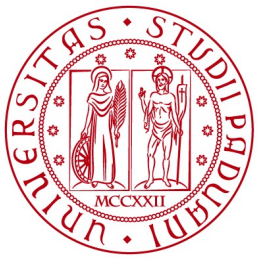


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Head Office: Università degli Studi di Padova
Department of Information Engineering
Department of General Psychology

Ph.D. Course in: Brain, Mind and Computer Science
Curriculum: Computer Science for Societal Challenges and Innovation

Computational Musical Creativity Inspired by Musicological and Statistical Analysis

A Serious Game for Rhythmic Interaction and

A Corpus-Based Hierarchical Representation Method for Style Imitation

Coordinator: Prof. Anna Spagnolli
Supervisor: Prof. Antonio Rodà
Co-supervisor: Prof. Massimo Grassi

Ph.D. Student: Filippo Carnovalini

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

First edition, April 15, 2022

Understand what is the will of the Lord. Be filled with the Spirit, addressing one another in psalms and hymns and spiritual songs, singing and playing to the Lord in your hearts.

Ephesians 5:17-19.

Acknowledgements

This thesis was funded by a doctoral grant by the **University of Padova**. During the period of the Ph.D., I also received funding from **Fondazione Ing. Aldo Gini**. I am extremely grateful towards my supervisors, **Prof. Antonio Rodà** and **Prof. Massimo Grassi** for all the help throughout these years. I also wish to thank **Prof. Sergio Canazza**, and all of the people at the **Centro di Sonologia Computazionale**, my home lab, as well as **Prof. Geraint Wiggins** and everyone at the **CC Lab at VUB** for welcoming me among them.

I also wish to thank everyone who worked with me, discussed my work, or even just shared a coffee or a spritz: exchanging ideas is not only a most important activity for a Ph.D. student, but also a most enjoyable one.

The final thanks go to my friends and my fraternity, and most importantly to Serena, who became my wife during this Ph.D. and who, even right now as I'm writing the finishing touches on my thesis, is sitting beside me.

Abstract

The field of Computational Creativity tries to obtain creative behaviours from computers, to further the understanding of what regulates creativity and what is possible to obtain from computational systems. One common effort within this field is that to have computers write music, as this is an activity that is recognized to require creativity.

In this thesis I review some of the main approaches to music generation as well as some questions that remain unanswered within the field. In the work described here, I focus on two of these, namely, interaction with generated music, and long-term structure, trying to understand the role these two aspects can have in music generation.

A serious game for rhythmic interaction is presented. This game allows two players to freely create a rhythm by interacting via MIDI drum pads. The software detects the tempo and meter they are playing, and adds a musical augmentation to their interaction, increasing the aesthetic value of their gaming and social experience. This is shown to be an effective way of creating a captivating experience for users.

A system for the analysis of musical structure is also presented. This algorithm uses ad hoc representations of musical content, based on tree representations used by musicologists. Starting from the basic representations that were known in literature, this algorithm builds further abstracted representations that summarize the structural aspects of an entire piece and then of an entire corpus. Some example applications are shown, including an algorithm for music generation that leverages these representations and Information Theory concepts to create novel music that shows a structure similar to the ones found in the example corpus. While the method for generating the melodic material used for these novel pieces is not fully capable of generating realistic melodies, the algorithm manages to create satisfactory long-term structure, thanks to the implemented representations.

Sommario

Il campo della Creatività Computazionale prova a ottenere comportamenti creativi dagli elaboratori, per migliorare la comprensione di cosa regola la creatività e approfondire cosa sia possibile ottenere dai sistemi informatici. Una applicazione comune in questo campo è la composizione automatica di musica, in quanto questa attività notoriamente richiede creatività.

In questa tesi descrivo i principali approcci alla generazione di musica, come anche alcune domande in questo campo che rimangono aperte. Nel lavoro qui descritto, mi focalizzo su due di queste, ovvero interazione con la musica generata, e la struttura a lungo termine, cercando di capire il ruolo che questi due aspetti possono avere nella generazione di musica.

Un gioco serio per l'interazione ritmica è qui presentato. Il gioco permette a due giocatori di creare liberamente un ritmo usando due pad ritmici MIDI. Il programma riesce a seguire il tempo e il metro che i due suonano, aggiungendo musica alla loro interazione, aumentando così il valore estetico della loro esperienza di gioco e sociale. Questo gioco si dimostra un mezzo efficace per creare una esperienza coinvolgente per gli utenti.

