Starting with $\mathcal{G}$ containing Bach's chorales BWV 3.6, BWV 10.7, BWV 11.6, BWV 12.7, BWV 13.6, BWV 14.5, BWV 20.7, BWV 20.11, BWV 31.9 and BWV 32.6 we have $|\mathcal{G}| = 10$, and so we have obtained 138 chords. The above set of rules, applied to the ground set $\mathcal{G}$ yields a total of $8,598$ in $\mathcal{R}_{\text{NMSS}}$.

Therefore, we can observe how both the number of chords and the number of rules are notably reduced compared with the Note representation (see Section 2.3.1).

2.4 Studies

We report in this Section the results of two studies aiming at analyze: *(1)* the performance of the Tonality-degree representation and *(2)* the easiness of the proposed representation from the end user point of view.

About the first study (Performance evaluation), we have compared the system with two other similar systems by measuring execution time, memory consumption, harmonic and melodic values.

About the second study (User evaluation) we have administered a evaluation test to some recruited people asking their opinion on various aspects. The aim of this study was to derive information about both the easiness of the representation, when compared with the same approaches used for the Performance study, and the user satisfaction, in terms of usefulness, when apply it to programming activities. For this study we used subjective metrics (i.e., analysis of the Java programs produced by programmers when coding some tasks), and objective metrics (i.e., standard quality metrics used to evaluate the quality of the software).

Section 2.4.1 reports the results of the first study while Section 2.4.2 reports the result of the second study.

2.4.1 Performance evaluation

We have implemented both the automatic composer based on the Tonality-degree representation and the one based on the Note representation. Moreover we also used the automatic composer EvoBassComposer [15], a multi-objective genetic algorithm that automatically composes music when provided with a bass line input. We have run tests for each of the three implementation and we have considered four metrics: execution time, memory consumption, harmonic and melodic values.

Before discussing the results of the first study, we provide some details about the implementation.

Implementation details. We now give details about the implementation of the automatic composer based on a music splicing system. Let $\mathcal{S}_{\text{NMSS}}$ (resp. $\mathcal{S}_{\text{TDMSS}}$) be the music splicing system considered, and $L = L(\mathcal{S}_{\text{NMSS}})$ (resp. $L = L(\mathcal{S}_{\text{TDMSS}})$) the language generated. We fix a number of iterations. A single iteration corresponds to an application of all the rules in $\mathcal{R}_{\text{NMSS}}$ (resp. $\mathcal{R}_{\text{TDMSS}}$); in each iterations the rules are applied to all possible pairs of words in the current language. Let $L_k(\mathcal{S}_{\text{NMSS}})$ (resp. $L_k(\mathcal{S}_{\text{TDMSS}})$) be the language obtained after k iterations. The value sof k that we have considered are $k \in \{10, 50, 100, 500, 750, 1000, 2500, 5000, 7500, 10000\}$.

Given a k and the language $L_k(\mathcal{S}_{\text{NMSS}})$ (resp. $L_k(\mathcal{S}_{\text{TDMSS}})$) we decide to choose one single word $w \in L_k(\mathcal{S}_{\text{NMSS}})$ ($w \in L_k(\mathcal{S}_{\text{TDMSS}})$) as the output of the algorithmic composer. The chosen output chorale is the composition represented by such a word. The selection of the output word is based on a function that measures the harmonic quality of the composition, proposed in [15]. The harmonic function is based on "weights" assigned to pairs of consecutive chords, chosen so that good harmonic pattern have heavier weights. In Table 2.2 we report the weights for stepping between chords in the same tonality.

Furthermore, we also use a function for the melodic quality whose definition is based on "weights" assigned to each type of melodic errors that

Major	Degree						
Degree	I	ii	iii	IV	V	vi	vii°
I	250	200	50	200	250	50	10
ii	100	100	100	150	2000	150	10
iii	100	100	100	200	100	250	10
IV	250	150	100	200	1500	100	10
V	2000	100	100	100	250	150	10
vi	100	200	150	150	200	200	10
vii°	1000	50	150	50	50	100	200

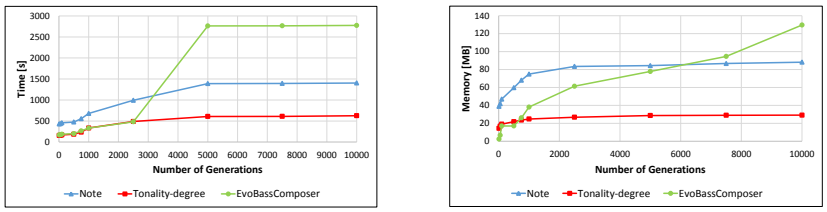
Minor	Degree						
Degree	i	ii°	III	iv	V	vi	vii°
i	250	250	50	200	250	50	10
ii°	100	100	100	150	250	150	10
III	100	100	100	200	100	250	10
iv	250	150	100	200	200	100	10
V	250	100	100	100	250	150	10
vi	100	200	100	100	200	200	10
vii°	250	50	150	50	10	10	200

Table 2.2: Table of the weights for pair of consecutive chords in the same tonality.

occur in the composition. We remark that also this measure is the one used in [15].

Performance results. Results of this study are summarized in Figs. 2.2 and 2.3. As shown in Fig. 2.2(a), the execution time of the implementation based on the Tonality-degree representation is the best one. With respect to Note representation, when considering a number of generations smaller than 2500, the behavior of EvoBassComposer and the Tonality-degree representation is almost the same. When increasing the number of generations, EvoBassComposer undergoes a deterioration in its behavior, mainly due to an increase of the size of the elite population, and therefore, to an increase of the number of individuals to which the genetic operators were applied.

The Note representation, instead, is always slower due to the generation of inconsistent individuals, given the absence of degrees information in the representation.



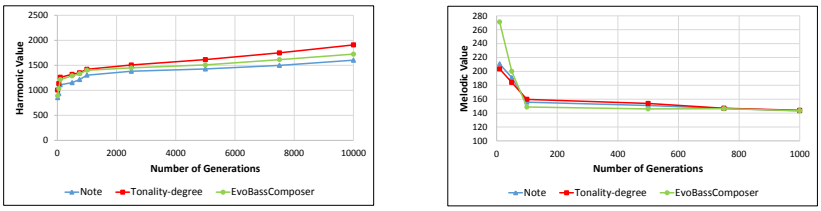
(a) Execution Time (b) Memory consumption

Figure 2.2: Performance results in terms of execution time and memory consumption

We observed a similar behavior for the memory consumption, as reported in Fig. 2.2(b)).

As shown in Fig. 2.3(a), the Tonality-degree representation generates a better solution in terms of the harmonic metric, while from the melodic point of view, the three approaches tend to generate solutions almost equivalent, as shown in Fig. 2.3(b).

In summary, as result of the Performance study, we can state that the Tonality-degree representation outperforms the other analyzed approaches by allowing to obtain high quality results in a shorter time and using less



(a) Harmonic value (b) Melodic value

Figure 2.3: Performance results in terms of harmonic and melodic metrics.

memory.

2.4.2 User evaluation

In this Section we first describe the methodology employed for the user study, then we discuss the results obtained during a code programming phase, performed by a group of people that, according to the Within-Subjects Design [38], tested the Tonality-degree representation comparing it with a different representation chosen by each member of the group.

The objective of this evaluation was to explore the advantages that the proposed representation is able to produce during the programming activities of programmers with background and skill in music and evolutionary methods. The metrics we evaluated include the quality of the produced code (both subjective and objective analysis) and the overall perception of the participants in terms of easiness and usefulness of the proposed representation.

Method. The User study was organized in three phases in which we carried out *(a)* a Preliminary Survey, *(b)* a Testing Phase, and *(c)* a Summary Survey [40, 39].

The aim of the first phase was to collect information about demographics, musical and programming background. In the Testing Phase, to solve specific problems, we asked users to write simple Java programs by using first the Tonality-degree representation, and then, a second representation decided by the users themselves. For each representation we asked users to perform three tasks: *(Task 1)* implementing a mutation operator on a musical composition, *(Task 2)* implementing a single-crossover operator and *(Task 3)* implementing a multiple-crossover operator.

To avoid biased results, tasks were submitted in a random order. At the end of the Testing Phase we asked users to rate how easy was completing the tasks (with questions from the ASQ questionnaire[4]), and which approach was the simplest in terms of data structures defined and implemented. In

[4] `http://garyperlman.com/quest/quest.cgi?form=ASQ`

the third phase we asked participants to express their overall satisfaction when using the Tonality-degree representation, in terms of metrics such as usefulness, ease of use, and attitude (with a subset of questions from the TAM questionnaire [45]). The overall study took 40 minutes for each user. Finally, the questionnaires submitted to the participants are available online[5], and their responses have been analyzed using SPSS version 20[6].

Results. Recruits comprised 10 participants, with 70% of them having a master level in Computer Science. The full sample was male with a mean age of 31. Fig. 2.4 shows that the full sample had music knowledge (no one of the participants was *Inexpert* in that field), 90% consider themselves more than *Competent* with regard to expertise in code programming, and 70% more than *Competent* with regard to expertise in evolutionary methods programming.

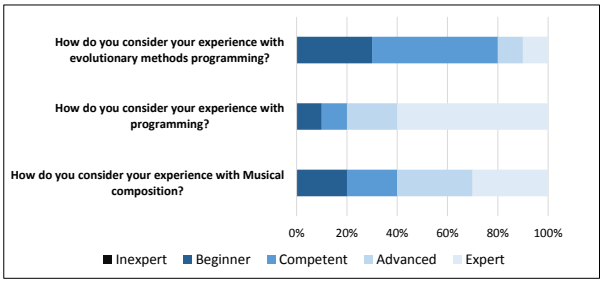


Figure 2.4: Preliminary Survey results.

As we can see in Table 2.3, the results of the Testing Phase showed that in terms of easiness of completing the tasks on average the users were more satisfied when using the Tonality-degree representation (mean across all tasks of 4.4, against 3.5 for the free representation, SD=0.1). Similarly, about the time it took to complete the tasks, users were more satisfied with the Tonality-degree representation.

We used statistical tests to verify if existed significant differences in terms of user satisfaction when completing the tasks, by using the Tonality-degree representation against the own approach. This test revealed that there is a significant difference with regards both the easiness of completing the tasks and the time it took to complete them (p-value $< .001$ for both questions).

To measure the quality of the produced code, as anticipated before, we used both subjective and objective metrics. Specifically, for the objective analysis we used a metric commonly used to evaluate the quality of software, that is, the source line of codes (Fourth and octave column in Table 2.3). The subjective analysis was intended to analyze the diligence (how carefully

[5] `http://www.di.unisa.it/~delmal/research/usability/Splicing`
[6] `http://www-01.ibm.com/software/analytics/spss/`

the participant does her work) of the participants when completing their tasks, that is, use of significative names for variables, code indentation, and so on. We assigned a score (from 1 to 10) to each of the these metrics, and the mean value across them is shown in the "Subjective Metrics" columns of Table 2.3.

As we can see from Table 2.3, the objective analysis showed that the Tonality-degree representation allowed participants to write a smaller number of source code lines (mean value of 98, against 195 for the representation freely chosen by users). We also found out a significant difference with regard to this metric (p-value $< .001$).

Table 2.3: Testing Phase results. Satisfaction measured with 5-point Likert scale questions. Results about Software metrics are in a range between 1 and 10.

	Tonality-degree representation				Freely chosen representation			
	Satisfaction: Easiness of the tasks	Satisfaction: Time to complete the tasks	Source line of codes	Subjective Metrics	Satisfaction easiness of the tasks	Satisfaction time to complete the tasks	Source line of codes	Subjective Metrics
Task 1	4.5	4.2	91	8.5	3.6	3.6	177	7.8
Task 2	4.3	4.2	102	8.5	3.5	3.3	209	7.5
Task 3	4.4	4.4	101	8.4	3.5	3.7	200	7.6
Mean	4.4	4.3	98	8.5	3.5	3.5	195	7.7
SD	0.1	0.1	6	0.1	0.1	0.2	17	0.1

Additionally, the subjective analysis showed that although the quality of the produced Java programs (Subjective Metrics) was high when using both representations, a slight positive result is obtained with our approach (mean value of 8.5 against 7.7 for the freely chosen representation).

From the analysis of the third phase, we discovered that, in general, in terms of easy of use, participants were more satisfied when interacting with the Tonality-degree representation (question Q1 in the Summary Questionnaire, mean = 4.1, SD = 0.9). Moreover, at the question "*In terms of used data structures and complexity of the code, which approach do you feel easier to use?*" (question Q2 in the Summary Questionnaire), on average, 87% of participants choose the Tonality-degree representation (the remaining 13% rated as "*Equivalent*" the tested approaches).

Finally, as shown in Table 2.4, all metrics about usefulness, ease of use and attitude toward using our representation, were rated highly positive. These results highlight how our representation was felt as easy to learn, easy to use, and useful to accomplish tasks in a more quick and effective way.

2.5 Conclusion

Various bio-inspired processes have been used to define algorithms for automatic music composition: evolutionary algorithms, bio-inspired algorithms, formal grammars, cellular automata, machine learning. In this work we pro-

Table 2.4: Measures and constructs. Rating on a 7-point Likert scale.

Metric		Mean	SD
Usefulness			
U1	Using the proposed representation would enable me to accomplish tasks more quickly	5.7	0.9
U2	Using the proposed representation would improve my performance	6.0	0.8
U3	Using the proposed representation would enhance my effectiveness	5.9	1.0
Easy of Use			
EOU1	Learning to operate the proposed representation would be easy for me	6.4	0.8
EOU2	I would find the proposed representation easy to use	5.9	1.1
Attitude Toward Using			
ATT1	Using the proposed representation is a (bad/good) idea	6.0	0.8
ATT2	Using the proposed representation is a (foolish/wise) idea	5.7	0.8

vided a new system for automatic music composition based on the splicing model, a generative mechanism of linear words, inspired by a recombinant behavior of linear DNA.

We have defined an augmented representation approach in which additional musical information is incorporated into the word representation of the chords. We have showed that the use of this representation improves musical splicing system in terms of time and memory performance, and in terms of harmonic and melodic quality. We also showed that our representation was evaluated as easy to learn, easy to use, and useful to accomplish tasks quickly and in a effective way.

Future work involves an investigation of other approaches based on the representation proposed in this work. It would be interesting to study the behavior and the efficacy of a music splicing system by changing the style composer, the ground data set that determines the output, and also the music information that could be integrated in the representation.

Chapter 3

EvoBackMusic

3.1 Introduction

Interest for real-time composition of *background music* has grown with the increasing diffusion of Intelligent Virtual Environments, movie music production and interactive media. A system for real-time composition of background music responds to changes of the environment by generating "*appropriate music*" that matches the actual state of the environment and/or of the user.

Background music has an accompanying role that enriches a particular activity or event such as an exhibition of paintings, a theater, the scenes of a film, a Web site, a video game or a virtual environment.

In general, the approach followed by developers, in the game or film industry for example, is to loop pre-composed music tracks which are attached to particular locations or events. One problem with this approach is that users get accustomed to listening the same music over and over again. Another problem is that users associate music to events, by predicting what will be going to happen next. Only recently some commercial sound engines started to include technology that allow to increase music diversity. Microsoft DirectMusic[1], for example, provides the possibility to compose several pieces of music which are randomly arranged at run time, but continues to do this in response to some specific events.

The problem of composing real-time background music has attracted considerable attention in the last few years. Most of the existing efforts produce music mainly considering the emotional state of the user, without involving objective aspects of the environment in which the user is immersed.

In this chapter, we describe a system [10] capable of: *(1)* learn the music preferences of a user, and *(2)* generate background music according to the current state of the environment and of the user.

From now on, when we talk about environment we implicitly include

[1] `http://www.microsoft.com/DirectX`

the user in the environment. In other words the environment includes both subjective aspects, the emotional state of the user, and objective aspects, the environment in which the user is immersed. Specifically the *subjective aspects* concern the emotional experiences of the user, such as happiness, sadness and so on, while the *objective aspects*, regard the dynamism of the elements in the environment, the density of the elements in the environment, the type of the activity being performed, and a predominant color of the environment.

In order to compose music we take into account several aspects: harmony, melody, rhythm, and timbre. Thus, we tackle the problem of composing background music with a multi-objective problem. Specifically, we consider: *(1)* the *instrumental objective*, i.e., to find an appropriate instrumental orchestration; *(2)* the *rhythmic objective*, i.e., to find an appropriate rhythmic section; *(3)* the *melodic objective*, i.e., to find good melodic lines; *(4)* the *harmonic objective*, i.e., to find good harmony. Moreover, the problem can be considered as a *dynamic optimization problem* in which the objective functions (i.e., the generated music) change continuously according to the changes of the environment.

From the technical point of view, EvoBackMusic consists of three agents: the Observer Agent, the Composer Agent, and the Sound Agent. The Observer Agent checks for environment state changes and provides such changes to an Artificial Neural Network (ANN). The ANN, trained according to the user preferences, maps the current environment state produced by the environment changes into a *musical state*. Then, the musical state is given to the Composer Agent that, using a multi-objective dynamic genetic algorithm, composes appropriate music. Finally, the Sound Agent converts music compositions into audio.

The three Agents have been implemented in Java. The implementation of the Artificial Neural Network uses the Encog library[2]. The Sound agent communicates with a sequencer implemented using the JFugue library[3].

To evaluate the overall system, we performed several tests that study its behavior and its efficacy. The quality of the produced music, instead, was analyzed by participants recruited for a preliminary evaluation study.

The problem of composing background music has been considered by several researchers. Downie [101] presents a system that produces Music Creatures following a reactive, behavior-based AI approach. Nakamura et al. [174] implemented a prototype system that generates background music and sound effects for short movies, based on Actors Emotions and Motions. A work done at the University of Edinburgh [198] describes a system that generates atmospheric music in real time to convey fear using suspense and surprise. The system's parameters are controlled by a human director.

[2] `http://www.heatonresearch.com/encog`

[3] `http://www.jfugue.org/`

Several systems are grounded on research made in the areas of Computer Science and Music Psychology. Systems that control the emotional impact of musical features modify emotionally-relevant structural and performative aspects of music [82, 154, 210] by using pre-composed musical scores [154, 155] or by making musical composition [145, 78, 206]. Most of these systems are built on empirical data obtained from works of psychology [107, 197]. Scherer and Zentner [194] established the parameters of influence for the experienced emotion. Meyer [164] analyzed structural characteristics of music and its relation with emotional meaning.

We remark that most of these works only take into account emotional aspects and they use pre-composed music for generating background music. In our approach we investigate the importance of both subjective and objective aspects of an environment for the composition of background music. Furthermore, our system always composes new music, in real-time, according to the musical preferences of the user.

3.2 Background

3.2.1 Feed-Forward Neural Networks for Regression problems

Neural networks are widely used to reproduce some activities of the human brain, for example, perception of images, pattern recognition and language understanding. They are a very useful tools also for gesture recognition.

Our problem can be seen as a *regression problem* in which the network, given an environment state, has to be able to predict an appropriate musical state according to the musical preferences of the user. The input to our problem is the environment state and the output is a musical state. The neural network has to be trained on a data set of pairs *(environment state, musical state)*. We used a fully connected three-layer feed-forward neural network with the sigmoid activation function to design and to implement the Observer Agent component. The supervised learning process uses the back-propagation algorithm.

3.2.2 Genetic algorithms background

In most cases a single fitness function is defined according to one particular target and this translates into a single-objective optimization (maximization or minimization) problem. However in some cases it is more appropriate to consider simultaneously several objectives. A multi-objective genetic algorithm assesses the fitness of individuals by using several evaluation functions. We do use a multi-objective genetic algorithm to consider various musical objectives.

We also use the technique known as *elitism* [214] which helps in preserving wanted characteristics and the technique known as *explicit memory* which helps in keeping the most recent best individuals in the current population.

Genetic Algorithms have been used in several scenarios where a dynamic output is required and the fitness value changes over time. Examples include dynamic GA for problems in which objective functions of parameters change periodically in a binary space [110] or in a real space [90, 111, 74].

In order to apply Genetic Algorithms to dynamic environments we need to make some adjustments to the standards GA used for static environments. One can adopt several kinds of strategies, each having its own advantages and drawbacks. In the literature four broad strategies can be considered: increase diversity after changes [83], maintaining diversity throughout the run [111], implicit or explicit memory [74, 179, 187] and multiple subpopulations [205].

3.3 Environment and Musical States

In this Section we provide details about how we selected the subjective and objective controllers that have to be taken into account when composing background music.

3.3.1 Environment State

The set of parameters needed to describe the environment and the musical state has been chosen considering the results of a test questionnaire administered to a group of 10 music teachers. The questionnaire (available online[4]) consisted of 3 questions, one regarding the environment state (Question 1) and two regarding the musical state (Questions 2 and 3).

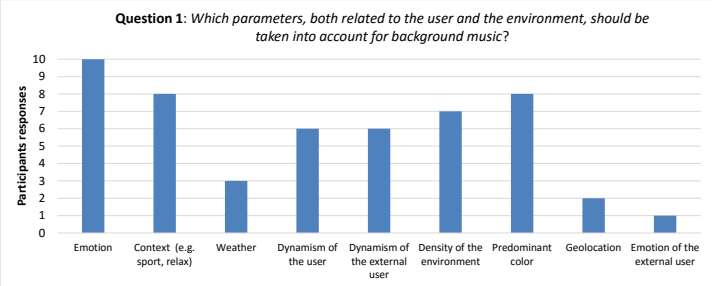


Figure 3.1: Environment parameters suggested choices (Question 1).

In Fig. 3.1 we show a summary of the responses for Question 1. As one can see in the figure, all participants considered the *emotion of the user*

[4]https://goo.gl/qAEu4Y

an important parameter while just one expressed his propensity to include the *emotion of other people* in the set of the parameters needed to compose background music. We decided to use the parameters that received a clear majority of supporting answers, namely the ones specified in Table 3.1. We refer to such parameters as *controllers*.

Feature description	Name	Range
Objective		
Dynamism of the user	e_1	$0, 1, \ldots, 100$
Dynamism of external objects	e_2	$0, 1, \ldots, 100$
Density of the environment	e_3	$0, 1, \ldots, 100$
Context	e_4	$0, 1, \ldots, 5$
Predominant color	e_5	$(x, y, z), 0 \leq, x, y, z \leq 255$
Subjective		
Emotion of the user	e_6	(x, y) coordinate

Table 3.1: Controllers.

The parameter e_1 indicates how rapidly the user is moving in the environment (0 means a slow user, 100 means a rapidly moving user). The parameter e_2 refers to the speed of other objects available in the environment. The parameter e_3 indicates how dense is the environment, i.e. how many entities, including the user, are present in the environment. The parameter e_4 describes the possible "contexts", such as: fun ($e_4 = 1$), sport ($e_4 = 2$), game ($e_4 = 3$), film ($e_4 = 4$) and relax ($e_4 = 5$). The parameter e_5 describes the predominant color of the environment, in terms of Red-Green-Blue (RGB) components.

Finally, e_6 describes the emotion of the user, represented by using a two-dimensional model of affective experience. Specifically, we used a graphical representation of the circumplex model of affect with the horizontal axis representing the valence dimension (i.e., the intrinsic attractiveness or aversiveness) and the vertical axis representing the arousal (i.e., a state of heightened activity that makes us more alert) or activation dimension [11]. As shown in Fig. 3.2, for this circular representation that includes 28 words, a greater similarity between two words is represented by their closeness in the space. We enhanced this model with a layered subdivision of the entire space in three subspaces, i.e., a high (serious-like), medium (light-like) and low subspace (quiet-like), according to the intensity of the emotion.

To summarize, we use a *6-tuple* $E^i = [e^i_1, \ldots, e^i_6]$ to describe the state of the environment at the i^{th} time interval. For example $E^{12} = [18, 12, 80, 3, (255, 0, 0), (10, -1)]$ says that at the 12^{th} time interval the user is slowly moving ($e_1 = 18$) and pleased ($e_6 = (10, -1)$, and there is a quite crowded environment ($e_3 = 80$) with other entities moving, on average, slower than the user ($e_2 = 12$), in a game context ($e_4 = 3$), and that the predominant color is red ($e_5 = (255, 0, 0)$).

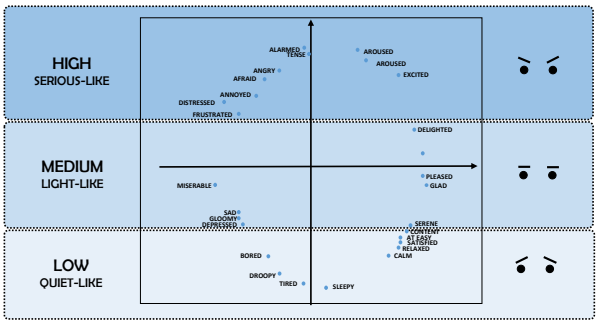
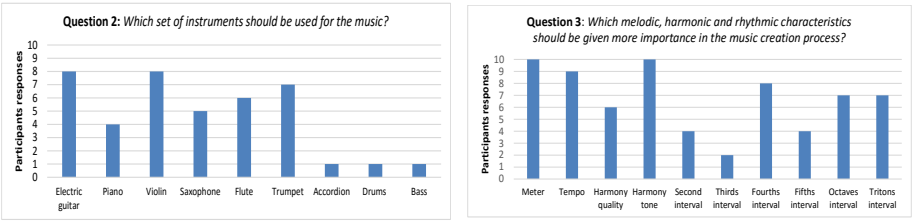


Figure 3.2: Circumplex model of affect organized in areas according to the intensity of the emotion.

3.3.2 Musical State

The environment state is built through information obtained in input from the surrounding environment, and it is not used to create music directly. Instead, it is used to compute a musical state.

The set of parameters included in the musical state has been chosen similarly to the set of parameters used for the environment state. Specifically, as shown in Fig. 3.3, we used the results of the Questions 2 and 3 that we asked to the group of 10 music teachers. Again, the set of parameters that we have included in the musical state, shown in Table 3.2, were chosen among the choices with a clear majority of supporting answers.



(a) Question 2. (b) Question 3.

Figure 3.3: Participants choices for the musical state.

These parameters are grouped into 4 classes: instrumental, rhythmic, harmonic and melodic class.

The instrumental class contains 5 boolean variables $m_1, \ldots, m_5$ that indicate whether the corresponding instrument is used. In the rhythmic class there are three parameters. The first one, m_6, indicates the meter of the music. We consider the following meters: $\frac{2}{4}$ meter ($m_6 = 0$), $\frac{3}{4}$ meter ($m_6 = 1$), $\frac{4}{4}$ meter ($m_6 = 2$), $\frac{6}{8}$ meter ($m_6 = 3$), $\frac{9}{8}$ meter ($m_6 = 4$), $\frac{12}{8}$ meter ($m_6 = 5$). The second parameter m_7 gives the velocity of the execution (tempo) ranging from *Grave* ($m_7 = 40$), that is very slow, to *Prestissimo* ($m_7 = 208$),

Feature description	Name	Range
Instrument		
Electric guitar	m_1	binary value
Violin	m_2	binary value
Saxophone	m_3	binary value
Flute	m_4	binary value
Trumpet	m_5	binary value
Rhythm		
Meter	m_6	$0, 1, \ldots, 5$
Tempo	m_7	$40, 41, \ldots, 208$
Harmony		
Quality	m_8	binary value
Melody		
Melodic thirds	m_9	Integer
Melodic fifths	m_{10}	Integer
Melodic octaves	m_{11}	Integer
Melodic tritons	m_{12}	Integer

Table 3.2: Melodic, harmonic and rhythmic parameters for the musical state.

that is very quick.

The harmonic class contains only one parameter, m_8 which is a boolean variable that indicates whether the music is in a minor tonality ($m_8 = 0$) or in a major tonality ($m_8 = 1$).

Finally in the melodic class we have four parameters, $m_9, \ldots, m_{12}$, which provide some information about the melodic lines. These parameters give the number of jumps of thirds, fifths, octaves and tritons in the melody, respectively.

To summarize, a musical state can be described with a 12-tuple $[m_1^i, \ldots, m_{12}^i]$. As an example, the musical state $M^{15} = (0, 1, 0, 0, 1, 2, 90, 8, 5, 6, 0)$ for the 15^{th} time interval says that for such an interval the Composer agent should use orchestral strings and a trumpet $((m_1, m_2, m_3, m_4, m_5) = (0, 1, 0, 0, 1))$, the meter of the music should be 4/4 ($m_6 = 2$), the tempo 90 beats per minute ($m_7 = 90$). The total melodic jumps to be used are 8,5,6,1 for, respectively, thirds, fifths, octaves and tritones $((m_9, m_{10}, m_{11}, m_{12}) = (8, 5, 6, 1))$.

3.4 EvoBackMusic components

In this Section we describe EvoBackMusic. EvoBackMusic consists of three agents: the *Observer Agent*, the *Composer Agent* and the *Sound Agent*. The Observer agent interacts with the environment getting input from it, and, after an appropriate transformation, relays such input to the Composer Agent. The Composer Agent is responsible of the music creation. The created music is passed to the Sound Agent which simply plays it and the music can be heard in the environment. Fig. 3.4 provides an overall view of the system showing how the agents interact among themselves and with the surrounding environment.

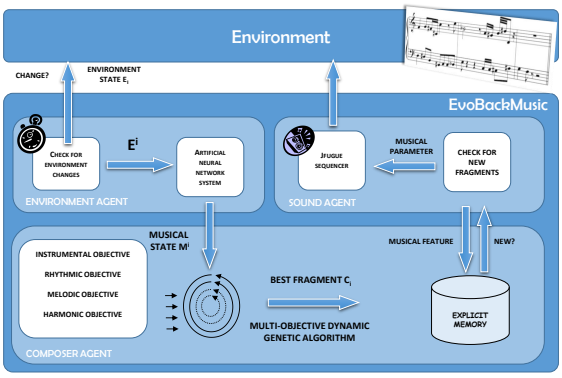


Figure 3.4: EvoBackMusic architecture: components and interactions.

3.4.1 The Observer Agent

The Observer Agent checks the state of the environment looking for changes with respect to the latest update, at fixed intervals of time. We consider interval of times of a fixed length, say T seconds, where T is a fixed global parameter of the system. We will denote with E^i the state of the environment at the beginning of the i^{th} time interval, that goes from $(i-1) \cdot T$ to $i \cdot T$ seconds, starting with the first time interval I^1.

Once obtained the environment state the Observer Agent maps it into a Musical State. In order to do so the Observer exploits an artificial neural network. Such a network must be trained in advance according to the musical preferences of the user. This means that the user, in a preliminary training phase, acts has an implicit composer for the music that he will hear later on.

The artificial neural network is a fully connected three-layer feed-forward neural network with sigmoidal activation function. The input layer models the environment, and therefore contains 9 neurons: e_1, e_2, e_3, e_4, $e_5(x)$, $e_5(y)$, $e_5(z)$, $e_6(x)$, $e_6(y)$, where $e_5(x)$, $e_5(y)$, $e_5(z)$ are the x, y, z coordinates of e_5, respectively, and $e_6(x)$, $e_6(y)$ are the x, y coordinates of e_6 (see the Section 3.3.1 to recall the meaning of the variables). The output layer represents a musical state and, therefore, contains 12 neurons: $m_1, \ldots, m_{12}$ (see the Section 3.3.2 to recall the meaning of the variables). We recall that each neuron (both input and output) represents a value normalized between 0 and 1.

The number of neurons in the hidden layer is a critical choice, since a high number could wrongly involve more equations than free variables, resulting in a system with a smaller generalization ability, while a low number could imply a system with a smaller learning ability and, therefore less robust. Also the learning rate and the momentum are crucial parameters of the neural network.

In order to fine tune the network structure we performed a training phase. For such a training phase we fixed a data set of 50 pairs of environment states and musical states. For each possible choice of the number of neurons in the hidden layer (10 random choices), value of the learning rate and the value of the momentum (9 possible values in a range between 0 and 1), we computed 10 times the training error by running the neural network for 100.000 epochs. Then for each of the above 810 experiments, each repeated 10 times, we computed the average training error. Finally we ranked all the 810 cases based on the average training error. Table 3.3 shows the best average training errors.

#	Error	Parameters	#	Error	Parameters	#	Error	Parameters
1	0.042	(18,0.7,0.5)	16	0.098	(9,0.5,0.7)	31	0.157	(18,0.5,0.7)
2	0.059	(54,0.7,0.3)	17	0.099	(36,0.5,0.7)	32	0.211	(21,0.7,0.3)
3	0.065	(41,0.7,0.7)	18	0.102	(18,0.7,0.7)	33	0.214	(9,0.7,0.5)
4	0.065	(45,0.5,0.7)	19	0.102	(13,0.5,0.7)	34	0.217	(13,0.7,0.5)
5	0.065	(54,0.5,0.7)	20	0.103	(41,0.7,0.5)	35	0.221	(36,0.7,0.5)
6	0.066	(41,0.5,0.7)	21	0.103	(30,0.7,0.3)	36	0.223	(30,0.5,0.7)
7	0.069	(13,0.5,0.5)	22	0.112	(54,0.7,0.7)	37	0.235	(36,0.7,0.7)
8	0.072	(45,0.7,0.7)	23	0.113	(45,0.7,0.5)	38	0.242	(54,0.7,0.5)
9	0.084	(13,0.7,0.3)	24	0.121	(30,0.7,0.7)	39	0.242	(36,0.7,0.3)
10	0.088	(21,0.7,0.7)	25	0.121	(27,0.5,0.7)	40	0.242	(21,0.5,0.5)
11	0.088	(27,0.7,0.7)	26	0.122	(18,0.7,0.3)	41	0.312	(30,0.7,0.5)
12	0.092	(21,0.5,0.7)	27	0.123	(9,0.5,0.5)	42	0.322	(27,0.7,0.3)
13	0.093	(9,0.7,0.7)	28	0.132	(18,0.5,0.5)	43	0.324	(9,0.7,0.3)
14	0.098	(21,0.7,0.5)	29	0.141	(13,0.7,0.7)	44	0.342	(27,0.5,0.5)
15	0.098	(27,0.7,0.5)	30	0.145	(45,0.7,0.3)	45	0.347	(41,0.7,0.3)

Table 3.3: Ranking of the training errors.

As we can see, the lowest training error is obtained by a network with 18 neurons in the hidden layer, a learning rate of 0.7 and a momentum of 0.5. Hence we choose such parameters as default values of the neural network used by EvoBackMusic. Once the network has been trained its use is straightforward: one gives the 9-value input representation of the environment state and the network produces the 13-value output that represents a musical state.

3.4.2 The Composer Agent

The Composer agent, given as input a musical state, composes a fragment of music lasting T seconds, using a dynamic multi-objective genetic algorithm. For each time interval I^i the algorithm evolves one generation and then creates a new music fragment according to the current musical state M^i.

The multi-objective genetic algorithm tries to compose "good" musical fragments. As described in Section 3.1, it considers the following objectives: the instrumental, the melodic, the harmonic, and the rhythmic objective functions. We use an *explicit memory* approach in order to continuously adapt the solution to a changing environment, reusing the information gained in the past. In the following we provide the details of the GA used by the Composer Agent.

Chromosome and gene representation.

The population is composed of individuals that are fragments of music. Each fragment lasts for T seconds which allows us to place n beats (the exact value of n depends on the actual value of T and the speed of execution of the music, which is a parameter in the musical state). So, each chromosome represents a fragment of music of duration T, and each gene of the chromosome represents a beat of the fragment.

We represent a chromosome (individual) of n genes as an array C of dimension $14 \times n$ where we store information related to the music fragment. For each beat j, the vector column at position j is a *gene*. We denote a chromosome C with $C = C_1, ..., C_n$ where C_i is the gene at beat i. Each gene C_i contains 14 values. Table 3.4 provides a summary of the music information parameters.

Feature description	Name	Range
Instrument layer		
Electric guitar activation	c_1	binary value
Violin activation	c_2	binary value
Saxophone activation	c_3	binary value
Flute activation	c_4	binary value
Trumpet activation	c_5	binary value
Rhythm layer		
Meter	c_6	$0, 1, \ldots, 5$
Tempo	c_7	$40, 41, \ldots, 208$
Melody layer		
Electrical guitar notes	c_8	array of *Note*
Violin notes	c_9	array of *Note*
Saxophone notes	c_{10}	array of *Note*
Flute notes	c_{11}	array of *Note*
Trumpet notes	c_{12}	array of *Note*
Harmony layer		
Quality	c_{13}	binary value
Tone	c_{14}	$0, 1, \ldots, 11$

Table 3.4: Gene Representation

Musical information in each gene is grouped into 4 classes: Instrumental, Rhythmic, Melodic, and Harmonic classes. The Instrumental class contains 5 boolean variables $c_1, \ldots, c_5$ that indicate whether the corresponding instrument is used. In the Rhythmic class there are three parameters. The first one, i.e., c_6, indicates the meter of the music: $\frac{2}{4}$ meter ($c_6 = 0$), $\frac{3}{4}$ meter ($c_6 = 1$), $\frac{4}{4}$ meter ($c_6 = 2$), $\frac{6}{8}$ meter ($c_6 = 3$), $\frac{9}{8}$ meter ($c_6 = 4$), $\frac{12}{8}$ meter ($c_6 = 5$). The second parameter c_7 gives the velocity of the execution ranging from *Grave* ($c_7 = 40$), that is very slow, to *Prestissimo* ($c_7 = 208$), that is very quick.

The Melodic layer contains the sequence of notes for each instrument in the instrument layer. Each note is represented as a triple *midi-value, duration, ligature*: midi-value is an integer between 21 and 108 representing the midi value of the note. Note that each type of instrument has a range of possible notes that we set as parameters. Therefore, we require that for each type of instrument, the midi value of a note has to range between the

minimum and maximum value for the instrument itself. Duration is 0 if the duration of the note is $\frac{4}{4}$, 1 if the duration of the note is $\frac{2}{4}$, 2 if the duration of the note is $\frac{1}{4}$, 3 if the duration of the note is $\frac{1}{8}$, 4 if the duration of the note is $\frac{1}{16}$, 5 if the duration of the note is $\frac{1}{32}$ and 6 if the duration of the note is $\frac{1}{64}$; ligature is a binary value such that 1 means that the note is ligate to the next note, 0 otherwise. The ligature is necessary in order to represents note whose duration exceeds the duration of a single beat. Notice that, although we could have use a representation that does not need ligatures by simply using a duration equivalent to the total duration of the ligated notes, we preferred to have at least one note in each beat because this ease the manipulation of genes.

The Harmonic layer contains the parameter c_{13} which is a boolean variable that indicates whether the music is in a minor tonality ($c_{13} = 0$) or in a major tonality ($c_{13} = 1$) and the parameter c_{14} which is an integer that indicates the current tone, ranging from 0 to represent C up to 11 to represent B (with 1 representing $C\#$, 2 representing D, and so on).

Initial population.

In the very first iteration, we start from a random initial population of K individuals, where $K = 50$ is a fixed parameter. Such a population is built by selecting a random gene for each beat. In subsequent iterations, the population is half random and half chosen among the solutions of the last iteration before the environment change. This is motivated by somehow tying the previous music to the current one so that to have a smooth passage from the music corresponding to the previous environment to that corresponding to the current environment.

Evaluation function.

The algorithm uses 4 objective functions: an instrumental objective function f_I, a rhythmic objective function f_R, a melodic objective function f_M and an harmonic objective function f_H.

Given a chromosome C and the current musical state M we compute: $f_I(C)$ by defining the *instrumental distance* between C and M, $f_R(C)$ by defining the *rhythmic distance* between C and M, $f_M(C)$ by defining the *melodic distance* between C and M and $f_H(C)$ by defining the *harmonic distance* between C and M. They represent measures of similarity (the lower the distance, the similar the states), each of them with respect to one of the classes of parameters.

The *instrumental distance* between C and M is $f_I(C) = \sum_{k=1}^{5} |m_k - \frac{\sum_{j=1}^{n} c_k^j}{n}|$. The *rhythmic distance* is $f_R(C) = \sum_{k=6}^{7} |m_k - \frac{\sum_{j=1}^{n} c_k^j}{n}|$. The *harmonic distance* is $f_h(C) = |m_8 - \frac{\sum_{j=1}^{n} c_8^j}{n}|$ (the formula for the *harmonic*

distance does not have the external summation because there is only one harmonic musical feature).

To define the *melodic distance* recall that gene C_i contains 5 melodic fragments for beat i and that there are n genes for $i = 1, \ldots, n$. For each $i = 1, \ldots, n$, let $c_{i,j}$ be the melodic fragment for beat i and for the instrument j, with $j = 8, \ldots 11$, and let $c^3_{i,j}$, $c^5_{i,j}$, $c^8_{i,j}$ and $c^T_{i,j}$ the number of melodic thirds, fifths, octaves and tritons in $c_{i,j}$ respectly. The *melodic distance* between C and M is:

$$f_m(C) = |(m_9+m_9+m_{10}+m_{11}) - \sum_{i=1}^{n} \frac{(\sum_{j=1}^{5} c^3_{i,j}) + (\sum_{j=1}^{5} c^5_{i,j}) + (\sum_{j=1}^{5} c^8_{i,j}) + (\sum_{j=1}^{5} c^T_{i,j})}{5n}|$$

The objective is to minimize each of the evaluation functions, that is we want to get a chromosome that is as close as possible to the originating musical state.

Evolution operators.

In order to evolve the population, we apply *mutation* and *crossover* operators. Moreover, we used the elitism technique to improve the individuals in the current population by keeping an *external population* P_e that selects chromosomes that have at least one of the following characteristics: *(1)* genes with the same harmonic features, *(2)* genes with the same instruments activated, and *(3)* genes with the same rhythmic features.

In each evolution step we apply the elitism operator to update P_e, by inserting new elements that satisfy the above criteria; we then choose the best K individuals to keep the external population at size K. Note that the elitism operator does not produce new individuals; it only preserves some individuals in the hope that future generations will inherit some of their characteristics.

We defined the *production population* P_p as the set consisting of K chromosomes from the current population and K chromosomes from the external population P_e. Moreover we also included in P_p the individuals that have been recently given in output; these individuals are kept in an *explicit memory*; this strategy has been suggested in [172]. The crossover and mutation operators produce new individuals starting from the ones in P_p.

- **Classic crossover.** This operator works like a standard crossover operator. Given two chromosomes $C^1, C^2 \in P_p$, it selects an index j randomly, and generates the chromosome $C^3 = C^1_1, \ldots, C^1_j, C^2_{j+1}, \ldots, C^2_N$.
- **Classic mutation.** This operator creates new chromosomes starting from a chromosome of P_p. $\forall C = C_1, ..., C_n \in P_p$, selects a random $j \in [1, n]$ and replaces the gene C_j with a random gene C'_j.
- **Emotion-Harmony mutation.** This operator acts on music quality (minor or major). Let E the actual environment state, $\forall C =$

$C_1, ..., C_n \in P_p$, if $e_6(x) \geq 0$ (positive emotion) then for the gene C'_j set $c_{14} = 1$ else $c_{14} = 0$.

- **Color-Instruments mutation.** This operator acts on the total active instruments using the number of light/dark colors. Let E the actual environment state, $\forall C = C_1, ..., C_n \in P_p$ and let $e_6(x), e_6(y), e_6(z)$ the predominant color RGB components. Let $\hat{e}_6$ be the average of the three components, then:
  - if $0 \leq \hat{e}_6 < 80$ then $\forall C'_j$ we set $c_3 = c_4 = 1$ and $c_1 = c_2 = c_5 = 0$.
  - if $80 \leq \hat{e}_6 \leq 150$ then $\forall C'_j$ we set $c_1 = c_2 = c_5 = 1$ and $c_3 = c_4 = 0$.
  - if $150 < \hat{e}_6 \leq 255$ then $\forall C'_j$ we set $c_1 = c_2 = c_3 = c_4 = c_5 = 1$.
- **Dynamism-Frequency mutation.** This operator acts on the dynamism of the environment. Let $E = e_1, e_2, ..., e_6$ the current environment state, we consider the dynamism information e_1 and e_2 and let $\hat{e} = (e_1 + e_2)/2$. $\forall C = C_1, ..., C_n \in P_p$, if $\hat{e} \leq 50$ (slow average dynamism), then $\forall$ gene C'_j, and for each sequence of notes, we first divide the sequence in two halves of equal length, possibly splitting the middle note, then we delete the second half of the sequence and we double the duration of the notes in the first half. Conversely, if $\hat{e} > 50$ we split each note of the sequence into two notes of duration equal to half of the duration of the original note.