Viene presentato anche un sistema per l'analisi di strutture musicali. Questo algoritmo usa rappresentazioni ad hoc della musica, basate su rappresentazioni ad albero usate dai musicologi. Partendo da queste rappresentazioni di base che erano note in letteratura, l'algoritmo costruisce ulteriori astrazioni che sintetizzano gli aspetti strutturali di un intero brano, e poi di un intero corpus di brani. Alcune applicazioni di esempio vengono riportate, incluso un algoritmo per la generazione di musica che sfrutta queste rappresentazioni e la Teoria dell'Informazione per creare nuovi brani che mostrino una struttura simile a quelle trovate nel corpus di esempio. Sebbene il metodo per la generazione di melodie usato in questi brani non è del tutto in grado di creare melodie realistiche, l'algoritmo riesce a generare strutture a lungo termine soddisfacenti, grazie alle rappresentazioni implementate.

Contents

List of Figures

List of Tables

1

Introduction

1.1 | The Search for the Creative Machine

What is *Creativity*?

While the term is of fairly common use in everyday life, giving a precise definition of this concept is not a trivial task. The general idea is that it relates to the ability that some human individuals possess to create something that did not exist before. Upon further reflection, one can notice that most of the times these "creations" start from concepts that already existed, or at least that could already have existed, but that nobody had already explicitly linked in a fixed product. This kind of "novel linkage" is what brought us works of art such as Dalí's "The Persistence of Memory": clocks had been painted before, and everybody has experienced that things can melt, but nobody had yet linked these two concepts in a painting.

There is another question relating to creativity that raises even more problematic considerations: can computers be creative? The usual experience with machines is that we humans give a set of instructions to the machine along with some initial data (the input), and we expect the machine to behave in a way that is fully deterministic, always giving the same output when the same input is given. Moreover, we expect that the output should be something that can be fully expected and computed even without the help of a computer, albeit the computation of the output could be extremely time-consuming (otherwise we would not have resorted to computers in the first place).

The word "deterministic" seems to be the exact opposite of our understanding of the concept of creativity, and yet the idea of obtaining creative behaviors from computers has inspired the writing of a notable amount of scientific publications, that can be collected under the field of *Computational Creativity* (CC), defined as:

> "The philosophy, science and engineering of computational systems which, by taking on particular responsibilities, exhibit behaviours that unbiased observers would deem to be creative." Colton and Wiggins (2012, p. 21)

Practitioners of this field share the interest in gaining a better understanding of how creativity works, and to what extent it can be replicated via a computer system. The above definition underlines the diversity in background of the people researching CC. Such a diverse community is for sure source of many interesting insights, but there is also room for many different goals and perspectives that sometimes can make it hard to understand what is the current general direction of the research, and what directions should be explored for the advancement of the field (Lamb et al., 2018).

One of this field's goals is for sure answering to the above question "can computers be creative?", that is most interesting to computer scientists and engineers that wish to create advanced models of artificial intelligence with creative capabilities. Yet, this is only one of the possible goals: artists could be more interested in finding out how computers can help express their own creativity, while psychologists and philosophers are more interested in using computer models of creativity to better understand creative processes that happen in humans (Pearce et al., 2002). The diversity of the goals is reflected in the diversity of the literature concerning CC. Some work was devoted to the definition or the assessment of creativity itself. A notable amount of contributions focus on the design of systems that are meant to be creative, sometimes designing them starting from some definition of creativity, and thus the trying to actually obtain output that is creative according to a certain evaluation method. Other times, the system is simply designed to tackle tasks that are commonly considered to involve creativity (usually artistic tasks like painting or composing music), but the process involved in the generation of such output does not necessarily involve creativity. More often than not, the creativity of the creative systems

is not evaluated in any formal way, either having just summary evaluations of the quality of the output or not having any evaluation at all (Jordanous, 2012). This might be fine for the artistic goal of empowering the creativity of humans through computational means, but is not acceptable for AI practitioners trying to understanding whether computers can exhibit creative behaviors.

One especially prolific research task of CC is that of music generation, that has interested computer scientists even before the birth of the term "Computational Creativity". Ever since the early days of computing, scientists and engineers have used computers for musical task, creating digital synthesizers, developing engraving software, and also writing procedures that generate musical scores, to be performed either by computers or by humans. This task was called "Algorithmic Composition", and as the name suggests was related to a well defined procedure (an algorithm, once again a concept distant from creativity (Nierhaus, 2009; Fernández and Vico, 2013; Papadopoulos and Wiggins, 1999). A name that is more common today is "Musical Metacreation", that suggests the fact that the programmer creates a system that in turn can create some kind of music (Bodily and Ventura, 2018). Music is especially interesting for the investigation in CC because of the broad possibilities that it offers in terms of mathematical and computational representations, and because it does not need explicit semantics like other forms of art such as poetry or non-abstract painting (Wiggins, 2020). Moreover, generating music via computational systems require a deep understanding of what defines music in general and within a particular style. For this reason, the study of music generation is also strongly intertwined with (Computational) Musicology, and with Music Information Retrieval (MIR), fields which both give tools for music generation and obtain further insights by the studies on Musical Metacreation (Pearce et al., 2002).

1.2 | Motivations

1.2.1 | Computational Creativity as the Final Frontier of AI

It is not uncommon for practitioners of Computational Creativity to be questioned about the usefulness of delegating creative tasks to computers. Some even argue that creativity is something that only humans can possess, and trying to simulate it via computers is either unethical or impossible (Sturm and Ben-Tal, 2021).

Most of these concerns derive from having a romanticized idea of what it means to be creative (Hodson, 2017; Boden, 2004), but it still interesting to address *why* it is useful to study the creative capabilities of machines. On this subject, it is important to clarify that while many Computational Creativity studies focus on artistic subjects, the field is not limited to such applications. As stated above, the definition of CC does not restrict the kind of "creative behaviours" to be shown by computational systems. The artistic applications are often chosen as a study case because these activities are widely recognized as needing creativity, but there are examples of creative systems that deal with mathematics (Lenat, 1976), chemistry (Buchanan and Feigenbaum, 1978), and even cooking (Pinel et al., 2015). More generally, creativity is understood as a fundamental feature of human intelligence, and a person completely devoid of creativity would not be considered intelligent, as the lack of creativity means that it would be impossible to solve (or even face) any problem that was never encountered before. Such adaptability, and the fact that we expect Computational Creative systems to be able to surprise even their developers, can make this field a futuristic research for what really is the limit of what computers can do (Colton and Wiggins, 2012). Aside from the mathematical limits given us by logic research, it is still unclear what is the limit of what can be practically obtained by computers. CC proposes to expand these limits by shifting the focus from *problem-solving* to *artifact generation*. A solution to a problem is not excluded in this paradigm, but in this case the solution becomes the artifact to be generated. Finally, another motivation for the study of Computational Creativity is the study of Human Creativity through computers. Software requires very strict definitions, and a deep understanding of what we are trying to obtain from the machine. Deepening

the understanding of Creativity in computers can also help understand the mechanisms and cognitive activities behind human creativity, which can become way less mysterious in this way, as Margaret Boden suggests (Boden, 1996).

1.2.2 | Computational Music as a Creative Tool for Humans

The above paragraph lists some motivations for the study of Computational Creativity in general, and those motivations can also be applied to the task of Music Generation as it is a often used as an example application of creativity in computational systems. Despite that, it would be possible to argue that generating music in itself is not beneficial to humans, as such an artistic activity should not be delegated to computers. It is thus worth to outline some motivations for the study of Music Generation that do not rely on the general study of creativity. First of all, it is important to notice that Music Generation Systems are complementary to human composers, as they can be used when a human composition is not feasible. For example, in interactive media, music generated by computers can adapt in real time to the narrative of the media, reacting to what the user is doing. One common example is that of video games: a human composer has limited control to how the player will interact with the game, and therefore cannot write music for any possible situation. It is common to write different pieces or variations of a different piece to allow some level of adaptability (e.g., often when the players enters a body of water in a video game the instrumentation is often altered to reflect the different sound transmission of the different material, making the sound more "muffled"), but it is simply impossible to write different variations for all possible interactions. Some games resort to procedural music for this reason, but a fully autonomous generation system could give a unique experience to the player, that is completely tailored on his playing style (Brown, 2012a). Additionally, music could become a novel mean of Human-Computer Interaction. If a computer is capable of generating music in real time to express feelings or to accompany certain situations, this would allow a sonified user experience, and allow for more effective Affective Computing. In particular, music generation can be helpful in creating ad-hoc Music Therapy experiences for people to enjoy at home (Cheatley et al., 2020, 2019). Finally, Music Generation Systems