For each generation the Composer Agent selects the best individual in the current population and writes it in P_e. Since we use a multi-objective fitness function the best individuals are those in the Pareto's front, that is the set of all non-dominated chromosomes. To have a single output we choose the rhythmic function to select the best chromosome in the Pareto's front.

Notice that, depending on the actual value of T, the evolution algorithm can produce several outputs for each time interval, since it needs less than t seconds to generate a single output. Since the Sound agent only needs one music fragment for each time interval, the algorithm takes only one of the generated outputs.

Music fragments are purged from the explicit memory after they become old enough: all the outputs relative to the last Z time intervals are kept in the memory (we used $Z = 5$), while all older the outputs are deleted.

3.4.3 The Sound Agent

The Sound Agent uses a JFugue sequencer to generate audio by using the fragments generated by the Composer Agent. Whenever the Sound Agent needs a new music fragment, that is, at the beginning of a new time interval,

it looks in the external memory and reads the first music fragment which has been given in output for the current time interval (eventually skipping older music fragments). Then, it converts the fragment to audio by mapping each fragment information (notes, rhythm and so on) into JFugue instructions (Patterns, Players and so on). The Sound agent adds also a bass and drum accompaniment by exploiting patterns from a pre-compiled library. Each specific pattern is chosen on the basis of the harmonic and rhythmic information contained in the music fragment.

3.5 Evaluation Study

Judging the music that a system produces is very difficult, since this is a uncertain process, heavily relying on personal evaluations. In fact, in [78] and [57] authors stressed the concept of subjective vision about both perceptions and emotions induced by music. They also state that music is inherently subjective and there is a strong correlation between what is seen and what is heard. Most of the works analyzed individuals' feelings in terms of impression adjectives (i.e., happy vs. sad, heartrending vs. not heartrending, and so on) [122], or subjective evaluations [123]. Finally, self-reported data has been often used to evaluate music-related systems/software [82, 145].

To evaluate EvoBackMusic we performed a subjective evaluation. Our goal was not to quantify and explain how the produced music influences the individuals' emotional states, but that of establishing whether the user is satisfied by the change in the produced music as a result of changes in the environment, which includes his/her emotional state.

3.5.1 Methodology

For our preliminary evaluation study we recruited 20 people amongst musicians and people interested in music but with little experience in music composition. We emphasize that participation in the study was voluntary and anonymous, and participants were not compensated for taking part in the interviews.

The study consisted of three phases, namely: *(1)* a *Preliminary phase* in which we administered a preliminary survey questionnaire, *(2)* a *Testing phase*, in which we asked participants to perform specific tasks, and finally, *(3)* a *Summary phase*, in which we asked our sample to fill out a summary questionnaire. All questionnaires are publicly available online[5].

The goal of the preliminary survey questionnaire was: to collect demographic information (i.e., age, education level), to get information about

[5] `https://goo.gl/qAEu4Y`

the participant's knowledge and attitude toward music, and to ask specific questions about EvoBackMusic (e.g., context for which it would be useful).

In the *Testing phase* we asked participants to create a dataset of 40 pairs of (environment states, MIDI file), by repeating the following task for 40 times:

1. Using the EvoBackMusic graphical interface, select an environment state according to some indications.

2. Select a MIDI file from a pre-defined set of MIDI files with music that the user judges appropriate for the selected environment state (this assumed that the user was familiar with the pre-defined set of MIDI file).

Then the user trained the artificial neural network using the EvoBackMusic graphical interface. Notice that although the default network uses the best parameters (number of neurons in the hidden layer, learning rate, momentum) that we have found in the training phase of the neural network, the system gives the user the possibility of changing these parameters at run time.

At the end of the *Testing phase*, the user evaluates (through a test questionnaire) EvoBackMusic by listening to the music it produces for the environment states in the dataset previously created.

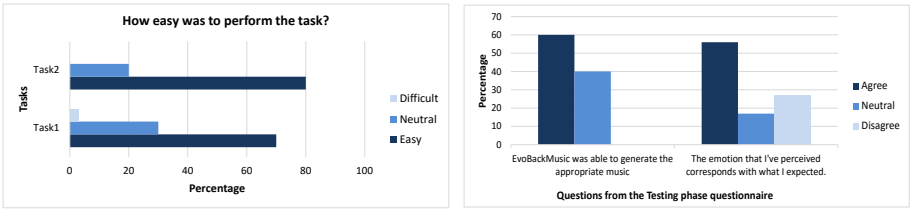
Finally, in the *Summary phase*, we administered the summary survey questionnaire whose goal was to detect whether any change has occurred in the users' opinions after gaining a greater consciousness about the potentialities of the proposed system. The summary questionnaire consisted of three questions that were already asked in the preliminary questionnaire plus one final question about the output of EvoBackMusic.

3.5.2 Results

The results of the preliminary questionnaire showed that most participants (80%) were keen on music (with 60% very keen on music), while only few (20%) were neutral. The age ranged from 30 to 50 years, and the education level included 30% with a bachelor degree, 30% with a master degree and 40% with a conservatory degree. Finally, 60% of the participants were acquainted with computers and computer music while the remaining 40% was not.

Results of the *Testing phase* showed that EvoBackMusic was very easy to use (see Fig. 3.5(a)), and that it was able to generate appropriate music, really matching the users' expectations (See Fig. 3.5(b)).

From this analysis we conclude that for almost all participants EvoBackMusic was able to produce a music suitable for their needs and individual preferences.



(a) Easiness of the tasks. (b) User expectations.

Figure 3.5: (a) 5-Point Likert scores, (b) Percentages of agreement/disagreement.

We recall that with the summary survey questionnaire we performed a comparative assessment between some specific questions asked before and after the testing phase, in order to evaluate if changes occurred in users' opinions about music importance, general knowledge about the addressed topic and the most suitable activities for which could be useful to use EvoBackMusic. About the last question, three activities, that is "Relax", "Work" and "Game", were selected more than the previous phase (percentage increase of 14%, 75%, and 25%, respectively). EvoBackMusic was also able to increase interest in music and in background music composition. Finally, at the question: *"Overall, I am satisfied with how this system produced music that matches my preferences"*, 80% of participants expressed their satisfaction.

3.6 Conclusions

In this work we have presented a real-time background music composer. The system takes into consideration many parameters in order to guide the composition process. We have implemented and tested the overall system. A video file with a simulation of the Pacman game, integrated into EvoBackMusic, can be found online (https://goo.gl/qAEu4Y). By using EvoBackMusic (previously trained) we change the states of the game, obtaining a background music for each of such changes. Future works include on one hand a more extensive test phase in order to validate the choice of the controllers that describe the environment and on the other hand the integration of the system into real or virtual environment (games, movies, augmented reality, and so on).

Chapter 4

MarcoSmiles

4.1 Introduction

In this chapter we describe a system [22], named *MarcoSmiles*, specifically designed to allow individuals to perform music in a easy and innovative way. The idea is to design new interaction modalities while composing music using *virtual* instruments, that is, by using hands without the support of a real musical instrument.

4.2 Background

4.2.1 Natural User Interfaces

Technology is constantly moving to provide tools enhancing life in a natural seamless manner, and engaging users in using computers, breaking the barriers raised since accessibility issues, inter-generational gaps, lack of attractiveness and playfulness. In the last few years, new, advanced, and enjoyable techniques for interacting with computers have been explored. New user interfaces utilizing gestures, contextual awareness, and 3D environments, both real and virtual, could entail a positive attitude towards modern technologies and computers, making their access and usage more easier, appealing, and intuitive, and improving the overall user experience as well.

The main goal of Natural user interfaces is to enhance the traditional physical interactions with a much wider set of innovative communication modalities, by overcoming limitations imposed by traditional interactions, and making them more enjoyable and usable. The popularity of these innovative interfaces is constantly growing, given the spread of low-cost devices for tracking movements, such as Leap Motion and Kinect, and the pervasiveness of voice recognition solutions. Although their high popularity in the computer games market [139], these technologies have been employed in several fields ranging from serious games [129] to rehabilitation [130], from

robotic [134] to home automation [135] and so on. Using innovative user interfaces to enrich, with devices for hand tracking, the user interaction with computers to compose music is the main objective of our work. Specifically, we focus to the field of Music Performance, by envisioning a system that all users, i.e., musicians, people generally interested in music, impaired users, and so on, could use for learning activities, music composition, or simply to entertainment activities.

In general, the approach used by developers in this field is to create virtual musical instruments very similar to the real ones, with mechanical or technical simplification only. Some other approaches are based on sensors placed only on the hands, sometimes through gloves, which provide information about hand gestures, used to generate music([121], [63], [95]). However, often these approaches, do not take into account the real needs of the musician, because they have not the objective to represent a real musical instrument; conversely, they realize a musical synthesis process, by means of a sound synthesizer, based on granular additive synthesis algorithms, that exploit information about to the hands.

For most of real musical instruments, musical notes can be expressed in terms of *hand configurations*. Each hand configuration contains information about the hand position in the space and the fingers bend in a given time. The real time mapping of gestures into music happens through the transformation of hand and fingers positions into specific music notes.

MarcoSmiles is system that maps hands' configurations into music in real-time, by allowing users to feel music, even interacting with an *abstraction* of the real musical instrument, and not with the real instrument itself.

4.2.2 Leap Motion Controller

Our interest in this system is about the use of Leap Motion devices to track finger movements and map them into musical notes.

Leap Motion is a small USB peripheral device whose was technology was firstly developed in 2008 and finally released by Leap Motion Inc. company in 2013. The device uses two monochromatic IR cameras and three infrared LEDs to observe the area (it has a 135° field of view) to a distance of about 1 meter. Specifically, the device is able to track all 10 fingers up to a precision of 1×10^{-3} of a millimeter [67]. The LEDs generate pattern-less IR light and the cameras generate almost 300 frames per second of reflected data, which is then sent through a USB cable to the host computer, where it is analyzed by the Leap Motion controller software, synthesizing 3D position data by comparing the 2D frames generated by the two cameras. It provides three types of spatial information: *(1)* the location of fingers, the hand, and pen-like objects in Euclidean space; *(2)* the motion vectors for individual fingers and pen-like objects; and *(3)* the spherical representations of hand curvature. These features enable users to interact with their desktop computer through

natural hand gestures, such as, waving, pinching or swiping. Open API can be used by developers to design and implement applications ranging from 3D graphic manipulation to motion games.

Important issues to be considered when using Leap Motion to implement applications are about the types and the number of gestures. Results of a study showed that bimanual interaction (with two hands) and linear gestures are able to significantly improve performance [138]. However, given the position to maintain to use it (hands in front of the body in order to perform tasks, to avoid the "Gorilla Arm Syndrome" it is important to avoid gestures that are overly complex and require high accuracy [139]. Finally, authors in [142] suggest, while designing an application, to limit the number of gestures since users can remember an average only three gestures at a time.

Several studies showed the using of the Leap Motion for music applications. In [143] the authors present a gestural composition and performance module for the Rubato Composer [144] software originally controlled with a mouse or multitouch trackpad, in order to support the Leap Motion device. They interpret the usable space above Leap Motion as a three-dimensional coordinate system where each fingertip is either interpreted as a musical object such as a note, oscillator, or modulator, or as an active agent within a transformation. In [49] a preliminary study and evaluation of the Leap Motion sensor as a tool for building Digital Musical Instruments is provided. The study focuses on conventional music gestures that can be recognized by the device and on the analysis of precision and latency of these gestures. Another example of system for composing music by means the Leap Motion controller is musicAir [52]. It is an interactive gesture-based technology that allows users to create music out of thin air in a fun and engaging way. Hand and finger gestures map to corresponding visuals and sounds in a collaborative and playful interface.

4.2.3 Feed-Forward Neural Networks Classifier

Neural networks are widely used to reproduce some activities of the human brain, for example, perception of images, pattern recognition and language understanding. They are a very useful tool also for gesture recognition.

One of the most commonly used neural network is the fully connected three-layer feed-forward neural network [9]. The neurons activation function is the tangent function, and we used it in our work.

Our problem can be seen as a *classification problem* in which the network, given an hand configuration, has to be able classify it in one of k classes $N_1, N_2, ..., N_k$, that are the musical notes. We remark that the set of musical notes is not fixed a priori, but is chosen by the user.

4.3 MarcoSmiles System

In this section we present MarcoSmiles. The system uses the information contained in the user's hand configurations only, and maps hand configurations into real-time musical notes.

The system consists of two components: an hardware component and a software component. The hardware component is the Leap Motion Controller and it is used for digitizing hands configurations. The software component is composed of three different parts. The first, through an implemented Java Graphical User Interface, allows the user to select a MIDI instrument (reference instrumental timbre) and to choose the range of notes to play. The second part implements the Artificial Neural Network, trained with the preferences of the user, whose main goal is to recognize the user's hand configurations. The last component associates hand configurations to musical notes.

There are not pre-represented instruments. The abstraction of physical and structural features of a musical instrument is built by using the preferences of the user about their hands configurations. We will describe both components in the following.

4.3.1 Input representation

An *hand configuration* contains information about position and orientation of an hand and each of his fingers, in a given time. We denote with C_{LH} a left hand configuration, and C_{RH} a right hand configuration. Furthermore, a pair (C_{LH}, C_{RH}) is called *hands configuration*.

The gestures of the hands are represented as a list (sequence) of hands configurations, as a function of the time:

$$(C_{LH}, C_{RH})^1, (C_{LH}, C_{RH})^2, (C_{LH}, C_{RH})^3, ...$$

So each $(C_{LH}, C_{RH})^i$ represents the hands configuration at time i. We get this list from the Leap Motion controller.

4.3.2 The hardware component

The hardware component, as aforementioned, is the Leap Motion Controller, used to capture information about the hands. The Leap Motion Controller is a low-price data glove suitable for gaming and 3D virtual environments.

It has been used in several works and in different fields, such as, industrial robotics [33], 2D video-based augmented reality [28], gesture and handwriting recognition applications [35] and so on.

Leap Motion was intended to be a human-computer interface, not general purpose 3D scanner, so it is optimised for recognizing human hands and pointy objects. It is a small USB peripheral device which is designed to be

placed on a physical desktop, facing upward (see from Fig. 4.1). The main data container we get from Leap Motion API is a Frame. One frame consists of hands, fingers, pointables (objects directly visible by controller), and additional information, like gestures recognized by simple built-in recognition mechanism, frame timestamp, rotation, translation, and scaling data. Using two monochromatic IR cameras and three infrared LEDs, the device observes a roughly hemispherical area, to a distance of about 1 meter. The LEDs generate pattern-less IR light and the cameras generate almost 300 frames per second of reflected data, which is then sent through a USB cable to the host computer, where it is analyzed by the Leap Motion controller software using complex maths in a way that has not been disclosed by the company, in some way synthesizing 3D position data by comparing the 2D frames generated by the two cameras.

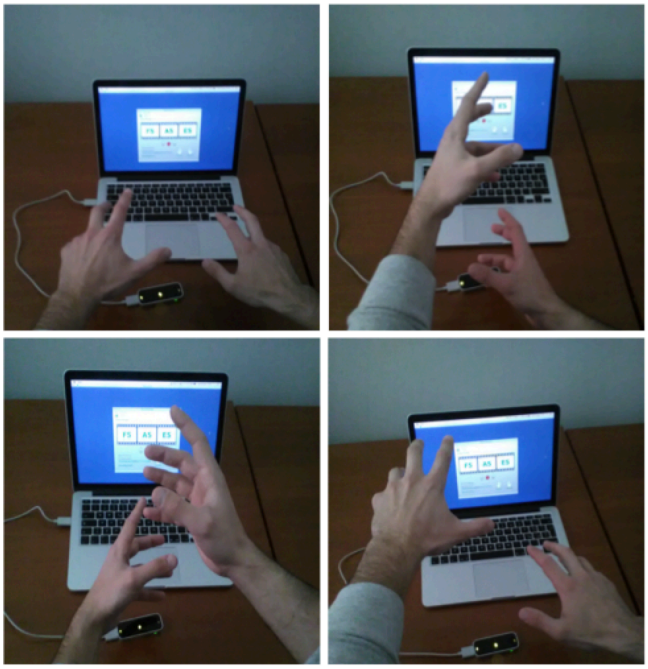


Figure 4.1: Leap Motion controller poses.

For purposes of this project, we have created our own data format. It contains only information necessary for us and allows to easily save captured frames to file, and read them later for processing and testing purposes.

4.3.3 The software component

The software component is composed of three different modules, each of them is described in the following.

The Musical module

This module provides a Java graphical interface through which the user can:

- Choose a reference instrumental timbre
- Choose the range of musical notes to play by using the Leap Motion controller

The proposed system is *continuous* in the sense that the user simply uses the Leap Motion controller without breaking down the entire sequence of gestures into separate single gestures: the system continuously checks for the hands configuration changes, and uses the neural network to recognize the new notes.

The Neural Network module

In this section we describe the neural network implemented for the recognition of music hands configurations. We provide details about the layers and the data representations.

The layers We remark that in this work we used a feed-forward three-layer neural network. We have tested several architectures of neural networks, with different learning rate and momentum. As we can see in the Section 4.4 we have obtained the best results by using the following feed-forward three-layer neural network. The input layer consists of a N_{in} neurons that take as input the hands configuration. The output layer consists of N_{out} neurons that represents the number of notes chosen by the user. The hidden layer consists of $N_{in} \times 2$ neurons.

The actual number of input neurons depends on the particular data representation that we use, while N_{out} is the number of notes in the range chosen by the user. As we will see shortly, we have used several data representations. Furthermore, we will show that we have obtained the best results with learning rate 0.4 and momentum 0.7.

Data representation Our goal is to recognize hands configurations. Such configurations, made by the user by using the Leap Motion controller, represent musical notes that the system has to play. We remark that the system is customizable so that the set of recognized hand configurations can be established by the final user.

In this section we describe the data representations that we use for the neural network. We remark that the choice of the data representation is a crucial step in the design of a neural network. The output of the network heavily depends on such a choice. The Leap Motion software uses an internal model of a human hand and provides information about each finger on a

hand. Fingers are identified in the model by type name, i.e. thumb, index, middle, ring, and pinky. For each finger, tip position and direction vectors provide the position of a finger tip and the direction in which a finger is pointing. All fingers contain four bones ordered from base to tip and they are identified as metacarpal, proximal, intermediate and distal. The thumb model, for ease of programming, includes a zero-length metacarpal bone so that this finger has the same number of bones as the other ones.

In a classification problem, for each sample we have to consider a proper set of features for the gesture recognition. A naive solution would be to use the raw data from Leap Motion sensor as the feature set. But this solution provides poor results as the raw data points are dependent on the position, orientation and scale of hand. Even a small movement in any direction resulted in decrease of classification accuracy. The literature suggests to compute a set of features invariant to wanted transformations, which can allow to fully distinguish between different classes [36, 42]. In our approach, a gesture is treated as the same one independently with respect to the translation, rotation and scale of the hand. This assumption means that the static gesture rotated by unknown angles, translated in the sensor coordinate system and also with different hand sizes should still be recognized as the same gesture.

Several feature sets were proposed and tested. For our goals, we have considered the following features:

1. **Finger flexion (FF):** for each finger we obtained the *flexion angle* by calculating the angle between the direction vectors associated to metacarpal and distal bones (see Figure 4.2).

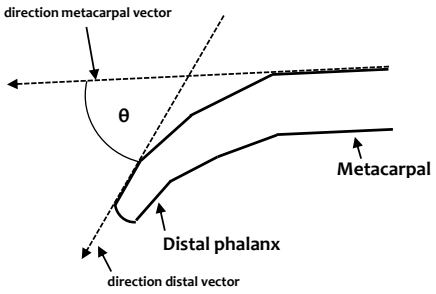


Figure 4.2: Finger flexion θ.

2. **Nearest fingers angles (NFA):** we obtained the *angle between each two consecutive fingers* by calculating the angle between finger directions vectors of two consecutive fingers (see Figure 4.3).

3. **Fingers-palm angles (FPA):** given the first finger relative to palm position, for each other finger we calculated the angles between these.

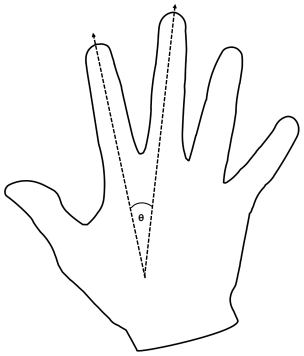


Figure 4.3: Nearest Finger angle θ.

To calculate this angle three positions are needed: two finger tip positions and a palm position. The next step is to determine the line segments between palm position and finger tip positions. The searched angle is between two designated segments (see Figure 4.4).

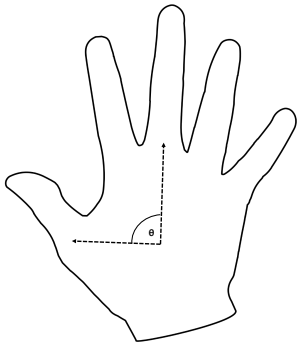


Figure 4.4: Finger-palm angle θ.

We experimented 7 alternative data representations obtained by the combination of the previously described features. We have compared the output of each type to determine the best data representation among the proposed 7 alternatives. Given a configuration (C_{LH}, C_{RH}) in input, we use the information of the Leap Motion controller to obtain the data. We denote with l_5, l_4, l_3, l_2, l_1 the left pinky, ring, middle, index and thumb finger respectly, while we denote r_5, r_4, r_3, r_2, r_1 the right pinky, ring, middle, index and thumb finger respectly.

- **FF:** we use only information about the finger flexion of each finger. Thus, we set $C_{LH} = [R(l_5),\ R(l_4),\ R(l_3),\ R(l_2),\ R(l_1)]$, and $C_{RH} = [R(r_5),\ R(r_4),\ R(r_3),\ R(r_2),\ R(r_1)]$, where $R(l_i)$ (resp. $R(r_i)$) is the radiant of the finger flexion of l_i (resp. r_i), for $1 \leq i \leq 5$. So, by

using this representation, the first-layer I of the network consists of 10 neurons.

- **NFA:** we use only information about the nearest fingers angles of each consecutive two fingers. Thus, we set $C_{LH} = [R(l_5, l_4), R(l_4, l_3), R(l_3, l_2), R(l_2, l_1)]$, and $C_{RH} = [R(r_5, r_4), R(r_4, r_3), R(r_3, r_2), R(r_2, r_1)]$ where $R(l_i, l_{i-1})$ (resp. $R(r_i, r_{i-1})$) is the radiant of the angle between of l_i and l_{i-1} (resp. r_i and r_{i-1}), for $2 \leq i \leq 5$. So, by using this representation, the first-layer I of the network consists of 8 neurons.

- **FPA:** we use only information about the fingers-palm angles. Thus, we set $C_{LH} = [R(l_5, l_1), R(l_4, l_1), R(l_3, l_1), R(l_2, l_1)$, and $C_{RH} = [R(r_5, r_1), R(r_4, r_1), R(r_3, r_1), R(r_2, r_1)$ where $R(l_i, l_1)$ (resp. $R(r_i, r_1)$) is the radiant of the angle between of l_i and l_1 (resp. r_i and r_1), for $2 \leq i \leq 5$. So, by using this representation, the first-layer I of the network consists of 8 neurons.

- **FF + NFA:** we use a combination of FF and NFA representations. So, we set $C_{LH} = [R(l_5), R(l_4), R(l_3), R(l_2), R(l_1), R(l_5, l_4), R(l_4, l_3), R(l_3, l_2), R(l_2, l_1)]$ and $C_{RH} = [R(r_5), R(r_4), R(r_3), R(r_2), R(r_1), R(r_5, r_4), R(r_4, r_3), R(r_3, r_2), R(r_2, r_1)]$ where $R(l_i)$ (resp. $R(r_i)$) and $R(l_i, l_{i-1})$ (resp. $R(r_i, r_{i-1})$) as defined before. So, by using this representation, the first-layer I of the network consists of 18 neurons.

- **FF + FPA:** we use a combination of FF and FPA representations. So, we set $C_{LH} = [R(l_5), R(l_4), R(l_3), R(l_2), R(l_1), R(l_5, l_1), R(l_4, l_1), R(l_3, l_1), R(l_2, l_1)]$ and $C_{RH} = [R(r_5), R(r_4), R(r_3), R(r_2), R(r_1), R(r_5, r_1), R(r_4, r_1), R(r_3, r_1), R(r_2, r_1)]$ where $R(l_i)$ (resp. $R(r_i)$) and $R(l_i, l_1)$ (resp. $R(r_i, r_1)$) as defined before. So, by using this representation, the first-layer I of the network consists of 18 neurons.

- **NFA +FPA:** we use a combination of NFA and FPA representations. So, we set $C_{LH} = [R(l_5, l_4), R(l_4, l_3), R(l_3, l_2), R(l_2, l_1), R(l_5, l_1), R(l_4, l_1), R(l_3, l_1), R(l_2, l_1)]$ and $C_{RH} = [R(r_5, r_4), R(r_4, r_3), R(r_3, r_2), R(r_2, r_1), R(r_5, r_1), R(r_4, r_1), R(r_3, r_1), R(r_2, r_1)]$ where $R(l_i, l_{i-1})$ (resp. $R(r_i, r_{i-1})$) and $R(l_i, l_1)$ (resp. $R(r_i, r_1)$) as defined before. So, by using this representation, the first-layer I of the network consists of 16 neurons.

- **FF + NFA + FPA:** we use a combination of FF, NFA and FPA representations. So, we set $C_{LH} = [R(l_5), R(l_4), R(l_3), R(l_2), R(l_1), R(l_5, l_4), R(l_4, l_3), R(l_3, l_2), R(l_2, l_1), R(l_5, l_1), R(l_4, l_1), R(l_3, l_1), R(l_2, l_1)]$ and $C_{RH} = [R(r_5), R(r_4), R(r_3), R(r_2), R(r_1), R(r_5, r_4), R(r_4, r_3), R(r_3, r_2), R(r_2, r_1), R(r_5, r_1), R(r_4, r_1), R(r_3, r_1), R(r_2, r_1)]$ where $R(l_i)$ (resp. $R(r_i)$), $R(l_i, l_{i-1})$ (resp. $R(r_i, r_{i-1})$) and $R(l_i, l_1)$ (resp. $R(r_i, r_1)$)

as defined before. So, by using this representation, the first-layer I of the network consists of 26 neurons.

For each representation, the output of the network is a string of m bits, where m is the number of different musical notes (classes) that we have to recognize. Each class corresponds to a sequence of $m-1$ bits set to zeroes and exactly one but set to one. The class C_1 is codified with $1, 0, \ldots, 0$, C_2 is codified with $0, 1, \ldots, 0$ and so on. Hence the third layer of the network consists of m neurons.

The Training-Playing module

The objective of this system is to learn the mapping between the hand configurations and the musical notes, as presented by the user, and then to play the notes. We can distinguish two phases respect to the use of the neural network: the training phase and the playing phase.

During the training phase, the user uses the Leap Motion controller to show the mapping between hand configurations and musical notes. In this phase the Virtual Environment creates the training set by using several *training-sessions*. Each training-session contains a list of notes randomly selected from the range of notes that the user has defined. For each note, the system allows the user to show the corresponding hand configuration within a time interval. So, the serial listener builds the configuration by using the values, according to the specific representation, and saves the data in the training set. When the user has completed the sessions, then he can start the training of the network.

During the playing phase, the user uses the Leap Motion controller to test the neural network and to play music. At each time i the serial listener receives data from the glove and builds the corresponding hand configuration. Once the hand configuration has been created, the system uses the trained network to recognize the finger configuration, and to return a music note to play. In order to play music, a Java *midi player* has been developed by using the Java Midi Library.

4.4 Efficacy Study

In MarcoSmiles each user designs his virtual musical instrument. This process consists of two main important steps: the learning of the configurations of the user associated to a range of musical notes by using a neural network, and the playing of music by using a Leap Motion controller.

In order to obtain these results we faced an important problem: the choice of the most efficient representation. To this, for each representation we have chosen the most efficient training and architecture parameters for

the neural network. In this Section we show some results about the efficacy of MarcoSmiles, according to the choices that we made.

We have to emphasize that in MarcoSmiles the training phase is separated from the use of the system, in fact, before using MarcoSmiles, the user has to train the system, in order to allow it to learn his hand configurations.

So, we give to the user the possibility to choose the training parameters in order to obtain the preferred level of accuracy.

4.4.1 ANN parameters

The choice of the architecture and of the training parameters of a neural network is an aspect very important. In this section we provide some results about them. As explained before in this work, we used a fully connected feed-forward artificial neural network with an input layer I, an hidden layer H and an output layer O.

In order to choose the best architecture (in terms of size of the layers) and the best training parameters, we have fixed a range of 36 musical notes (3 octaves), a data set of 2000 pairs *(configuration, note)* partitioned in: 1200 for the training set, 500 for the validation set and 300 for the testing set. We have run several tests for each of the 3 data representation described in the previous section. All networks were trained for about 20000 epochs.

Given a representation, in order to run a test we set the number of neurons N_H of the hidden layer H, the momentum value M and the learning rate value L, with $N_H \in \{|I|, |O|, 2|I|, 2|O|, 3|I|, 3|O|, 4|I|, 4|O|, 5|I|, 5|O|\}$, $M, L \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$. Each test configuration is run 10 times. For each of these test execution we calculated the error rate (validation error) and finally we consider the average error rate obtained with the specific test configuration.

Table 4.1 summarizes the best average results about the recognition rate. As we can see, in a range of 15000 epochs, we have obtained the highest recognition rate using the representation $FF + NFA + FPA$, with $N_H = 2|I|$, $M = 0.4$ and $L = 0.7$.

4.5 Evaluation

In this Section we describe the evaluation study that we performed to analyze the users' acceptance and, therefore, their behavioral intention to use the developed system. We first describe the methodology and, then, the interesting insights we derived when analyzing the results of the questions available in the questionnaire we administered to a sample of users recruited through word of mouth advertising.

The technology acceptance model (TAM) was first created by Davis [45], based on the theory of reasoned action (TRA) [46] in psychology research. It is a widely used theoretical model to explain and/or predict potential users'

Table 4.1: Test configurations with average recognition rate.

Representation	N_H	M	L	recognition rate %
FF + NFA + FPA	$2\|I\|$	0.4	0.7	97.2
FF + NFA + FPA	$2\|I\|$	0.3	0.3	94.3
FF + FPA	$\|O\|$	0.3	0.5	91.3
FF + NFA	$2\|O\|$	0.4	0.5	87.4
FF + NFA + FPA	$\|O\|$	0.6	0.5	84.3
NFA + FPA	$2\|I\|$	0.7	0.7	83.2
FF + FPA	$2\|O\|$	0.4	0.6	82.2
FPA	$2\|I\|$	0.3	0.3	79.5
FPA	$2\|I\|$	0.5	0.3	76.4
FF + NFA + FPA	$2\|I\|$	0.5	0.7	71.3
NFA + FPA	$2\|O\|$	0.3	0.3	70.4
NFA + FPA	$2\|I\|$	0.4	0.5	70.1

behavioral intentions to access a technology or a new system. TAM has been applied in numerous studies testing user acceptance of information technology, for example, word processors [45], spreadsheet applications, email, Web browser, websites, e-collaboration, and blackboard. In TAM, perceived usefulness refers to "*the degree to which a person believes that using a particular system would enhance his or her job performance*", while perceived ease of use refers to "*the degree to which a person believes that using a particular system would be free of effort*". According to Davis, a system that was perceived to be easy to use is also likely to be accepted by users, and user friendliness in the design of a system can increase user acceptance of that system.

However, additional explanatory variables may be needed beyond these two constructs. Moon and Kim [48], for example, extended TAM to a WWW context, by introducing "*playfulness*" as an intrinsic belief factor that affected user acceptance of the WWW. Bruner and Kumar [51] extended TAM to hand-held Internet devices and introduced a "*fun*" attribute as one of the motivational factors in the adoption of Internet devices.

We extended the TAM model in order to analyze whether perceptions of playfulness and attitude toward using, in addition to ease of use and usefulness, appear to influence behavioral intention to use MarcoSmiles.

4.5.1 Method

Participants

For our study we recruited 20 participants among both musicians and people interested in music but with little experience in music composition. Some of the musicians participants were recruited at the "Conservatorio di Potenza

Gesualdo da Venosa"[1] and at the "Conservatorio di Musica, G. Martucci", in Salerno[2]. Participants were recruited through word of mouth advertising.

We have to emphasize that the participation in the study was voluntary and anonymous, and participants were not compensated for taking part in the interviews. Participants were informed that all the information they provided would remain confidential.

Procedure

The survey consisted of four sections. The first section asked for participant demographics, technical experience, attitudes toward music and music preferences, opinions about the use of computer-based systems as tools to support people with disabilities. The 11 questions included in this first part were open-ended questions, closed-ended questions, dichotomous questions (yes/no), and finally, questions with a rating on a 5-point Likert scale with *strongly agree/strongly disagree* as verbal anchors.

In the second section we asked participants to wear the P5 glove and interact with it by performing a specific task. Additionally, at the end, we asked them to express their opinion about easiness and efficacy metrics. The task involved the following steps:

- Setting of the range of the notes
- Training of a neural network (the user uses the P5 Glove to show the mapping between hand configurations and musical notes)
- Testing and playing the system (the user creates testing sessions and evaluates the learning capabilities of the system).

Next, we asked participants to answer to the extended TAM questions. To fit the specific context of using MarcoSmiles, items about easy of use (PEOU) and usefulness (PU) were adapted from [45], items about attitude toward using (ATT) and playfulness (PPL) were adapted from [5, 48, 6]. All items were measured using a 7-point Likert scale, with 1 being *strongly disagree* and 7 being *strongly agree*.

Finally, in the last section, we submitted to participants some of the questions previously asked in the first part, in order to make a comparative assessment and to inspect whether some changes occurred after using the system. Similarly to the organization of the first part, the 11 questions were open-ended questions, closed-ended questions, dichotomous questions and questions with a rating on a 5-point Likert scale. The questionnaire administered to participants is publicly available[3] while the whole evaluation study lasted approximately 1.5 hour.

[1] http://www.conservatoriopotenza.it/

[2] http://www.consalerno.com/

[3] `http://music.dia.unisa.it/Studies/MarcoSmiles/Questionnaire.pdf`

Data Analysis

By using regression analysis we want to find the influence of the independent variables usefulness, easy of use, attitude toward the use, and playfulness (PU, PEOU, ATT, PPL), on the dependent variable behavioral intention (BI). The internal consistency reliability among the multi-item scales was examined with the Cronbach alpha [7]. Finally, questionnaire responses were analyzed using SPSS version 20[4].

4.5.2 Results

The results of the preliminary questionnaire showed that all participants were keen on music (50% of participants spent almost 2 hours per day playing music, no one less that 1 hour). The sample was fully male, with a average age of 34.6 (SD=6.8). The education level included 10% with a bachelor degree, 20% with a master degree and 70% with a conservatory degree. In terms of technical skills, 50% of the participants were acquainted with computers, and only 30% had familiarity with virtual reality. Finally, only 20% of participants had a confidence with Leap Motion device.

Results of the Testing phase showed that, on average, about 80% of participants found easy to perform the posed task. Participants also stated that the system was able to efficiently learn their fingering (100% of agreement). Finally, results about the efficacy (of the implemented neural network) was very positive since *MarcoSmiles* was able to recognize, on average, 94% of the notes defined during the *Configuration & Training* phase.

At the end of the Testing phase we asked users to respond to our modified version of the TAM questionnaire. Reliability values (Cronbach's Alpha) in terms of all participants answers is 0.96 (above the recommended threshold value of 0.70 given in the literature [2]).

As shown in Fig. 4.5, results were highly positive for all metrics. ATT was the most rated metric, showing that our participants exhibited a positive attitude toward the system with a positive evaluation of the overall idea behind *MarcoSmiles*.

The correlation analysis between the analyzed subscales is shown in Table 4.2.

Furthermore, in order to identify which variables influenced the use of *MarcoSmiles*, a regression analysis was carried out. The dependent variable was the behavioral intention to use metric (BI). The independent predictor variables were the TAM subscales. The regression analysis in Table 4.3 shows that a good predictor for the behavioral intention to use was the attitude toward the use. Specifically, it is observed that at the 0.000 significance level, ATT influences the behavioral intention to use the system. When ATT increases, the BI increases by a 1.180 factor. More precisely, 85.7%

[4] `http://www-01.ibm.com/software/analytics/spss/`

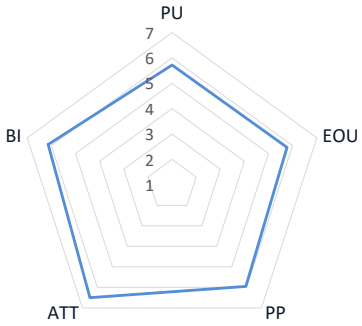


Figure 4.5: TAM results. Rating on a 7-point Likert scale. PU, Perceived Usefulness; EOU, Perceived Easy of Use; ATT, ATTitude toward using; BI, Behavioral Intention to use.

Table 4.2: Correlation coefficients between subscales. PU, Perceived Usefulness; EOU, Perceived Easy Of Use; ATT, Attitude toward using; PP, Perceived Playfulness. All correlations are significant at .01 level.

Subscale	PU	EOU	ATT	PP	BI
PU	1.0	.885	.765	.765	.791
EOU		1.0	.950	.920	.916
ATT			1.0	.978	.933
PP				1.0	.953
BI					1.0

of the BI variation is explained by ATT. Our result shows that a positive attitude toward the proposed system and interactions could influence its adoption, confirming works about that some attitudes are strongly predictive of corresponding behaviors [94].

Table 4.3: Results of multiple linear regression analysis. B, Unstandardized coefficient; β Standardized coefficient; SE, Standard Error; Adjusted $R^2 = 85.7\%$.

Predictor variables	B	SE(B)	β	t value	P value
(Constant)	-1.563	1.039		-1.504	.171
Attitude	1.180	.159	.934	7.416	.000

Results about users perceptions in terms of involvement are shown in Table 4.4. All questions were positively rated. Specifically, our participants rated very positively the responsiveness of the system (μ=6.0, $\sigma = 0.7$) and the naturalness of the interactions (μ=6.0, $\sigma = 0.8$). Furthermore, most importantly, participants felt the overall (*virtual*) experience engaging (μ=5.9) and very consistent with the real-world experience ($\mu = 5.6$).

Finally, results of the Summary Survey showed how the overall system

Table 4.4: Presence and Immersion. 7-Point Likert scale. μ=mean, σ= standard deviation.

Question	μ	σ
How responsive was the environment to actions that you initiated (or performed?)	6.0	0.7
How natural did your interactions with the environment seem?	6.0	0.8
How much did your experiences in the virtual environment seem consistent with your real-world experiences?	5.6	0.7
How well could you identify sounds?	6.5	0.5
How involved were you in the virtual environment experience?	5.9	0.7
How much delay did you experience between your actions and expected outcomes?	5.7	0.5
How proficient in moving and interacting with the virtual environment did you feel at the end of the experience?	6.6	0.5
How well could you concentrate on the assigned tasks or required activities rather than on the mechanisms used to perform those tasks or activities?	5.5	0.5
Were you involved in the experimental task to the extent that you lost track of time?	5.6	0.5

was perceived as useful, easy to use, and with an interesting underlying idea (μ=4.8, 4.7, and 5, respectively). Moreover, at the question "*Do you think that people with disabilities could successfully benefit of MarcoSmiles functionalities?*", the whole sample agreed with this possibility (μ=4.6).

Chapter 5

Music Visualization

5.1 Introduction

Music is a ubiquitous activity, that exists in every human culture, in different forms and with different users' perceptions. Everyday, individuals listen to music for several reasons: to create a remarkable atmosphere in which to dream and to evoke memories, to influence emotions, to get through health problems or to express inner feelings, to enhance creativity behaviors. The musical everyday listening is also strongly related with the mood and/or preferences of the people. No complex task or a special ability is needed. Users can sense the nature of music and what music want to evoke, without any need to understand its underlying structure.

A different situation arises listening to music requires a conscious experience and participation, when musicians have to compose music, when students have to learn complex musical rules, and so on. Here, some efforts are required to understand the structure of musical compositions. In particular, a difficult field in this area is the study of classical music. Untrained people may only be able to feel the sound elements such as pitch, rhythm, volume, and speed. Conversely, the form of the music and the *harmonic*, *melodic*, and *rhythmic* structural aspects are usually known only by musicians who have received extensive training in music history and theory. The corresponding learning curve is steep and makes classical music apparently sophisticated and of difficult comprehension.

Music expertise is the ability to understand the harmonic, melodic, and rhythmic structural elements of music compositions by reading musical scores or even simply listening to music performance. Learning musical rules is hard, especially for classical music, where the rigidity of its structures and styles require greater efforts in terms of both their understandability and applicability. The most common way to learn music is through the study of musical scores, which contains the objective notations of a music composition. However, the analysis of musical scores is demanding and beginners

have to spend considerable amount of time to learn the basics of music theory, before being able to understand the musical notations. It is also well-known that musicologists study the harmonic structure of pieces and annotate them by hand. This is a time-consuming process, especially when large corpora have to be analyzed [18]. Finally, it is important to emphasize that the knowledge of music theory is essential to understand the harmonic structure of musical compositions.

Making this task accessible to everyone, even for those who do not have *strong* knowledge of music theory, is an issue that we address in our work. We investigate whether Information Visualization techniques could be efficiently employed to support users in quickly understand complex theoretical rules. Furthermore, visualization could be efficiently used to convey music concepts in a meaningful and pleasant way.

Different attempts were made to visualize the structure of the music. A critical issue in this field is how to create an informative and insightful visualization from which users, with a poor music background, are able to *see* the sound and, at the same time, are able to *semantically understand* the underlying structure of music. The most important challenges to address when visualizing musical structures, lie in both the pre-processing of the input data (i.e., music elements) and the design of a reasonable (meaningful and intuitive) graphical representation for music.

In this chapter we describe and compare two color-based graphical representations ([25], [26]), that without involving pre-processing of musical data, allow users to understand the harmonic structure of musical compositions in meaningful and pleasant way. We extend an earlier work presented in [26], where a single approach was defined, and preliminarly evaluated.

We focus our attention on the *Harmonic Analysis Problem*: given a musical composition, the objective is to find the harmonic structure, that is, the best *harmonic succession* of chords. The harmonic succession is fundamental to ensure coherence in tonal music. Additionally, the harmonic succession term does not only refer to the simple sequence of chords, but also that such a sequence is organized according to certain rules and a certain order [12]. In most cases harmony exercises are written on two lines, using a schema of the *4-voice chorales*, which is, for example, available in the Bach's chorales harmonizations [12, 13]. Specifically, a chorale consists of 4 independent voices, called *bass*, *tenor*, *alto* and *soprano*, connected through classical music rules [12]. In our work we focus on this type of music genre.

We discuss the main features of our approaches by comparing them in terms of effectiveness (i.e., learning ability) and pleasantness (i.e., visualization). We also implemented a tool, named VisualHarmony, that implements the visualization approach evaluated as the best representation in a preliminary study involving participants from both musicians and students in the classical music field. Finally, the tool was tested by a different sample of participants, in order to achieve feedback about system usability and user

satisfaction.

The rest of the chapter is organized as follows. In Section 5.2 we present interesting works in the same field. In Section 5.3 we define the harmonic analysis problem while in Section 5.4 we discuss how we apply visualization for our aims and the characteristics of the designed approaches. In Section 5.5 we present the results of their comparison, by discussing interesting insights about improvements suggested by the participants at the study. In Section 5.6 we present VisualHarmony, while in Section 5.7 we discuss the results of a usability study aiming at explore the user satisfaction when interacting with it. Finally, in Section 5.8 we conclude with some final remarks and future directions.