can be inserted in the artistic activity of a human composer in many ways (Pearce et al., 2002). The simplest way is to generate some musical excerpts from which the composer can draw inspiration and work upon, but there are also generation systems that are explicitly designed with the goal of co-creation between human and computers, and these can help human composers express themselves in ways they would not have imagined, as if it were another human collaborator. Potentially, these systems can also help people who have little musical knowledge create music, as a sort of "musical assistant" (Navarro et al., 2016; Collins and Coulon, 2012).

1.3 | Research Directions

This paragraph will outline the research questions taken into account in this thesis, and formulate the research directions that this work tries to address. There are two principle directions: the social impact of generated music, and structure in music generation. While seemingly quite divergent, the substantial importance of each makes them interesting to study in conjunction so as to lay a foundation for more a holistic study of computational creativity. Moreover, while exploring these direction this thesis will include examples of how computational music generation can remain centered on the human user, as long as socially useful applications are considered and the tools employed have roots in human cognition.

1.3.1 | Social Interaction and Music Generation

Music is a fundamentally social phenomenon, and is a vital part of most cultures (Wiggins, 2020; Meyer, 1989). Given the non-semantic nature of music, its social role does not derive from intrisic meaning in music, but rather from the interplay between music and culture, as well as the interaction between musicians and listeners, and between musicians. These kinds of interactions will shape the cultural meaning added to music, while the cultural and social conventions will alter the nature of the musical interactions at the same time (Pearce, 2018; Meyer, 1956).

When music is being generated by a computer, its social value becomes dubious, as in genereal algorithms for music composition lay outside the "social loop" described above. One side effect of this is the different perception of the value of computer-generated music (Moffat and Kelly, 2006), as well as the difficulty in seeing the utility of having computer-generated music at all (Sturm and Ben-Tal, 2021). Even when the composition is not entirely delegated to the machine but rather arises from an interplay between human and computer, as in the case of Computer-Assisted Algorithmic Composition, the social implications diverge from the traditional interplay between humans adding aspects of human-computer interaction (Sturm and Ben-Tal, 2021; Biles, 2013a). Some questions arise when considering Socialty, Interaction, and Computational Creativity applied to music.

What is the role of computationally creative system for humans? What should the computers achieve for the utility of human users? Can the computers actually be more useful than human composers within certain situations? Can algorithms capture the social standards and implications that regulate music making among humans? Can a computer understand and respond to human emotion in a social environment? Can emotional content allow for more creativity in a computational system? Can emotional and social-aware content make computer-generated music more useful for end users? Can generated music be helpful to affective computing? Can generated music be useful in music therapeutic settings?

In this thesis I explore how real-time music generation can be useful as a Human-Computer Interface, and how it can be useful in a Music Therapy setting. The application described in chapter 3 is one that is designed to be entirely human-centric, as the user benefits from the presence of generated music, and is one where Computational Creativity is necessary, as it would be nearly impossible to employ a human composer with the same degree of effectiveness and non-intrusiveness. In my work, the social context is not explicitly considered, but rather arises from the usage of the system, and is mainly related to sociality and affiliation (Hove and Risen, 2009).

1.3.2 | Structure in Music Generation

Structure is a term with multiple meanings, even within the context of music. In general, it refers to reuse and variation of musical content within a piece, but also to contrast between different sections. Both these aspects can be analyzed at different levels of details: considering groups of just few notes (and comparing motifs), or taking sets of many bars in considerations (exploring what is usually called "form" in music theory). Within the context of Computational Creativity and of Music Generation specifically, the problem of structure is usually related to the fact that generated melodies seem convincing for a few seconds, but then start to "wander off" (Dai et al., 2018), calling for algorithms that can consider the long-term development of a piece.