5.2 Related Works

The ability to understand the harmonic structure of a musical composition is important in every genre of music, from classical to pop music. In particular, the harmonic analysis is an important step for many techniques and problems of classical musical composition, for example in the so-called *unfigured bass harmonization problem*: a bass line is given and the composer has to write other 3 voices to have a complete 4-voice piece of music with a 4-note chord for each bass note. Solving such a problem means finding the appropriate chords to use for each bass note as well as find a placement of the four notes within each chord so that melodic concerns are addressed, especially for the highest voice (soprano). In order to correctly complete this task, the composer needs to understand the harmonic structure inducted by the bass line [14, 15, 16].

A rich body of works that aim at explore the harmonic analysis rules has emerged and has attracted numerous computer music researchers to investigate the automaton of the analysis and generation of harmony. Today, automated harmonic analysis is an important and interesting music research topic. Many frameworks have been developed for the automatic harmonic analysis. An example is the Rameau framework [17], that contains a collection of re-implemented existing algorithms for the harmonic analysis in the literature and that allows to easily evaluate their accuracy, to study their errors, and compare their merits and flaws. In [18] the authors present the HarmTrace system, in which the relations between the structural elements in the harmony are represented by the productions of a CFG (context-free grammar).

From the visualization point of view, different attempts have been made to visualize music. *Arc Diagrams* represents one of the first examples to visualize repetitions in music compositions using information visualization [19, 20]. The *Isochords* system [21] is a method for visualizing the chord structure, progression and voicing of musical compositions represented in MIDI

format. Taking advantage of a Tonnetz grid, that emphasizes consonant intervals and chords, Isochords shows how harmony changes over time.

Other approaches use 3D views to visualize music components. Smith and Williams [27] discussed the possibility of visualizing MIDI in 3-dimensional space by using color to mark timbre. The *comp-i* system [29] shows the structure of music as a whole by using 3-dimensional piano roll visualization. Its main goal is to allow users to perform visual exploration of a given MIDI dataset in an immersive and intuitive manner. In [30] authors designed and implemented a simplified 3D particle system to generate real-time animated particle emitter fountains choreographed by music, for visual entertainment and for music composition. Another MIDI-based system available in literature is *Music Animation Machine* [31]. It encompasses a number of visualizations including a basic 2-dimensional piano roll notation for visualizing structure. This visualization is additionally expanded with colors based on pitch classes using the well-known circle of fifths[1]. Colors based on a circle of fifths for visualization of tonal distributions and for understanding consonance and dissonance intervals have also been explored in other works [32, 37].

Finally, in [34] the author visualizes hierarchy of key regions of a given composition, where x-axis represents time (from the start to the end of the composition) and y-axis represents the duration of key-finding algorithm's sliding window. When the window size increases, more notes are included and may affect the analyzed tonality. These hierarchical key analysis diagrams are useful for comparing the impact of using different time scales, and for viewing the harmonic structure and relationships between key regions in the composition.

From the *musical point of view*, the most significant difference with our work is that most of the discussed works focus on melodic patterns and melodic intervals. Conversely, we designed a visualization approach to address the complexities that arise when studying the harmonic structure of music compositions, that is, understanding and remembering the strict rules about modulations and sequences of degrees (i.e., cadences). Additionally, in our approaches we also take into account the relations between the degrees of a tonality, and not only between the modulations of tonalities.

From the *visualization point of view*, the graphical representations that the aforementioned works propose are disconnected by the musical scores. Conversely, our aim is to provide visual cues directly over the musical score (and therefore, without pre-processing of musical data), to convey *augmented* information, when studying harmonic compositions. Although colors based on a circle of fifths for visualization have been already proposed, none of the discussed works has conducted formal studies to assess

[1]The Circle of Fifths shows the relationships among the twelve tones of the chromatic scale, their corresponding key signatures and the associated major and minor keys.

the efficacy of the approaches they provide. In our work, we conducted a preliminary evaluation study to compare two color-based graphical representations with the aim of discovering the best one in terms of effectiveness, usefulness, and pleasantness. A usability study, instead, was conducted to assess the usability of the tool implementing this representation. Participants were both musicians with classic music skills and students without a classical musical background.

5.3 The harmonic analysis problem

We consider the tempered music system used in Western countries. Tempered music is based on the notion of *tonality*. To wit, a tonality is a group of notes which form a scale. Each tonality can be either *major* or *minor*. For example, the major scale of C is C, D, E, F, G, A, B, while the major scale of D is D, E, F#, G, A, B, C#. Western music's twelve well-tempered tonalities can be arranged in the well-known circle of fifths that orders them according to the number of alterations in their key signatures.

Usually, given a music composition, it exists a *main tonality*, and therefore notes of the corresponding scale are considered more important than notes outside the scale. Moreover, a music composition consists of a sequence of measures, whereas each of them consists of a given number of beats. Each beat is associated to a musical *chord*, that is a set of notes. Chords are identified primarily by their position in the tonality or, by the scale degree serving as root. Hence, chords succession can be reduced to root succession (or root progression), which in turn can be translated into Roman numerals representing a succession of scale degrees. Therefore, the notes of a scale are often denoted also by I, II, III, IV, V, VI, VII especially when it is important to emphasize the degree of the scale only, and not the particular note, which depends on the tonality. We refer the interested reader to a standard textbook on harmony for more detailed explanations [12].

Western music is based upon well-established harmonic and melodic rules. Several rules concern sequences of chords. Some sequences are "*better*" than others, where the term better is hard to define given its subjective evaluation. Anyway, in music community it is largely accepted that particular sequences of chords work better than others. Specifically, some chords are "*more important*" than others because they suggest, prepare, enforce or device tonal centers. Overall, the art of tonal music consists precisely in arranging chords in such a way that their interplay is pleasant and meaningful.

The *harmonic succession* of a music composition is a sequence of chords that represents one of the harmonic structures of the composition. It is worth to note that, for each musical composition, is possible to find several harmonic successions.

We focus our attention on the *harmonic analysis problem*: given a musical composition, the objective is to find the best harmonic successions of chords. Obviously, the harmonic analysis can be made on compositions of any *musical genre*. In this work, as anticipated in Section 5.1, we focus on the *chorales* genre. As an example, a fragment of the Bach's BWV26.6 chorale, with the corresponding harmonic analysis, is shown in Fig. 5.1.

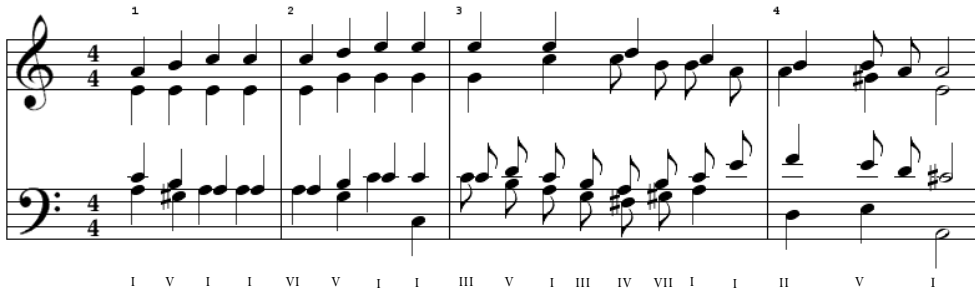


Figure 5.1: Harmonic analysis performed on a fragment of the chorale BWV26.6.

5.4 Visualization approaches for music harmony

In this section we describe the two harmonic visualization approaches designed to improve music learning activities, simplifying the process of harmonic analysis of music compositions and making it an enjoyable experience. For each approach we explain the mapping between musical constructs (i.e., tonality and degree) and the chosen graphical representations.

5.4.1 Graphical elements and concepts

In order to define a visual representation for the harmonic structure of a chorale we need to choose: *(1)* a visual representation for tonalities, and *(2)* a visual representation for the degrees. In the following we will provide details about them.

Tonality representation A musical composition starts with a main tonality, but during the sequence of chords the piece can undergo a modulation, i.e., it may change tonality. In the specific context of chorales, the harmonic rules drive the modulation among tonalities that are close in the circle of fifths.

Therefore, the choice for a visual representation has to ensure that similar tonalities have similar representations. Our idea is to map similar tonalities to similar colors, by assigning a color wheel to the circle of fifths, as shown in Fig. 5.2. In our approaches, given the musical score, we decided to:

- assign to each tonality a specific color in our enhanced circle of fifths (as an example the G major tonality corresponds to the light blue color, while the Db major tonality corresponds to the light brown color)
- highlight the area of a *Major* tonality with a rectangle having continuous lines; two rectangles, having as background color the color assigned to the tonality, are shown at the top and the bottom of the musical score (see Fig. 5.3).
- highlight the area of a *Minor* tonality with a rectangle having dashed lines; two rectangles, having as background color the color assigned to the tonality, are shown at the top and the bottom of the musical score (see Fig. 5.3).

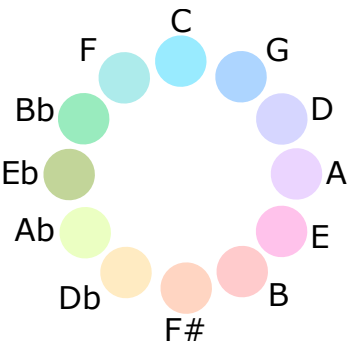


Figure 5.2: Mapping of colors to tonalities in the circle of fifhts.

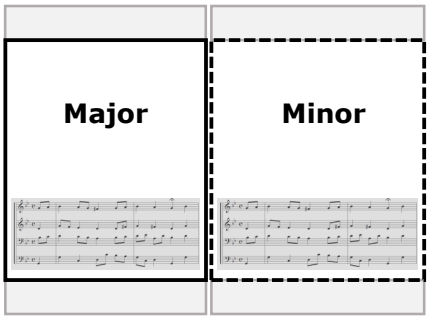


Figure 5.3: Mapping of Major and Minor tonality representations.

Degree representation Given a tonality, we decided to represent the seven degrees with rectangle shapes. It is worth to note that each degree can be classified according to its harmonic function. Specifically, I degree has *tonic* function, V degree has *dominant* function, and IV degree has *subdominant* function. Given their most important role, we decided to represent them with the primary colors (sets of colors that can be combined to make a useful range of colors), according to the following mapping: (1) I → RED, (2) IV → GREEN, (3) V → BLUE (See Fig. 5.4).

The II degree has two notes in common with the IV and the V degrees, so we decided to represent it with the secondary color CYAN, that is the color obtained by the combination of the GREEN (IV) and BLUE (V) colors. The III degree has two notes in common with the I and the V degrees, so we decided to represent it with the secondary color MAGENTA, that is the color obtained by the combination of the RED and BLUE colors. The same

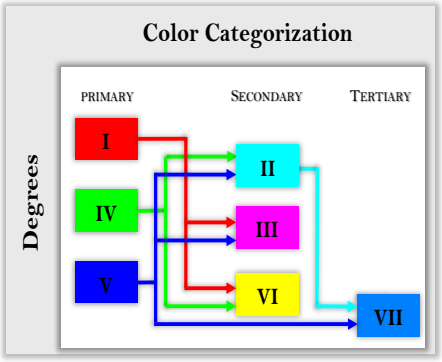


Figure 5.4: Colors assigned to degrees.

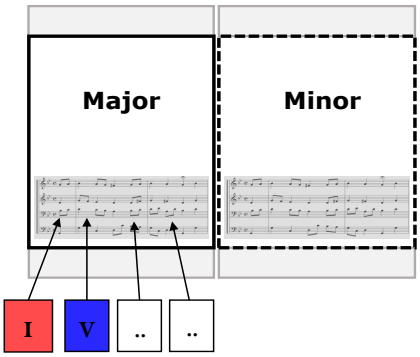


Figure 5.5: Degree representation.

happens for the VI and VII degrees whose colors, as shown in Fig. 5.4, are YELLOW and AZURE, respectively.

As an example, let us consider the degrees for the C major tonality. Notes for the I degree are C, E, and G; the notes for the V degree are G, B, and D. Notice that the notes for the III degree are E, G and B and so the III degree has two notes in common with the I degree (E, G) and two notes in common with the V degree (G, B). As result, we decided to assign as background color for the III degree the MAGENTA color, that is the color output of the combination of the background colors assigned to the I degree (RED) and the V degree (BLUE).

Moreover, as shown in Fig. 5.5, given a degree, the rectangle shapes have as background color the color assigned to the degree. If the chord of the degree is major, we use rectangle shapes having continuous lines and conversely, if the chord of the degree is minor, we use rectangle shapes having dashed lines. We want to emphasize that, if the tonality is major, then degrees I, IV, V have major chords, while degrees II, III, VI, VII have minor chords.

Finally, it is important to emphasize that we applied a slight transformation to colors (i.e., increase of transparency) in order to avoid to fully cover the musical information on the score.

5.4.2 Design and implementation

The first approach, named "*Overlay* Approach", implements the mapping of music elements in the graphical representations presented in Section 5.4.1. Specifically, given a degree, we show a single rectangle, *embedded* into the musical score, having as background color the color corresponding to the degree. It must be the smallest rectangle that includes all the notes in the corresponding beat. In this way, each rectangle could also give a melodic

description of the degree. A large rectangle denotes a beat with many notes (horizontal view), while a tall one denotes a beat with very large vertical intervals between the 4 voices (vertical view). In Fig. 5.6 we show an example of the application of this approach for the chorale BWV26.6.

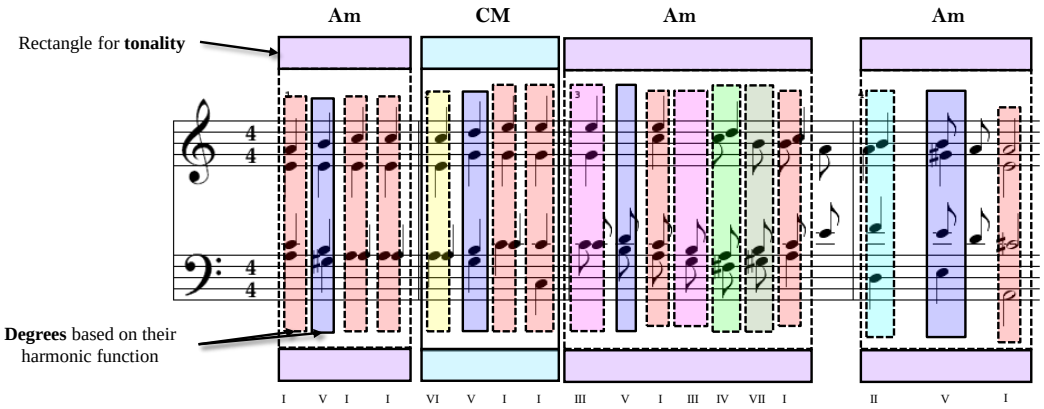


Figure 5.6: Implementation of the Overlay Approach: visualization embedded into the musical score.

Similar to the Overlay Approach, the "*Edge* Approach" uses the same mapping described in Section 5.4.1. The main difference between them is about the number and position of the rectangles for the degree. Specifically, given a degree, we show two rectangles having as background color the color corresponding to the degree, placed at the top and at the bottom of beat. What we want to do here is to verify if *embedding* visualization in the musical score may result in a confusing/disturbing/invasive representation for users and conversely, if placing the graphical elements at the *edge* of the score can avoid all the aforementioned contraindications.

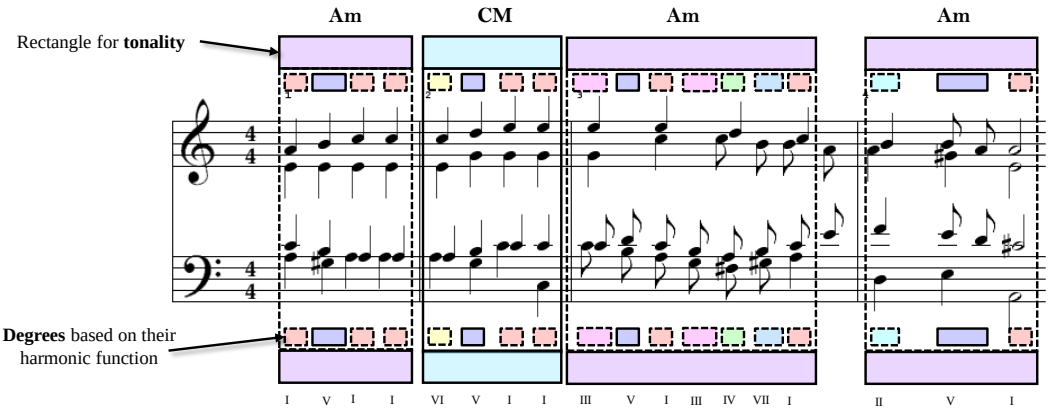


Figure 5.7: Implementation of the Edge Approach: visualization at the edge of the musical score.

In Fig. 5.7 we show an example of the application of this approach for the chorale BWV26.6 shown in Fig. 5.1.

5.5 Evaluation Study

This section begins with a description of the methodology that we employed for our evaluation study and then we discuss the results obtained. We evaluated the two approaches described in Section 5.4, in order to choose the best representation in terms of effectiveness and pleasantness. The assessment will be based on the participants performance while performing harmonic analysis, and their overall satisfaction.

Specifically, we addressed the following questions:

- Are visual constructs helpful when employed to learn complex musical rules to perform harmonic analysis?
- What is the opinion of users about the graphical representations presented? Which one impacts on their learning ability?
- What is the feeling about the best representation in terms of appreciation and satisfaction?

5.5.1 Methodology

The objective of this study is to derive users' perceptions about the best visual representation to learn classical music, in the most effective, meaningful and pleasant way. We made this analysis by measuring the effectiveness of both approaches, with the aim to develop a tool implementing the best solution.

Performance of participants were tested by comparing the two defined approaches against a standard way (*"Standard Approach"*) to perform the harmonic analysis (i.e., simply looking at the musical scores). According to the between-group design [38], we organized participants in two groups, one for approach to be tested. To avoid biased results, the participants that took part at the study were assigned to the groups in a random way.

The study consisted of three phases, namely, a Preliminary Survey, a Testing Phase, and finally, a Summary Survey, as defined and implemented in other contexts [39, 40].

In the Preliminary Survey phase, we asked participants to fill in a preliminary questionnaire in order to collect: *(a)* demographic information, *(b)* information about music background (time spent playing music, played instruments, and so on), *(c)* general attitudes toward the classical music and harmonic analysis. The questions included in this questionnaire were questions asking to give a preference up to 9 possible choices and questions on a 5-point Likert scale (e.g., Inexpert (1) to Expert (5)). In the Testing Phase

we asked users to perform the harmonic analysis of three different chorales, organized in three different tasks. Tasks were submitted in a random order. Users were provided with printed musical scores. To evaluate the quality of the provided solution, we calculated the musical distance between such a solution and the best musical harmonization [41]. Being the chorales chosen from the J.S. Bach chorales, their best harmonization is well-known. Briefly, the musical distance between two chords is given by the sum of the distance between the tonalities of the chords and the distance between the degrees of the chords; the distance between tonalities is given by their distance in the circle of fifths; the distance between the degrees is given by the difference between the inversion numbers of the bass notes. However, the shorter the distance, the better the proposed solution. A zero value corresponds to the optimal solution.

Next, we asked users to perform a Training Phase, looking at several chorales enhanced with our visual representations. No information were given to the participants about the meaning of colors, lines, and size of objects. At the end of this phase we asked participants to repeat the harmonic analysis on the three chorales analyzed in the Testing Phase. Here, chorales were augmented with our visual representations. We provided three different solutions for each task, by including the best harmonization, too. Participants had to select what they thought was the best harmonization, trying to explain (as open-ended text) the reasons behind their choice. Finally, four further questions (5-point Likert scale with strongly agree/strongly disagree as verbal anchors) were submitted to inspect whether participants were able to understand the rules behind our approach. The main goal here was to assess the efficacy of the visual approaches to understand the harmonic analysis.

In the fourth and last phase we asked participants to express their opinion about the usefulness of the proposed approach. We were also interested in gathering perceptions about the pleasantness of the aesthetic choices made as well as perceptions about the usefulness of an instrument that automatically could show different graphical representations of a given chorale.

The preliminary (first phase) questionnaire was distributed to participants once. However, the questionnaires used in the second and the third phases were distributed to users after the testing of each approach. The overall study lasted between 30 and 35 minutes and the full test schedule took 2 weeks. The data set used in the study has been taken from J.S. Bach chorales; the questionnaires that were used are available online[2].

[2] `http://www.di.unisa.it/~delmal/research/usability/VisualHarmony/Evaluation`

5.5.2 Results

In this Section we discuss the results from each of the three test phases. First of all, recruits comprised 30 participants, with 12 of them having a conservatory degree (40%) and 19 participants with bachelor (23%) and master (37%) levels in Computer Science, Electrical Engineering, Economics, and Physics. Most of the sample was male (87%) with an mean age of 34. Table 5.1 shows that 67% of participants considered themselves "*Competent/Expert*" in the field of classical music, and 40% "*Competent/Expert*" with regard to expertise in harmonic analysis.

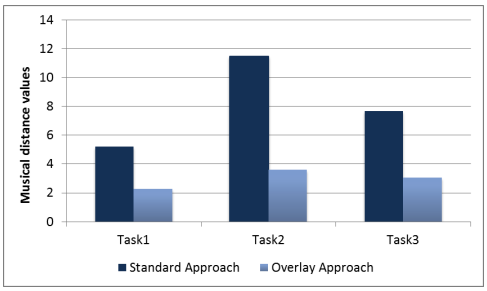
Table 5.1: Participant's Demographics

	Number	**Percentage**
Total Participants	30	
Gender		
Male	26	87%
Female	4	13%
Age		
30-35 years old	9	30%
36-40 years old	9	30%
40+ years old	12	40%
Education Level Attained		
Conservatory	12	40%
University Master/Bachelor	19	60%
Time spent playing music per week		
0-1 hour	4	13%
1-2 hours	9	30%
2+ hours	17	57%
Classical Music Expertise		
Beginner	10	33%
Competent	11	37%
Expert	9	30%
Harmonic Analysis Expertise		
Beginner	18	60%
Competent	7	23%
Expert	5	17%

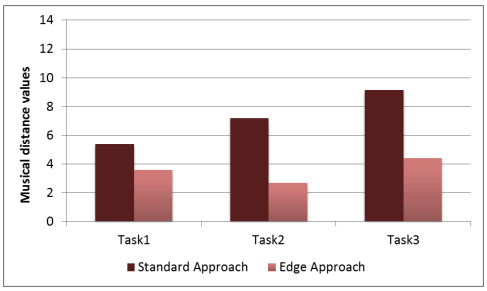
Results about the Testing Phase are shown in Fig. 5.8, where we report the results of the comparison between the Standard and our visualization approaches in terms of overall participants' performance. Specifically, in Fig. 5.8(a) we show the average distance values, calculated for all tasks, across all participants for the Overlay approach. Similarly, the same results for the Edge approach are shown in Fig. 5.8(b). As we can see, both approaches are able to improve performance for all the analyzed tasks, whereas high improvements are more evident for the Overlay approach.

Additionally, as shown in Fig. 5.9, although participants in both groups were able to correctly identify the rules behind our approaches, better results were obtained by people testing the Overlay approach.

Finally, as shown in Table 5.2, the majority of participants, in both groups, agreed with the usefulness and with the pleasantness of the proposed approaches. Most importantly, they fully agreed with the usefulness



(a) "Overlay" Approach



(b) "Edge" Approach

Figure 5.8: Comparison between the Standard and our color-based approaches in terms of overall participants' performance.

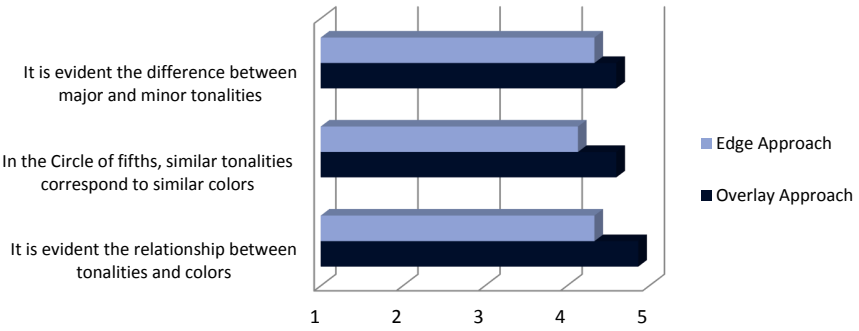


Figure 5.9: Assessment of the understanding of the rules behind the designed approaches.

of a system implementing a visual approach for understanding complex classical rules. It is worth to note that the participants belonging to the group that tested the Overlay approach, expressed more satisfaction with regard to the choices made for the graphical representation (Question Q3 in Table 5.2). We also found out a statistical difference between the two groups (p-value < 0.001). We reported the questions with the same ID used in the questionnaires that we submitted to the participants at the study and that are available online.

Table 5.2: Summary Survey. 5-point Likert scores.

		Mean		Agreement	
ID	**Question**	**Overlay**	**Edge**	**Overlay**	**Edge**
Q1	I found very useful the graphical representation provided to understand the harmonic rules of classical music	4.5	4.4	93%	93%
Q2	I found very interesting the idea of studying the rules through a visual representations of chorales	4.3	4.1	87%	93%
Q3	I found aesthetically pleasing the graphical representation used for the visualization of constructs on the musical score	4.7	3.7	100%	60%
Q4	Do you think it can be helpful to use an application that automatically offers you various graphical representation for a chorale?	4.8	4.9	100%	100%

Finally, in the form of open ended-questions, we obtained further comments and suggestions about how to improve the tested graphical representations. Specifically, the interviewed participants stated that, it could be useful:

- remove the border of the rectangles (67%, n=30)
- use a dashed background rather than the dashed border to represent degrees (63%, n = 30)
- clearly specify on the score, for each rectangle, the corresponding tonalities (58%, n = 30)
- for each degree, limit the width of the corresponding rectangle, to the width of the bass note in the score (52%, n = 15)

In summary, as result of this study, the Overlay approach was rated most effective in terms of harmonic analysis performance and most pleasant in terms of aesthetic choices. Moreover, we decided to take into account the aforementioned suggestions, since they were provided by more that half of participants. Therefore, the enhanced version of the Overlay approach appears as in Fig. 5.10.

We decided to use this enhanced version as visual representation for the tool to implement. Details will be provided in the following section.

5.6 VisualHarmony

In this section we first describe the tool implementing the visual approach evaluated as the best representation in the previous section, and next we discuss some interesting insights about how to change preferences (i.e., colors) to address the system's accessibility.

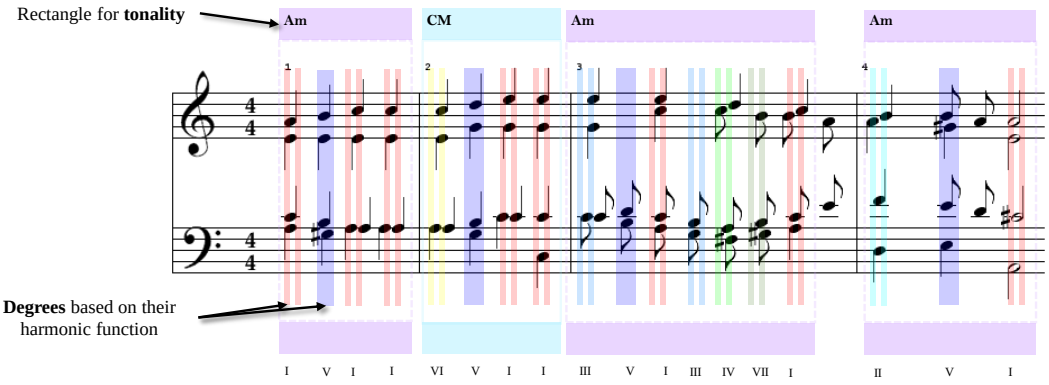


Figure 5.10: Enhancements made to the Overlay Approach to address users' comments and feedback.

5.6.1 Functionalities and Use Cases

Feedback, suggestions as well as criticisms raised during the evaluation study were addressed during the development of a visualization tool, named *VisualHarmony* (see Fig. 5.11). This system has been designed and developed to provide individuals with augmented information about the music and its background structure, in order to complement the users' experience and facilitate the study of classical compositions. The tool is also able to assist musicians during the composition of chorales, and in particular, during the definition and the analysis of the harmonic structure of the composition itself. It must be emphasized that VisualHarmony can also be used for the study of harmonic analysis of chorales already composed. The system has been implemented in Java language, using the Swing API for the development of the graphical user interface and the JFugue library [43] for Music Programming.

The functionalities provided with this system can be summarized as follows:

- **Music editing**. VisualHarmony provides a music editor for the composition of 4-voices music (basso, tenor, alto, and soprano). The tool is designed to edit most of the musical figures.

- **Harmonic visualization**. VisualHarmony allows to display a visual representation of the harmonic structure of a choral. The provided representation follows the rules defined by the Overlay approach, enhanced with changes provided as feedback by participants at that preliminary evaluation study. Users can save both the score and its corresponding harmonic visualization (in .vis format).

- **Melodic checker**. During the composition of a chorale, or during the

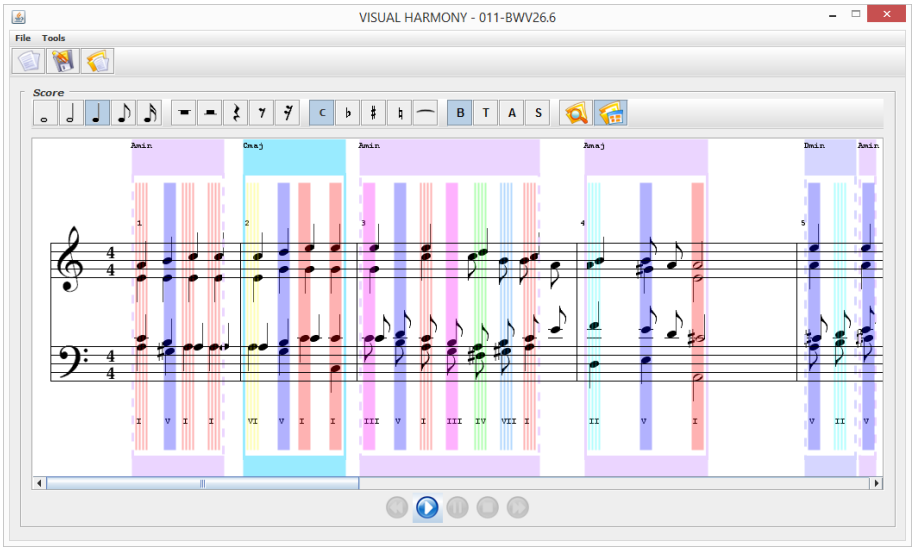


Figure 5.11: VisualHarmony Tool. The enhanced version of the Overlay approach is applied as graphical representation.

definition of the harmonic structure of the chorale, the melodic error checker can be used by musician to improve the music composition. VisualHarmony, in fact, allows to display the melodic errors, and to resolve each problem by intervening on the score through the editor.

- **Music playing.** VisualHarmony allows to play the composed music. During the configuration phase it is possible to assign a specific instrument to play for each voice.

Given the main objective of VisualHarmony to be used for learning purposes, we will describe in the following a typical usage of the tool, organized in steps, within a class of Classic Harmony in a music Conservatory.

- The teacher provides students with explanations about the theoretical concepts needed to understand the harmonic structure of musical compositions (standard learning).
- The teacher next proceeds with the explanation about the foundation rules of our visualization approach (innovative learning).
- Students absorb the theoretical concepts as well as the idea behind the visualization approach.
- The teacher provides students with a set of chorales for training purposes (that can be loaded through the tool).

- Students start with the learning process. They are encouraged to harmonize bass lines to 4 parts, step by step, with the aid of the representation provided by VisualHarmony.

Now, the teacher can assist students during the training phase, and only observe them during the learning phase. At this stage, in fact, students can leverage VisualHarmony functionalities, exploiting also the feedback provided by the Melodic checker. Specifically, the tool is able to show, for each musical piece, the correct visualization. If errors were made, students will be immediately aware of them, since the wrong visualization on the musical score. Students, only glancing at the score, can immediately understand the errors made, and therefore, re-perform the analysis, continuously querying the tool for the corresponding visualization.

The software, a user guide with a detailed description of the tool functionalities, and some examples of J.S. Bach's chorales, that could be loaded through the tool, are available online[3].

5.6.2 Circle of Fifths color customization

An interesting feedback obtained during the evaluation study was about the colors chosen for the circle of fifths. 13% of participants at the study had a color visual deficiency. To this aim, given the usage of colors to represent music constructs, in order to make the visualization accessible for all people, including people with color deficiencies we decided to allow a color customization.

We have to emphasize that the most common forms of color deficient vision, called Protanopia and Deuteranopia, are characterized by difficulties in distinguish between green and red. Colorblind is not normally a problem except in cases where the colors convey important information. In our case, since: (1) colors convey important information about how to distinguish tonality and degree, and (2) we do not want to provide an additional means of obtaining the same information as described, for example in [44, 47], we decided to change the graphical representation. Specifically, with the support of two colorblind users, we chose a different set of colors for our circle of fifths, trying to identify distinguishable colors without violate our rules described in Section 5.4.1 (similar colors that map similar information). In the left side of Fig. 5.12 we show our original circle of fifths, and how people with protanopic and deuteranopic color deficiencies perceive it[4]. As we can see from that figure, most of the selected colors are indistinguishable. In the right side of Fig. 5.12, we show how we modified colors to make them recognizable for colorblind. In Section 5.7 we describe the results of the

[3] `http://www.di.unisa.it/~delmal/research/usability/VisualHarmony/Tool`

[4] Images obtained by using the Vischeck simulator, `http://www.vischeck.com/`

usability study that we carried out to test the usability of the tool, and in particular, the satisfaction of colorblind users in terms of color vision.

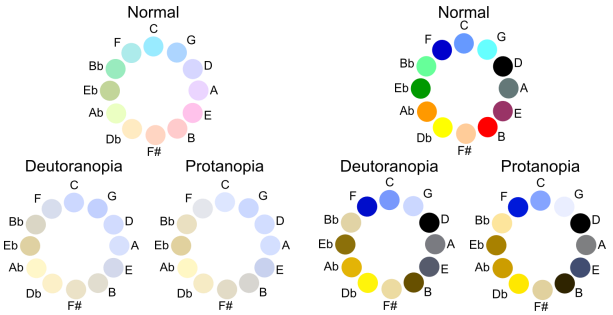


Figure 5.12: Changes made to our circle of fifths to make colors distinguishable for all people, including people with protanopic and deuteranopic deficiencies.

We provided, as default option, the configuration shown in Fig. 5.2. We allowed people to change the settings in two ways:

- selecting the configuration that we designed with the aid of two colorblind (top-right side of Fig. 5.12, Normal). It corresponds to the *Option 2* in Fig. 5.13.
- customizing the configuration according to the own needs (selecting the more visible colors for each tonality). It corresponds to the *Option Custom* in Fig. 5.13.

In Fig. 5.14 we show how the musical score appears when applying the Option 2 customization, built with the aid of two colorblind people.

5.7 Usability Study

In this Section we describe the results of the usability study conducted to test the system usability and the user satisfaction when interacting with VisualHarmony.

5.7.1 Methodology

For this study recruits comprised 11 participants among music students (60%) and music experts (with a Conservatory degree, 40%). The sample was fully male with an mean age of 38. Prior research has shown that five users is the minimum number required for usability testing, since they are able to find approximately 80% of usability problems in an interface [50, 53]. However, other research studies stated that five users are not sufficient and specifically, authors in [56] expressed that the appropriate number depends

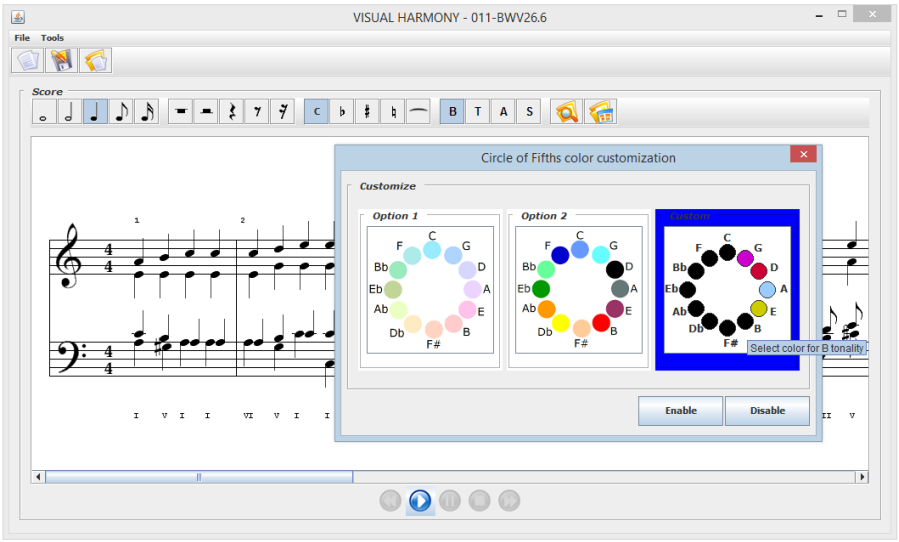


Figure 5.13: VisualHarmony Tool: how to customize colors.

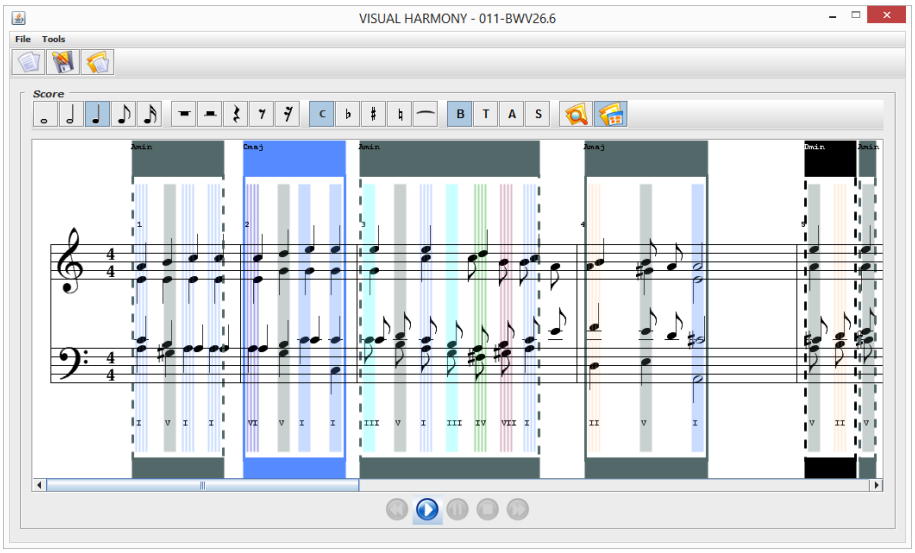


Figure 5.14: VisualHarmony Tool: score with colors changed when selecting the Option 2.

on the size of the project, with 7 users being optimal in small projects and 15 users being optimal in a medium-to-large project.

The aims of this study were: (1) derive the general audience's opinion and reaction to the software, (2) collect qualitative feedback on the visualization, and (3) inspect opinions about the behavioral intention to use the tool in the future.

Similar to the evaluation study, the usability study envisioned three dif-

ferent phases in which we carried out: a Preliminary Survey, the Tool Testing Phase, and finally, a Summary Survey, respectively. In the first phase we asked participants to fill in a preliminary questionnaire, composed of 8 questions asking for *(a)* demographic information, *(b)* information about music expertise and background, *(c)* information about ICT expertise. We also added to the questionnaire a colorblindness test, in order to verify if participants had some color deficiency, such as a colorblind deficiency. The test consisted of a colored plate, called Ishihara plate [58], which contained a circle of dots appearing randomized in color and size. Within the pattern, dots form a number clearly visible to those with normal color vision, and invisible, or difficult to see, to those with a red-green color vision defect, or the other way around. In this way, we were able to identify people with normal color vision (they will see a 42), Protanopic colorblind people (they will see a 2) or Deuteranopic colorblind people (they will see a 4).

In the tool testing phase, we asked users to use VisualHarmony for a 15-minutes session. Users were free to make a composition or to use any of the provided chorales (5 predefined chorales available online). We also asked them to freely use any of the tool's functionalities. We gave them details about VisualHarmony goals and main features. We also provided them with basic information on how to use it. Users were not directly monitored, so that they could feel free to test and explore the tool, but they could call for assistance if they did not understand any of the instructions posed. The test was performed in an isolated environment in order to avoid distractions due to the presence of other people. Users were also encouraged to provide informal feedback such as general comments, suggestions or observations for developers.

At the end of the testing phase we asked users to spend other 10 minutes to fill in the standard QUIS [59] and CSUQ [61] questionnaires. The aim was to provide additional information about system usability and user satisfaction when using VisualHarmony. Specifically, the original QUIS questionnaire was composed of 27 questions. We dropped 8 that did not seem to be appropriate to our tool (e.g., questions about task to execute). Each question was a rating on a 10-point scale with appropriate anchors at each end (e.g., "Overall Reaction to the software: Terrible/Wonderful"), where small values corresponded to unsatisfactory or negative responses and large values corresponded to satisfactory results. The original CSUQ questionnaire was composed of 19 questions. As we did for the QUIS questionnaire, we dropped 3 of them that not seem appropriate for our objectives. Specifically, we asked users to answer to the provided questions indicating their agreement or disagreement through a 7-point Likert scale with *strongly agree* and *strongly disagree* as verbal anchors.

Finally, in the third phase, we asked participants to fill in a summary questionnaire composed of 12 questions. The questions included in this questionnaire were questions asking to give a preference up to 5 possible choices

and questions on 5-point Likert scale (e.g., Strongly disagree to Strongly agree). The entire study lasted between 35 and 45 minutes. The preliminary survey, the summary survey and the QUIS and CSUQ questionnaires are publicly available[5].

5.7.2 Results

In this Section we describe the results of the study aiming at inferring users perceptions and general satisfaction about the experimented tool.

As we can see in Table 5.3, on average, all questions were rated positively. The most positive result (mean value of 8.3) was relative to the Learning metric, highlighting the easiness of the tool in terms of general usage and learning to operate it.

Table 5.3: User Satisfaction Questionnaire. 9-point Likert scale.

Metric	Mean	Dev.st
Overall reaction to the software		
Terrible/Wonderful	7.5	0.8
Difficult/Easy	8.0	1.1
Frustrating/Satisfying	7.5	1.0
Dull/Stimulating	7.5	1.1
Rigid/Flexible	7.2	1.3
Screen		
Reading Characters on the screen	7.5	1.4
Organization of information	7.2	1.7
Sequence of screens	6.8	1.4
Terminology and system information		
Use of terms throughout the system	7.5	0.7
Terminology is intuitive	7.5	1.0
Position of messages on the screen	6.8	1.3
Prompts for input	7.6	0.8
Error messages	6.6	1.1
Learning		
Learning to operate the system	8.3	1.3
Performing tasks is straightforward	8.3	0.9
System capabilities		
System speed	8.0	0.9
System reliability	7.9	0.9
System tends to be	7.3	1.3
Designed for all levels of users	7.8	1.3

Items in CSUQ relate to efficiency, ease of use, likability of the system interface, overall satisfaction. Specifically, we computed five factor scores: System Efficacy, Usefulness, Satisfaction, Easy of Use, and Easy of Learning. Similar to the QUIS, as we can see from Fig. 5.15, all questions were positively evaluated, especially for the Easy of Use, Easy of Learning, and Satisfaction metrics. The most positive answers (on average 6.6), in fact,

[5] `http://www.di.unisa.it/~delmal/research/usability/VisualHarmony/Usability`

were about the question 1 ("*It was simple to use this system*") and question 7 ("*It was easy to learn to use this system*") in the questionnaire available online.

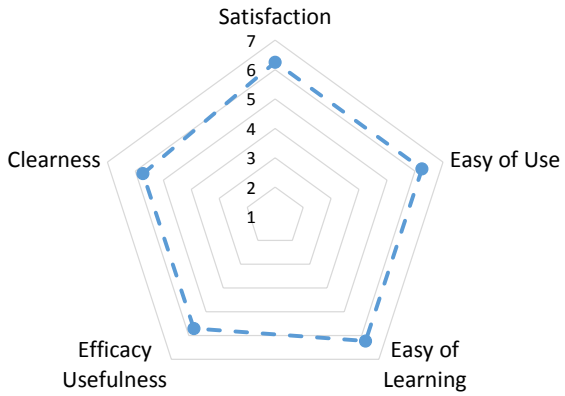


Figure 5.15: CSUQ results organized according to five metrics: Satisfaction, Easy of Use and Learning, Efficacy/Usefulness, and Clearness.