What would be the best algorithm for such a long-term approach to music generation? What kind of information can be extracted from music and learnt from a corpus that is useful to this goal? How can one formalize structure while considering human cognition? Should the algorithm for music generation be the same at a short-term local level and at a long-term structural level? Would listeners perceive a structured piece be as more creative, or at least more realistic, than one without clear structure? Does structure relate to musical narration and emotional content? Can such narration be computationally studied in a cognitively aware manner? Is it possible to apply what we learn about musical structures in other fields, such as language?

In this thesis, structure is approached by using hierarchical representations of music, that try to describe latent structural information of melodies. These representations will serve as a way to compare different sections of a melody, and to define common structures within a corpus. Chapter 4 will describe the algorithms to analyze these structure, and how I applied Information Theory concepts to the obtained representations within a music generation context.

1.4 | Contributions

The following is an outline of the contributions described in this thesis:

- A review of the state of the art of Computational Creativity and Music Generation systems, underlining open challenges in the field.
- A musical serious game where two players are required to create a rhythm together on shared digital pads. The game analyzes their rhythmic interaction, inferring the tempo and the meter in real time, and automatically generates a musical accompaniment that fits the interaction.
- A system for describing typical structures of a musical corpus through a probabilistic description, that can be used in combination with Information Theory concepts to describe the typicality of a new piece given a corpus.
- A system for generating music that imitates the style and structure of a given corpus, based on the same probabilistic description.

1.5 | How to Read this Thesis

This section will outline the contents of the rest of the thesis, giving an indication on how to frame what is presented. The rest of the thesis is divided into three main chapters, followed by a conclusions chapter. Since the chapters cover relatively diverse topics, it is useful to describe how the thesis was developed to understand how the topics are related to each other.

The presented work is framed within the scope of a three-years PhD program. Since this work spanned from the end of 2018 to the end of 2021, this work was severely affected by the Covid-19 pandemic, which forced non-optimal work conditions for the majority of the PhD duration.

It is important to read the rest of the thesis keeping in mind that the presented work has the ultimate goal of advancing the knowledge on music generation for the ultimate goal

of computational creativity. As such, this is a computer science thesis, and despite the fact that the topics this thesis discusses are inherently multidisciplinary, it would be a mistake to expect this thesis to deal with the matter at hand in a musicological, neuroscientific, music therapeutic, psychological, or design point of view. While these disciplines play a role in the following chapters, the contributions should be mainly considered as computer science contributions.

Chapter 2 is the result of the first year of the PhD program, and tries to give a broad introduction to the fields in which the thesis tries to give a contribution. As such, it describes the field of Computational Creativity in general, giving a review of definitions of creativity as well as of the evaluation of creativity, which is one of the main research areas in this field. The latter part of the chapter is devoted to the study of music generation systems, describing techniques and algorithms used to this goal and discussing open problems in the field.

The chapter tries to frame the thesis within the state of the art of music generation with a focus on creativity, but does not represent a complete introduction to all the works relevant to the topics described in this thesis. Since the following chapters introduce applications of music generation that required further study of technical solutions to more specific problems, those chapters also include introductory section which further explore the literature related to those sub-tasks.

What this chapter instead includes is a brief discussion on the open problems of music generation, which was the driving factor for choosing the topics to cover in the subsequent years of the PhD program and the subsequent chapters. The main focus which was chosen as a result of this study of the literature is that of structural and narrative development of a musical piece. This led to the study of music representations designed to capture musical structures, and to the study of music as a social tool, with the long term goal of exploring social and emotional meaning in music generation.

The main application described in Chapter 3 is a serious game for musical interaction, inspired by music therapy. Such game can exemplify both how computational music generation can be necessary in some real-time applications and how social interaction plays a role in creativity and in creative activities, such as music making, as well as providing the grounds for further research on how such interactions can be used to describe emotional development in generated music.

The design goal was to create an interactive system that would allow people without musical expertise interact in a musical manner, similarly to what would happen between musicians playing together. To obtain this goal, a game was designed. The game requires players to interact solely rhythmically by hitting midi pads and subsequently adds a musical augmentation to their interaction, generating music that is synchronized rhythmically with the interaction but adds melodic and harmonic layers to increase the aesthetic experience. To obtain this synchronization, a system for tempo and meter tracking was embedded into the system. Since the data available to the tempo tracking system is limited to purely rhythmic information without pitch nor timbre, this algorithm represents a novel contribution despite not being the focus of the work in the chapter and having strict requirements that rarely apply to other real-life applications for which better performing algorithms can be found in literature.