In Tables 5.4 and 5.5 we can see the results of the summary survey. Specifically, in Table 5.4 we can see that the reaction to the software was strongly positive, and that the sample was fully agree with willingness to use the tool in the future.

Table 5.4: Summary Survey. 5-point Likert scores. N=11.

ID	Question	Mean	Dev.st
Q1	Overall, I found easy to use the proposed system	4.4	0.5
Q2	In general, the proposed system was very interesting	4.5	0.5
Q3	In general, the proposed system was very useful	4.3	0.6
Q4	In general, I find useful the functionalities offered to users who have visual impairments	4.4	0.7
Q9	Would you consider the possibility to continue to use this system?	4.3	0.5
Q10	Would you recommend it to a friend / colleague?	4.4	0.5

In Table 5.5 we show the results of the questions answered only by participants with colorblind deficiency (n = 3). They rated very useful the support for colorblind and very easy the task of configuring colors to make them distinguishable. It is worth to note that the lowest result (even if above the mean value) was about the question Q6, i.e., *"I found effective the options provided (I was able to distinguish colors on the musical score"*. One out of three participants, a red colorblind, had difficulties in distinguishing two colors in the circle of fifths. A possible explanation is that colors were assigned by taking into account the suggestions of only two colorblind people, probably with a different color's anomaly. A larger number of individuals could be useful to find colors more distinguishable.

Table 5.5: Summary Survey. 5-point Likert scores. Answers by colorblind. N=3.

ID	Question	Mean	Dev.st
Q5	I found useful the functionalities offered to make accessible the system for colorblind users	4.0	0.0
Q6	I found effective the options provided (I was able to distinguish colors on the musical score	3.7	0.6
Q7	I found easy to select the Colorblind options	4.0	1.0
Q8	I found easy to configure the Custom option, which allows to select the most suitable colors for me	4.3	1.2

In summary, the result of this study is that our participants found very useful and easy to use the experimented tool. Their overall reaction to the software was very positive, and they expressed high satisfaction and their willingness to continue to use the system in the future.

5.8 Conclusion

The ability to understand the harmonic structure of a musical composition is important in every genre of music, from classical to pop music. In particular, the harmonic analysis is an important step in many techniques and problems of classical musical composition. Furthermore, the study of the harmonic analysis of classical compositions is considered, in this field, as a time-consuming and tedious task, given the need to understand and remember complex music rules. In this work we defined and compared two graphical approaches that exploit visualization techniques to unveil the structure of 4-voices music composition with the aim of making easier, quick, and intuitive the study of classical notations. Musicians, and general audience can be provided with a mechanism able to clarify complex relationships in music using visual clues. A preliminary evaluation study allowed us to assess which of these two approaches was most useful in assisting individuals during music learning activities, and specifically, for the study of the harmonic analysis techniques. The Overlay approach with some changes suggested by participants at the study, was rated most useful in terms of both participants performance and aesthetic choices. Results of this study have been translated in a software implementation, named VisualHarmony, whose main goal is to assist users during learning activities as well as during composition activities.

VisualHarmony was finally tested in order to analyze system usability and user satisfaction. The results of these studies provided us with positive feedback about the effectiveness of the idea, the pleasantness of the graphical choices, the satisfaction of the users with regard to the easiness of the tool and the willingness of participants to advertise it and to continue to use it in the future.

As future work we will investigate the design and the implementation

of a visual representation for the melodic structure of a music composition. In addition to the rules about chord sequences, there exist rules about melodic lines. In music, several rules concern the melodic voices and are aimed at avoiding voice errors. Rules about melodic lines can refer to the movement of a single voice (for example, normally a jump bigger than of an octave is not allowed; transitions within the notes of the musical scale are preferred respect to transitions to notes that are not part of the scale) or also to the movements of two voices (for example, two voices that proceed by parallel fifth are not allowed). The idea is to integrate the representation for the harmonic structure of a composition, that we presented in this work, with a graphical visualization of the melody, in order to improve the ability of the musicians to compose music. Finally, we are planning an extensive and representative experimental study involving a large sample of students, from Conservatory classes, mainly interested in learning complex music constructs, and musicians mainly interested in music composition. A larger number of subjects would also provide more statistically significant results.

Bibliography

[1] H.A. Abbass, "The Self-Adaptive Pareto Differential Evolution Algorithm," in Proceedings of the *IEEE Congress on Evolutionary Computation (CEC 2002)*, vol. 1, pp. 831-836, Piscataway, New Jersey, May 2002.

[2] Jum C. Nunnally, Ira H. Bernstein, "Psychometric theory," McGraw-Hill, 1994.

[3] H.A. Abbass, and Ruhul Sarker, "The Pareto Differential Evolution Algorithm," *International Journal on Artificial Intelligence Tools*, 11(4):531-552, 2002.

[4] Head, T., "Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviours," Bulletin of Mathematical Biology, pp. 737-759, 1987.

[5] Davis, F. D. and Bagozzi, R. P. and Warshaw, P. R., "Extrinsic and intrinsic motivation to use computers in the workplace," Journal of Applied Social Psychology, vol. 22, num. 14, pp. 1111-1132, 1992.

[6] Nayak, Laxman U. S. and Priest, Lee and White, Allan P., "An application of the technology acceptance model to the level of Internet usage by older adults," Universal Access in the Information Society, vol. 9, num. 4, pp. 367-374, 2010.

[7] Cronbach, Lee J., "Coefficient alpha and the internal structure of tests," Psychometrika, vol. 16, num. 3, pp. 297-334, 1951.

[8] De Haas, W and Magalhães, J and Wiering, F and C.Veltkamp, R, "Automatic Functional Harmonic Analysis," Computer Music Journal, vol. 37, num. 4, pp.37-53, 2013.

[9] Peter Auer and Harald Burgsteiner and Wolfgang Maass, "A learning rule for very simple universal approximators consisting of a single layer of perceptrons," Neural Networks, vol. 21, num. 5, pp. 786-795, 2008.

[10] Roberto De Prisco, Delfina Malandrino, Gianluca Zaccagnino, Rocco Zaccagnino, "An Evolutionary Composer for Real-Time Background Music," EvoMUSART, pp. 135-151, 2016.

[11] Russell, J.A., "A circumplex model of affect," Journal of personality and social psychology, vol. 39, pp. 1161-1178, 1980.

[12] Walter Piston, "Harmony: Fifth Edition," W. W. Norton & Company, Inc., 1987.

[13] Christopher Czarnecki, "J.S. Bach 413 Chorales: Analyzed. A study of the Harmony of J. S. Bach," SeeZar Publications, 2013.

[14] De Prisco, Roberto and Zaccagnino, Rocco, "An Evolutionary Music Composer Algorithm for Bass Harmonization," Applications of Evolutionary Computing, vol. 5484, pp. 567-572, 2009.

[15] De Prisco, Roberto and Zaccagnino, Gianluca and Zaccagnino, Rocco, "EvoBassComposer: A Multi-objective Genetic Algorithm for 4-voice Compositions," in Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, pp. 817-818, 2010.

[16] De Prisco, R. and Zaccagnino, G. and Zaccagnino, R., "A multi-objective differential evolution algorithm for 4-voice compositions," 2011 IEEE Symposium on Differential Evolution (SDE), pp. 1-8, 2011.

[17] Pedro Krger and Re Passos and Marcos Sampaio and Givaldo De Cidra, "Rameau: A system for automatic harmonic analysis," in Proceedings of the 2008 International Computer Music Conference, pp. 273-281, 2008.

[18] , De Haas, W and Magalhães, J and Wiering, F and C.Veltkamp, R, "Automatic Functional Harmonic Analysis," Computer Music Journal, vol. 37, num. 4, 2013.

[19] M. Wattenberg, "The shape of song," http://www.turbulence.org/Works/song

[20] Wattenberg, M., "Arc diagrams: visualizing structure in strings", IEEE Symposium on Information Visualization, pp. 110-116, 2002.

[21] Bergstrom, Tony and Karahalios, Karrie and Hart, John C., "Isochords: Visualizing Structure in Music", in Proceedings of Graphics Interface, pp. 297-304, 2007.

[22] Roberto De Prisco, Delfina Malandrino, Gianluca Zaccagnino, Rocco Zaccagnino, "Natural Users Interfaces to support and enhance Real-Time Music Performance", AVI, 2016.

[23] Clelia De Felice, Roberto De Prisco, Delfina Malandrino, Gianluca Zaccagnino, Rocco Zaccagnino, Rosalba Zizza, "Splicing Music Composition", Information Sciences Journal, 385: 196 ? 215 (2017).

[24] Clelia De Felice, Roberto De Prisco, Delfina Malandrino, Gianluca Zaccagnino, Rocco Zaccagnino, Rosalba Zizza, "Chorale Music Splicing System: An Algorithmic Music Composer Inspired by Molecular Splicing", EvoMusart, pp.50?61, 2015.

[25] Roberto De Prisco, Delfina Malandrino, Donato Pirozzi, Gianluca Zaccagnino, Rocco Zaccagnino, "Understanding the structure of music compositions: is visualization an effective approach?", Information Visualization Journal, 2016.

[26] ? Delfina Malandrino, Donato Pirozzi, Gianluca Zaccagnino, Rocco Zaccagnino, "A Color-Based Visualization Approach to Understand Harmonic Structures of Musical Compositions", IV 2015: 56-61.

[27] Smith, SM. and Williams, G., "A visualization of music," in Proceedings of the 8^{th} Conference on Visualization, pp. 499-503, 1997.

[28] Regenbrecht, Holger and Collins, Jonny M. and Hoermann, Simon, in Proceedings of the OZCHI 2013, 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration, 2013.

[29] Miyazaki, Reiko and Fujishiro, Issei and Hiraga, Rumi, "Exploring MIDI Datasets," ACM SIGGRAPH 2003 Sketches &Amp; Applications, 2003.

[30] Joyce Horn Fonteles and Maria Andria Formico Rodrigues and Victor Emanuel, "Creating and evaluating a particle system for music visualization", journal = "Journal of Visual Languages & Computing," vol. 24, pp. 472-482, 2013.

[31] Stephen Malinowki, "The Music Animation Machine "Music Worth Watching"," http://www.musanim.com/

[32] Ciuha, Peter and Klemenc, Bojan and Solina, Franc, "Visualization of Concurrent Tones in Music with Colours," in Proceedings of the International Conference on Multimedia, pp. 1677-1680, 2010.

[33] Bock, Thomas and Georgoulas, C. and Gttler, J. and Linner, Thomas, "Intuitive and Adaptive Robotic Arm Manipulation using the Leap Motion Controller," Proceedings of the joint conference of the 45th International Symposium on Robotics (ISR 2014) and the 8th German Conference on Robotics (ROBOTIK 2014), 2014.

[34] Sapp, Craig Stuart, "Visual Hierarchical Key Analysis", Comput. Entertain. vol. 3, num. 4, pp. 1-19, 2005.

[35] Sharad Vikram and Lei Li and Stuart Russell, "Handwriting and Gestures in the Air, Recognizing on the Fly," in Proceedings of the CHI 2013 Extended Abstracts, 2013.

[36] Bishop, Christopher M., "Pattern Recognition and Machine Learning (Information Science and Statistics,", 2006.

[37] Mardirossian, Arpi and Chew, Elaine, "Visualizing Music: Tonal Progressions and Distributions," in Proceedings of the 8^{th} International Conference on Music Information Retrieval, pp. 189-194, 2007.

[38] Jonathan Lazar, Jinjuan Heidi Feng, Harry Hochheiser, "Research Methods In Human-Computer Interaction," Wiley, 2010.

[39] Malandrino, Delfina and Scarano, Vittorio and Spinelli, Raffaele, "How Increased Awareness Can Impact Attitudes and Behaviors Toward Online Privacy Protection," International Conference on Social Computing, pp. 57.62, 2013.

[40] Malandrino, Delfina and Manno Ilaria and Palmieri, Giuseppina and Scarano, Vittorio and Filatrella, Giovanni, "How Quiz-based Tools can improve students' engagement and participation in the classroom," International Conference on Collaboration Technologies and Systems, pp. 379-386, 2014.

[41] De Prisco, Roberto and Eletto, Antonio and Torre, Antonio and Zaccagnino, Rocco, "A Neural Network for Bass Functional Harmonization," Applications of Evolutionary Computation, vol. 6025, pp. 351-360, 2010.

[42] Chang, Chih-Chung and Lin, Chih-Jen, "LIBSVM: A Library for Support Vector Machines," ACM Trans. Intell. Syst. Technol., vol. 2, num. 3, pp. 1-27, 2011.

[43] David Koelle, "Music Programming for Java and JVM Languages," http://www.jfugue.org/.

[44] Iaccarino, Gennaro and Malandrino, Delfina and Scarano, Vittorio, "Personalizable Edge Services for Web Accessibility," in Proceedings of the 2006 International Cross-disciplinary Workshop on Web Accessibility (W4A): Building the Mobile Web: Rediscovering Accessibility?, pp. 23-32, 2006.

[45] Davis, Fred D., "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology," MIS Q., vol. 3, num. 3, pp. 319-340, 1989.

[46] Martin Fishbein and Icek Ajzen, "Belief, Attitude, Intention, and Behavior: An Introduction to Theory and Research,", 1975.

[47] Ugo Erra and Gennaro Iaccarino and Delfina Malandrino and Vittorio Scarano, "Personalizable edge services for Web accessibility," Universal Access in the Information Society, vol. 3, num. 3, pp. 285-306, 2007.

[48] "Extending the {TAM} for a World-Wide-Web context," Information and Management, vol. 38, num. 4, pp. 217-230, 2001.

[49] Silva, Eduardo S. and de Abreu, Jader Anderson O and de Almeida, Janiel Henrique P and Teichrieb, Veronica and Ramalho, Geber L, "A preliminary evaluation of the leap motion sensor as controller of new digital musical instruments," 2013.

[50] Virzi, Robert A., "Refining the Test Phase of Usability Evaluation: How Many Subjects is Enough?," Hum. Factors, vol.34, num. 4, pp. 457-468 1992.

[51] Gordon C. Bruner II and Anand Kumar, "Explaining consumer acceptance of handheld Internet devices," Journal of Business Research, vol. 58, num. 5, pp.553-558, 2000.

[52] Kwok,Irene and Lee, Charlene and Okerlund, Johanna and Zhu, Qiuyu and Shaer, Orit, "musicAir Creating Music Through Movement," ACM, 2014.

[53] James R. Lewis, "Legitimate Use of Small Samples in Usability Studies: Three Examples," IBM Human Factors, 1991.

[54] Păun, G., "On the splicing operation," Discrete Applied Mathematics, vol. 70, pp. 57-79, 1996.

[55] Pixton, D., "Regularity of splicing languages.", Discrete Applied Mathematics, vol. 69, pp. 101-124, 1996.

[56] Nielsen, Jakob and Landauer, Thomas K., "A Mathematical Model of the Finding of Usability Problems," in Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems, pp. 206-213, 1993.

[57] Boenn, G. and Brain, M. and De Vos, M. and ffitch, J., "Automatic composition of melodic and harmonic music by answer set programming", ICLP, pp. 160-174, 2008.

[58] S. Ishihara, "Tests for Color-blindness,"

[59] Chin, John P. and Diehl, Virginia A. and Norman, Kent L., "Development of an Instrument Measuring User Satisfaction of the Human-computer Interface," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 213-218, 1988.

[60] H.A. Abbass, R. Sarker, and C. Newton, "PDE: A Paretofrontier differential evolution approach for multi-objective optimization problems," in Proceedings of the *IEEE Congress on Evolutionary Computation (CEC 2001)*, pp. 971-978, 2001.

[61] Lewis, James R., "IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use," Int. J. Hum.-Comput. Interact., vol. 7, num.1, pp. 57-78, 1995.

[62] T, Back, "Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms," *Oxford University Press US*, January 1996, ISBN: 0-1950-9971-0, 978-0-19509-971-3.

[63] , Malloch, Joseph and Wanderley, Marcelo M, "The T-Stick: From Musical Interface to Musical Instrument," in proceedings of the 7th International Conference on New Interfaces for Musical Expression, ACM, pp. 66-70, 2007.

[64] Head, T. and Păun, G. and Pixton, D., "Language theory and molecular genetics: generative mechanisms suggested by DNA recombination", Handbook of Formal Languages, vol. 2, pp. 295-360, 1996.

[65] Splicing systems, "Zizza, R.", Scholarpedia, vol. 5, num. 7, 2010.

[66] Bonizzoni, "The structure of reflexive regular splicing languages via Schützenberger constants," Theoretical Computer Science, vol. 334, pp. 71-98, 2005.

[67] M. Buckwald, D. Holz "The Leap Motion 3D Controller," https://www.leapmotion.com/, 2010.

[68] T. Back, D. B. Fogel, Z. Michalewicz, editors, Handbook of Evolutionary Computation, Computational Intelligence Library, *Oxford University Press* in cooperation with the *Institute of Physics Publishing / CRC Press*, Bristol, New York, ringbound edition, April 1997, ISBN: 0-7503-0392-1, 978-0-75030-392-7, 0-7503-0895-8, 978-0-75030-895-3.

[69] T. Back, U. Hammel, and H. Schwefel, "Evolutionary computation: comments on the history and current state," in Proceedings of *IEEE Transactions on Evolutionary Computation*, 1(1), pp.317, April 1997.

[70] B. Bel, J. Kippen, "Bol Processor Grammars," Understanding music with AI. *AAAI Press/MIT*, Cambridge, 1992.

[71] MI. Bellgrad, CP. Tsang , "Harmonizing music the Boltzmann way," Connection Science, 6, 1994.

[72] J. A. Biles, "GenJam: A genetic algorithm for generating jazz solos," in Proceedings of the *International Computer Music Conference*, pp.131-137, 1994.

[73] D. Bradshaw, K. Ng, "Tracking Conductors Hand Movements Using Multiple Wiimotes," in Proceedings of the *International Conference on Automated Solutions for Cross Media Content and Multi-channel Distribution* (AXMEDIS 2008), Florence, Italy, pp. 93–99, Digital Object Identifier 10.1109/AXMEDIS.2008.40, IEEE Computer Society Press, ISBN: 978-0- 7695-3406-0. 4. 17-19 Nov. 2008.

[74] J. Branke, "Evolutionary approaches to dynamic optimization problems - introduction and recent trends," in Proceedings of *GECCO workshop on evolutionary algorithms for dynamic optimization problems*, pp.2-4, 2003.

[75] R. S. Brindle, "Serial Composition," *Oxford University Press*, London, 1966.

[76] B. Bruegge, C. Teschner, P. Lachenmaier, E. Fenzl, D. Schmidt, S. BierbaumBruegge, "Pinocchio: Conducting a Virtual Symphony Orchestra," in Proceedings of the *International Conference on Advances in Computer Entertainment Technology* (ACE), Salzburg, Austria, pp. 294-295, 2007

[77] Cambridge Advanced Learners Dictionary, *http://dictionary.cambridge.org/define.asp?key=2032&dict=CALD*, Cited 17 Jan 2006.

[78] P. Casella, P. Paiva, "Magenta: An Architecture for Real Time Automatic Composiiton of Background Music," in Proceedings of *International Workshop on Intelligent Virtual Agents*, pp. 224-232, Springer, 2001.

[79] Cattermole, Tannith, "Farseeing inventor pioneered computer music", Gizmag. Retrieved 28 October 2011. "In 1957 the MUSIC program allowed an IBM 704 mainframe computer to play a 17-second composition by Mathews. Back then computers were ponderous, so synthesis would take an hour."(May 9, 2011)

[80] N. Chomsky, "Syntactic structures," Mouton, Den Haag, Reprinted byWalter de Gruyter, Berlin, New York, 1989. ISBN 9027933855

[81] N. Chomsky, "Aspects of the theory of syntax," *MIT Press*, Cambridge, Mass, 1965.

[82] J. Chung, G. Vercoe, "The Affective Remixer: Personalized Music Arranging," in Proceedings of *Conference on Human Factors in Computing Systems*, pp. 393-398, 2006, *ACM Press*, New York.

[83] H. G. Cobb, "An Investigation into use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments," Technical Report AIC-90-001, Naval Research Laboratory, Washington, USA.

[84] D. Cope, *Experiments in Musical Intelligence*. A-R Editions, 1996. ISBN 0895793377.

[85] D. Cope, *The Algorithmic Composer*. A-R Editions, 2000. ISBN 0895794543.

[86] D. Cope, *Virtual Music*. The MIT Press, 2004. ISBN 0262532611.

[87] D. Cope, web page *http://artsites.ucsc.edu/faculty/cope/*.

[88] RB. Dannenberg, B. Thom, D. Watson, "A machine learning approach to musical style recognition," in Proceedings of the *International Computer Music Conference*, International Computer Music Association, 1997, San Francisco.

[89] C. Darwin, "On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life", 1859.

[90] K. De Jong, "An analysis of the behavior of a class of genetic adaptive systems", PhD thesis, University of Michigan, Ann Arbor MI, 1975.

[91] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, 6(2), pp.182-197, April 2002.

[92] R. De Prisco, R. Zaccagnino, "An Evolutionary Music Composer Algorithm for Bass Harmonization," In Proceedings of the *EvoWorkshops 2009 on Application of Evolutionary Computing*, pp.567–572, 2009.

[93] R. De Prisco, G. Zaccagnino, and R. Zaccagnino, "A multi-objective differential evolution algorithm for 4-voice compositions," In Proceedings of the *IEEE Symposium on Differential Evolution - SDE , 2011.*

[94] Krosnick, J.A. ands Petty, R.E., "Attitude Strength: Antecedents and Consequences," R.E. Petty and J.A. Krosnick, 1995, Lawrence Erlbaum Associates.

[95] Felipe Bacim and Mahdi Nabiyouni and Doug A. Bowman, "Slice-n-Swipe: A Free-Hand Gesture User Interface for 3D Point Cloud Annotation," IEEE Symposium on 3D User Interfaces (3DUI), 2014.

[96] R. De Prisco, A. Eletto, A. Torre, R. Zaccagnino, "A Neural Network for Bass Functional Harmonization," In Proceedings of the *EvoApplications 2010*, pp.351–360.