The game was implemented as a fully functional prototype, and a fist user study was performed collecting positive users impressions. The arrival of the Covid-19 pandemic soon after the completion of the prototype, halted further development and testing of the system, since it is strongly rooted in the close co-presence of two users. This also meant that it was not possible to plan, within the time limit of the PhD program, the further evaluation needed to assess the quality of the system in a therapeutic setting and as a creative tool. While this hinders the quality and the scope of the contributions represented by this work, it was still included in this thesis as it is the result of a notable amount of time and resources within the PhD, and because the contributions it can give are still significant despite not being able to fully assess the impact of socialty on the creative process.

The second main line of research, which is reported in Chapter 4, is the study of representations for musical content that capture structural aspects of music. The goal in this case was to enable music generation systems to explicitly consider structure to obtain better long-term generation of music, which is another open problem in music generation recognized by many researchers.

The basic idea behind this work was to study representations that would consider reuse of melodic material in a way that is not directly dependent on the actual melodic content of the piece, meaning that different melodies built in a similar manner could ideally be represented by the same structural representation. To do this, a first representation inspired by Schenkerian analysis was implemented, and two further representations that build upon the first completed the analysis system to allow the representation of a set of pieces instead of a single piece. It is worth noting that this system performs musical analysis, but in this context the term analysis is not meant to mean a musicological analysis. While this tool could potentially be useful for musicologists if further expanded, its main goal is to extract structures useful to a music generation system.

To exemplify how this system can be used in such context, a simple music generation system is also presented in this chapter. Being based on first-order Markov chains, the quality of the generated melodic material is limited, but the capability of structuring melodic phrases beyond what this simple Markovian approach would allow for is still evident.

The final chapter gives a summary of the work described in the rest of the thesis, also adding a list of the scientific publications and deliverables that resulted from it, closing with a little discussion.

Originality

While this thesis represents an original contribution and contains substantial advancements from any prior work, part of it was published in prior publications.

Parts of Chapter 2 were published in an open access journal (Carnovalini and Rodà, 2020).

Parts of Chapter 3 were published in conference papers (Carnovalini et al., 2019; Carnovalini and Rodà, 2019b).

Parts of Chapter 4 were published in conference papers (Carnovalini et al., 2021a,b).

2

Background and Literature Review

In this chapter, I provide a broad introduction to the field of Computational Creativity, and a review of the state of the art on Music Generation.

Throughout this thesis, rather than using the terms "Algorithmic Composition" or "Musical Metacreation", which sometimes entail the fact that certain applications are favored over others, I will use the more neutral term "Music Generation Systems" (MGSs). This is not the only review of the subject (some are cited in Section 2.2.2), but most reviews focus solely on the technical approaches used for music generation. This chapter will include such a description of the main methods for generating music, but its latter part will also try to give a picture of the challenges that are still not fully addressed, and what kind of solutions have been tried or proposed to overcome those problems.

2.1 | Computational Creativity

2.1.1 | Defining Creativity

What is Creativity? This is a question that many researchers have faced before, especially in the last seven decades. A specific event gave the inception to research in creativity, as reported by James Melvin Rhodes (1961, pp. 305-306):

> "The big push of interest in the subject of creativity began in 1950 when J. P. Guilford of the University of Southern California was president of the American

> Psychological Association. Guilford said in his presidential address to that organization that he found an appalling lack of research on creativity. He said he had searched Psychological Abstracts for a quarter of a century and found out that only 186 out of 121000 entries dealt in any way with creativity, imagination, or any topic closely related."

Since that year, psychological research on creativity exploded, exploring many facets of what defines and stimulates creativity in humans. Some work was focused on the study of what are the personality traits of the creative person (Rhodes, 1961; Getzels and Jackson, 1962), as well as what external factors can positively or negatively influence creativity (Amabile, 1983b,a). Other researchers were more interested on the mental processes that happen in the creation of something creative, and finally much work were dedicated to the definition of creativity itself.