[97] D. P. Radicioni and R. Esposito. Advances in Music Information Retrieval, "BREVE: an HMPerceptron-Based Chord Recognition System. Studies in Computational Intelligence," Zbigniew W. Ras and Alicja Wieczorkowska (Editors), Springer, 2010.

[98] R. De Prisco, G. Zaccagnino, R. Zaccagnino, "A Genetic Algorithm for Dodecaphonic Compositions," In Proceedings of the *EvoApplications 2011*, pp.244–253.

[99] R. De Prisco, P. Sabatino, G. Zaccagnino, R. Zaccagnino, "A Customizable Recognizer for Orchestral Conducting Gestures Based on Neural Networks," In Proceedings of the *EvoApplications 2011*, pp. 254–263.

[100] A. K. Dewdney, "Cellular universe of debris, droplets, defects and demons", Scientific American, August, 1989.

[101] M. Downie, "Behavior animation, music: the music and movement of synthetic characters," Master's thesis, MIT, 2011. Preliminary version.

[102] D. Eck, J. Schmidhuber, "A first look at music composition using LSTM recurrent neural networks," Technical Report IDSIA-07-02, 2002.

[103] K. Ebcioglu. An expert system for harmonizing four-part chorales. In *Machine models of music*, pp- 385–401, MIT Press, 1992.

[104] H. Eimert. Lehrbuch der Zwölftontechnik. Wiesbaden. Breitkopf & Härtel, 1950.

[105] Encyclopedia Britannica Online, *http://cache.britannica.com/eb/article-9005707*, Cited 17 Jan 2006.

[106] C. Fonseca, M. Fleming, "J. Genetic Algorithms for Multiobjetive Optimization: Formulation, Discussion and Generalization," in Proceedings of the *Fifth International Conference on Genetic Algorithms*, pp. 416–423, 1959).

[107] A. Gabrielsson, E. Lindstrom, "The Influence of Musical Structure on Emotional Expression," Music and Emotion: Theory and Research, pp. 223-248, *Oxford University Press*, 2001.

[108] A, Gartland-Jones, "Can a genetic algorithm think like a composer?," In Proceedings of *5th International Conference on Generative Art*, Politecnico di Milano University, Milan, 2002.

[109] A. Gartland-Jones, "MusicBlox: a real-time algorithmic composition system incorporating a distributed interactive genetic algorithm," In Raidl G et al (eds) Applications of evolutionary computing, Lecture notes in computer science, vol 2611, Springer, Berlin, pp 490501.

[110] D. E. Goldberg, R. E. Smith, "Nonstationary function optimization using genetic algorithms with dominance and diploidy," in J. J. Grefenstette, editor, International Conference on Genetic Algorithms, pp. 59-68, Lawrence Erlbaum Associates.

[111] J. J. Grefenstette, "Genetic algorithms for changing environments," in R. Maenner and B. Manderick, editors, Parallel Problem Solving from Nature 2, pp. 137144. North Holland, 1992.

[112] H. Hild, J. Feulner, W. Menzel. Harmonet, "A neural net for harmonizing chorales in the style of J.S. Bach," In Advances in Neural Information Processing Systems 4, pp. 267–274. San Mateo, CA, Morgan Kaufmann, 1992.

[113] L. Hiller, Computer music. *Scientific American*, n. 201 (6), pp. 109–120, 1959).

[114] L. Hiller, L.M. Isaacson. *Experimental music*. McGraw-Hill, New York, 1959.

[115] G.E. Hinton , D.E. Rumelhart, R. Williams, "Learning internal representations by error propagation," in Rumelhart D, McClelland JL (eds) (1986) Parallel distributed processing: Explorations in the microstructure of cognition, 1: Foundations, MIT Press, Cambridge, Mass, 1986.

[116] G.E. Hinton , D.E. Rumelhart, R. Williams, "Learning internal representations by backpropagating errors," Nature, 323, 1986.

[117] F. G. Hofmann, P. Heyer, G. Hommel, "Velocity profile based recognition of dynamic gestures with discrete hidden markov models," in Proceedings of the *International Gesture Workshop on Gesture and Sign Language in Human-Computer Interaction*, pp. 8195, London, UK, 1998. Springer-Verlag.

[118] J. H. Holland, *Adaption in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[119] D. Horowitz, "Generating rhythms with genetic algorithms," In Proceedings of the *1994 International Computer Music Conference*, International Computer Music Association, San Francisco.

[120] J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in Proceedings of the *National Academy of Sciences of the United States of America*, 79, 1982.

[121] Wallace Lages and Mahdi Nabiyouni and Javier Tibau and Doug A. Bowman, "Interval Player: Designing a Virtual Musical Instrument Using In-Air Gestures," IEEE Symposium on 3D User Interfaces (3DUI Contest), 2015.

[122] Legaspi, Roberto and Hashimoto, Yuya and Moriyama, Koichi and Kurihara, Satoshi and Numao, Masayuki, "Music Compositional Intelligence with an Affective Flavor,", pp. 216-224, 2007.

[123] , Unehara, Muneyuki and Onisawa, Takehisa, "Music composition system based on subjective evaluation,", SMC, pp. 980-986, 2003.

[124] J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," in Proceedings of the *National Academy of Sciences of the United States of America*, 81, 1984.

[125] A. Horner, L. Ayers, "Harmonization of musical progression with genetic algorithms," in Proceedings of the *International Computer Music Conference*, pp.483–484, 1995.

[126] Shiratuddin, MohdFairuz and Wong, KokWai, "Game Design Considerations When Using Non-touch Based Natural User Interface," Transactions on Edutainment VIII, vol. 7220, pp. 35-45.

[127] A. Horner, D. E. Goldberg, "Genetic algorithms and computer assisted music composition," Technical report, University of Illinois, 1991.

[128] T. Ilmonen, "Tracking Conductor of an Orchestra Using Artificial Neural Networks," Master's thesis. Telecommunications Software and Multimedia Laboratory. Helsinki, University of Technology, 1999.

[129] Michael, David R. and Chen, Sandra L. "Serious Games: Games That Educate, Train, and Inform," Muska & Lipman/Premier-Trade, 2005.

[130] Chien-Yen Chang and Lange, B. and Mi Zhang and Koenig, S. and Requejo, P. and Noom Somboon and Sawchuk, A.A. and Rizzo, A.A., "Towards pervasive physical rehabilitation using Microsoft Kinect,", pp. 159-162, 2012.

[131] A. W. Iorio, and X. Li, "Solving rotated multi-objective optimization problems using differential evolution," in Proceedings of *Advances in Artificial Intelligence (AI 2004)*, pp. 861-872, Springer- Verlag, Lecture Notes in Artificial Intelligence vol. 3339, 2004.

[132] B. L. Jacob, "Composing with genetic algorithms," Technical report, University of Michigan, 1995.

[133] J.R. Jang, "ANFIS: An Adaptive-Nework-Based Fuzzy Inference System," in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, NO. 3, May,June 1993.

[134] Bassily, D. and Georgoulas, C. and Guettler, J. and Linner, T. and Bock, T., "Intuitive and Adaptive Robotic Arm Manipulation using the Leap Motion Controller," International Symposium on Robotics, pp. 1-7, 2014.

[135] Carvalho Correia, AnaCarla and de Miranda, LeonardoCunha and Hornung, Heiko, "Gesture-Based Interaction in Domotic Environments: State of the Art and HCI Framework Inspired by the Diversity," Human-Computer Interaction, pp.300-317, 2013.

[136] A. Kandel, "Fuzzy expert systems," *Addison-Wesley*, 1988.

[137] A. Kandel, "Fuzzy expert systems," *CRC Press*, Boca Raton, FL, 1992.

[138] Nancel, Mathieu and Wagner, Julie and Pietriga, Emmanuel and Chapuis, Olivier and Mackay, Wendy, "Mid-air Pan-and-zoom on Wall-sized Displays," CHI '11, pp. 177–186, 2011.

[139] Shiratuddin, MohdFairuz and Wong, KokWai, "Game Design Considerations When Using Non-touch Based Natural User Interface," Transactions on Edutainment VIII, vol. 7220, pp. 35-45, 2012.

[140] C. Kiefer, N. Collins, G. Fitzpatrick, "Evaluating the Wiimote as a Musical Controller," in Proceedings of the *2008 International Computer Music Conference* (ICMC), 2008.

[141] F. Kiernan, "Score-based style recognition using artificial neural networks," in Proceedings of the *first International Conference on Music Information Retrieval (ISMIR 2000)*, Plymouth, Mass.

[142] Jego, J.-F. and Paljic, A. and Fuchs, P., "User-defined gestural interaction: A study on gesture memorization," IEEE Symposium on 3D User Interfaces (3DUI), pp. 7-10, 2013.

[143] Tormoen, Daniel and Thalmann, Florian and Mazzola, Guerino, "The Composing Hand: Musical Creation with Leap Motion and the BigBang Rubette," NIME, pp. 207-212.

[144] Milmeister,G., "The Rubato Composer Music Software:Component-Based Implementation of a Functorial Concept Architecture," Springe Berlin, 2009.

[145] S. Kim, S. André, "Composing Affective Music with a Generate and Sense Approach," Flairs, Special Track on AI and Music, *AAAI Press*, 2004

[146] P. Kolesnik, "Conducting Gesture Recognition, Analysis and Performance System," Thesis.

[147] C. C. Lee, "Fuzzy logic in control systems: fuzzy logic controller-part 1-2," in Proceedings of the *IEEE Transactions on Systems, Man, and Cybernetics*, vol 20(2),pp. 404-435, 1990.

[148] J. C. Lee, "Hacking the Nintendo Wii Remote," volume 7, Piscataway, NJ, USA, 2008. IEEE Educational Activities Department.

[149] D. Lehmann, D. Gang and N. Wagner, "Tuning neural network for harmonizing melodies in real-time," In *International Computer Music Conference*, Ann-Arbor, Michigan, 1998.

[150] D. Lehmann, "Harmonizing melodies in real time: the connectionist approach," in Proceedings of the *International Computer Music Conference* Thessaloniki, 1997.

[151] F. Lerdahl, R. Jackendoff, "A generative theory of tonal music," MIT Press, Cambridge, Mass, ISBN 0-262-62107-X., 1993.

[152] J. Lewis, E. Hart, and G. Ritchie, "A comparison of dominance mechanisms and simple mutation on non-stationary problems,", In E. Eiben, T. Back, M. Schoenauer, and H.-P. Schwefel editors, pp. 139148.

[153] H. Liang, M. Ouhyoung, "A Real-Time Continuous Gesture Recognition System for Sign Language," in *Third IEEE International Conference on Automatic Face and Gesture Recognition* (FG'98),pp.558, 1998.

[154] S. R. Livingstone, R. Muhlberger, A. R. Brown, A. Loch, "Controlling Musical Emotionality: An Affective Computational Architecture for Influencing Musical Emotions", Digital Creativity, 18, pp. 43-53, Taylor and Francis, 2007.

[155] A.R. López, A. P. Oliveira, A. Cardoso, "Real-Time Emotion-Driven Music Engine," in Proceedings of *International Conference on Computational Creativity*, 2010.

[156] S. J. Louis, Z. Xu, "Genetic algorithms for open shop scheduling and re-scheduling,", In M. E. Cohen and D. L. Hudson, editors, ISCA *Eleventh International Conference on Computers and their Applications*, pp. 99102, 1996.

[157] G. Loy, *Musimathics, Vol. 1.* The MIT Press, 2006. ISBN 0262122820.

[158] N.K. Madavan, "Multiobjective Optimization Using a Pareto Differential Evolution Approach," in Proceedings of *Congress on Evolutionary Computation (CEC 2002)*, vol. 2, pp. 1145-1150, Piscataway, New Jersey, May 2002.

[159] Y. Maeda, Y. Kajihara, "Automatic Generation Method of Twelve Tone Row for Musical Composition Using Genetic Algorithm," in Proceedings of the *IEEE 18th International conference on Fuzzy Systems FUZZY2009*, pp. 963-968, 2009.

[160] M. Mathews (1 November 1963), "The Digital Computer as a Musical Instrument,", Science 142 (3592), pp. 553557.

[161] E. Mayr, "The Growth of Biological Thought," Harvard University Press, 1982, Cambridge, MA, USA. ISBN: 0-6743-6446-5.

[162] W. McCulloch, W. Pitts, "A logical calculus of the ideas immanent in nervous activity," Bulletin of Mathematical Biophysics, 5, 1943.

[163] R. A. McIntyre, "Bach in a box: The evolution of four-part baroque harmony using a genetic algorithm," In First *IEEE Conference on Evolutionary Computation*, pp.852-857, 1994.

[164] L. Meyer, "Emotion and Meaning in Music", *University of Chicago Press*, 1956.

[165] Microsoft Directmusic. web page *http://www.microsoft.com/DirectX*.

[166] E.R. Miranda, *Composing Music with Computers*. Focal Press, 2001. ISBN 0240515676.

[167] E.R. Miranda, web page: *http://neuromusic.soc.plymouth.ac.uk/*.

[168] E. R. Miranda, J.S. Biles, "Evolutionary computer music," Springer, 2007.

[169] E.R. Miranda, S., Kirby, and P. Todd, "On Computational Models of the Evolution of Music: From the Origins of Musical Taste to the Emergence of Grammars," *Contemporary Music Review*, Vol. 22, No. 3, pp. 91-111, 2003.

[170] F. Xue, A.C. Sanderson, and R.J. Graves, "Pareto-based Multi-Objective Differential Evolution," in Proceedings of the *Congress on Evolutionary Computation (CEC 2003)*, vol. 2, pp. 862-869, Canberra, Australia, IEEE Press, 2003.

[171] N. Mori, H. Kita, and Y. Nishikawa, "Adaptation to a changing environment by means of the thermodynamical genetic algorithm," volume 1141 of LNCS, pp. 513522. Springer Verlag Berlin, 1996.

[172] N. Mori, S. Imanishi, H. Kita, and Y. Nishikawa, "Adaptation to changing environments by means of the memory based thermodynamical genetic algorithm," In T. Back, editor, International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, 1997.

[173] MC. Mozer , "Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multiscale processing," Connection Science, 1994.

[174] J. Nakamura, T. Kaku, T. Noma, and S. Yoshida, "Automating Background Music Generation Based on Actors' Emotion and Motions" in S.Y. Shin and T.L. Kunii(eds.), Computer Graphics and Applications (Proc. Pacific Graphics '93), World Scientific, pp. 147-161, 1993. (The revised/extended version appeared as 19.)

[175] T. M. Nakra, "The Digital Baton: a Versatile Performance Instrument," International Computer Music Conference, Thessaloniki (Greece), 1997.

[176] T. M. Nakra, R. Picard, "The Conductor's Jacket: A Device for Recording Expressive Musical Gestures," International Computer Music Conference, Ann Arbor (MI), 1998.

[177] T. M. Nakra, Y. Ivanov, P. Smaragdis, C. Ault, "The UBS Virtual Maestro: an Interactive Conducting System," in the Proceeding of *NIME09*,Jun3-6, 2009.

[178] F. Neri, V. Tirronen, "Scale factor local search in differential evolution," in *Memeric Comp.*, 1:153-171, 2009

[179] K. P. Ng, K. C. Wong "A new diploid scheme and dominance change mechanism for non-stationary function optimization," In Proceedings of the *Sixth International Conference on Genetic Algorithms*, pp. 159166, Morgan Kaufmann, 1995.

[180] Oxford Advanced Learners Dictionary, *http://www.oup.com/oald-bin/web getald7index1a.pl*, Cited 17 Jan 2006.

[181] K.E. Parsopoulos, D.K. Taoulis, N.G. Pavlidis, V.P. Plagianakos, and M.N. Vrahatis, "Vector Evaluated Differential Evolution for Multiobjective Optimization," in Proceedings of the *IEEE Congress on Evolutionary Computation (CEC 2004)*, vol. 1, pp. 204-211, Portland, Oregon, USA, June 2004.

[182] A. Pazos, SA. del Riego, J. Dorado, J.J. Romero-Cardalda, "Connectionist system for music interpretation," in Proceedings of *International Joint Conference on Neural Networks (IJCNN99)*, vol 6. IEEEE, New York, pp 40024005

[183] S. Phon-Amnuaisuk, "Composing Using Heterogeneous Cellular Automata," in Proceedings of the *EvoWorkshops 2009 on Application of Evolutionary Computing*, pp.547-556, 2009.

[184] S. Phon-Amnuaisuk, G. Winnings, "The four-part harmonization problem: a comparison between genetic algorithms and rule-based system," in Proceedings of the *AISB99 Symposium on Musical Creativity*, pp. 28-34, 1999.

[185] W. Piston, M. DeVoto. *Harmony.* W.W. Norton, 1987. ISBN 0393954803.

[186] J. C. Pun, "Gesture Recognition with Application in Musical Arrangment," University of Praetoria, 2006.

[187] C. L. Ramsey, J. J. Grefenstette, "Case-based initialization of genetic algorithms", In S. Forrest, editor, *Fifth International Conference on Genetic Algorithms*, pp. 8491, Morgan Kaufmann, 1993.

[188] F. K. H. Quek, "Eyes in the Interface, Image and Vision Computing", volume 13, 1995.

[189] C. Roads, "An overview of music representations," in Baroni M, Callegari L (eds) Musical grammars and computer analysis, Casa Editrice Leo S. Olschki, Florence. ISBN 2147483647, 1982.

[190] C. Roads, "Grammars as representations for music," in Roads C, Strawn J (eds) Foundations of computer music. MIT Press, Cambridge, Mass. ISBN 0-262-18114-2 37, 1985.

[191] T. Robic, and B. Filipic, "DEMO: Differential Evolution for Multiobjective Optimization," in Proceedings of the *Third Internation Conference on Evolutionary Multi-Criterion Optimization EMO 2005*, pp. 520-533, Guanajuato, Mexico, Springer, Lecture Notes in Computer Science vol. 3410, 2005.

[192] J. D. Schaffer, "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms," in Proceedings of the *First International Conference on Genetic Algorithms and their Applications*, pp. 93-100, Lawrence Erlbaum, 1985.

[193] H.Schenker, (1935) "Der freie Satz, 2nd edn. Neue musikalische Theorien und Phantasien, 3," Universal-Edition, Vienna. ISBN B0000BN9B5

[194] K. Scherer, M. Zentner, "Emotional effects of music: Production rules," Music and Emotion, Theory and research, pp. 361-392, 2001.

[195] T. Schlmer, B. Poppinga, N. Henze, S. Boll, "Gesture recognition with a wii controller," in *TEI 08: Proceedings of the 2nd international conference on Tangible and embedded interaction*, pp 11-14, New York, NY, USA, 2008. ACM.

[196] B. Schottstaedt, "Automatic species counterpoint," Tech Rep. STAN-M-19, Stanford University CCRMA. A short report appeared in *Current Directions in Computer Music Research*, Mathews and Pierce eds., MIT Press, 1989.

[197] E. Schubert, "Measurement and Time Series Analysis of Emotion in Music", PhD thesis, University of New South Wales, 1999.

[198] T. Stapleford, J. Robertson, A. Quincey, and G. Wiggins. "Real-time music generation for a virtual environment", in Proceedings of the *ECAI 98 Workshop on AI/ALife and Entertainment*, Brighton, 1998. Dep. of Artificial Intelligence, University of Edinburgh.

[199] M. Steedman, "Formal grammars for computational musical analysis," INFORMS, Atlanta, October 2003

[200] R. Storn, and K. Price, "Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces," Technical Report TR-95-012, International Computer Science Institute, Berkeley.

[201] T. Takagi, M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," in Proceedings of *IEEE Transactions on Systems, Man, and Cybernetics*, vol 15, pp. 116-132, 1985.

[202] B. Tillmann, "Connectionist simulation of tonal knowledge representation," Publications Journees dInformatique Musicale, Issy-Les-Moulineaux, 1999.

[203] B. Tillmann, J.J. Bharucha, E. Bigand , "Implicit learning of tonality: A self-organizing approach," Psychological Review, 107/4, 2000.

[204] PM. Todd , "A connectionist approach to algorithmic composition," Computer Music Journal, 13/4, 1989.

[205] R. K. Ursem, "Multinational GA optimization techniques in dynamic environments," in D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H. G. Beyer, editors, *Genetic and Evolutionary Computation Conference*, pp. 19-26, Morgan Kaufmann.

[206] K. Wassermann, K. Eng, P. Vershure, J. Manzolli, "Live soundscape composition based on synthetic emotions," in Proceeding of *IEEE Multimedia*, pp. 82-90, 2003.

[207] `www.jsbchorales.net`. Website on Bach's chorales. A list with BWV numbers can be found here: `www.jsbchorales.net/bwv.shtml`.

[208] G. Wiggins, G. Papadopoulos, S. Phon-Amnuaisuk, A. Tuson, "Evolutionary methods for musical composition," In *International Journal of Computing Anticipatory Systems*, 1999.

[209] Wiimote Gesture Recognition, In Procediings of the *15th Conference and Competition STUDENT EEICT 2009*,vol. 4, pp.344-349, 2009.

[210] R. Winter, "Interactive Music: Compositional Techniques for Communicating Different Emotional Qualities," Master Thesis, University of York, 2005.

[211] C. G. Wolf and J. R. Rhyne, "A taxonomic approach to understanding direct manipulation," In Journal of the Human Factors Society 31th Annual Meeting, 1987.

[212] I. Xenakis, "Musiques formelles = Revue Musicale," n. 253-254, 1963, pp.232, Edition : Paris, Stock, 1981.

[213] F. Xue, A.C. Sanderson, and R.J. Graves, "Multi-Objective Differential Evolution and Its Application to Enterprise Planning," in Proceedings of the *IEEE International Conference on Robotics and Automation*, 2003.

[214] E. Zitzler, K. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," in Proceeding of the *Genetic and Evolutionary Computation Conference*, 1999.

[215] V. Zouhar, R. Lorenz, T. Musil, J. Zmolnig, R. Holdrich, "Hearing Vareses Poeme Electronique inside a Virtual Philips Pavillion," in Proceedings of *ICAD 05-Eleventh Meeting of the International Conference on Auditory Display*, Limerick, Ireland, July 6-9, 2005.