2.1.1.1 | Creativity as Novelty and Value

The newfound abundance of research led to having hundreds of definitions of creativity in literature. In their papers, Sarkar and Chakrabarti analyzed over 200 of those (Sarkar and Chakrabarti, 2008, 2011; Ranjan et al., 2018), finding that the factors that have been used as indicators for creativity can be grouped in two main categories: *Novelty* (or unusualness, unexpectedness, surprise, originality) and *Value* (or usefulness, quality, appropriateness, meaningfulness). This subdivision is not new at all, as Stein (1953) had already proposed a definition of a creative work as novel and useful or satisfying. Novelty is usually considered the defining characteristic of a creative artifact, but value is also necessary: it is easy to think of something that has never been built before, like a car with fifteen wheels, but while such car would be novel, it would have higher maintenance costs, with little or no increase in performance. This kind of novelty lacks value: creativity (that includes value) introduces innovations useful to the purpose of the created object, possibly leading to a general advancement in its own field. One example of creativity in the field of car manufacturing could be the introduction of hybrid cars: the idea of using two different energy sources was novel, but hybrid cars are now common because of the advantages they

bring to their owners in terms of efficiency. On the contrary, it is highly unlikely that our fifteen-wheeled car could become an industrial standard.

While Novelty and Values are surely important features of creativity, these give only a vague description of creativity. From their study of the literature, Sarkar and Chakrabarti (2008, p. 6) reached a somewhat more complete definition of creativity:

> "Creativity occurs through a process by which an agent uses its ability to generate ideas, solutions or products that are novel and valuable."

2.1.1.2 | The Four Perspectives of Creativity

The above definition points out that creativity is a concept that cannot be ascribed to the final artifact, but must consider its creation. In particular, it underlines the existence of a process, used by an agent, to create a product. These represent three of the four "P's" of creativity, first identified by Rhodes (1961): Person, Process, Product, Press. Rhodes was interested in the educational aspects of creativity and in educating children to be creative, and his discussion on the four P's reflects this interest. To sum up his reflections:

- **Persons**: there are personality traits that make some individuals more prone to creativity than others;
- **Process**: while personality cannot be taught, there are processes (usually some kind of heuristics) that can help to find novel and creative ideas;
- **Press**: the environment in which a person lives is fundamental to the creative result, and more often than not creative deeds can be considered the work of a group of people. The importance of press is also shown by the amount of quasi contemporaneous inventions around the world from people that had no contact with each other;
- **Products**: once the idea is distilled in a fixed media, it becomes a product. The products can be classified and studied more easily than the other aspects, making them useful to the scientific study of creativity.

It is important to notice that while many researchers on creativity and CC use this notion of the four P's in their work, their meaning has changed, especially for the Press. For example, Lamb et al. (2018) describe the four terms as follows:

- **Person**: is the human (or non-human agent) who is seen as creative. Person theories study what it is about the agent that makes them creative.
- **Process**: is the set of internal and external actions the agent takes when producing a creative artifact. Process theories study what sort of actions are undertaken when creative work is done.
- **Product**: is an artifact, such as an artwork or a mathematical theorem, which is seen as creative or as having been produced by creativity. Product theories study what it is about the product that makes it worthy of being called creative.
- **Press**: is the surrounding culture which influences people, processes, and products and which judges them as creative or uncreative. Press theories study what it is that leads a culture to view something as creative.

Beside the loss of the plural on Person and Product, the focus shifted from the study of factors that can help the development of creativity to the study of what makes something be considered creative. Press is now listed last, as it includes not just the influences on the creative Person before the creation of the artifact, but also the cultural impact of the Product and its judgement. This shift is probably partly due to a paper by Anna Jordanous, who revisited the concept of the four P's under the light of the evaluation of creativity (Jordanous, 2016), but the definition of the four P's had already started changing soon after the original paper by Rhodes: in Golann (1963) we find the terms *Measurement* and *Personality* instead of Press and Person, showing that while the general idea of the four P's was soon utilized by the scientific community, it took some time to reach widely accepted definitions.

This useful subdivision into four perspectives helps frame the various contributions on creativity, as often each work focuses on only one or two of the above perspectives. For

example, the definitions of creativity as Novelty and Value are focused on the Product, even if Sarkar and Chakrabarti's definition encompasses almost all four P's. In the following sections I review some contributions that focus on the other three perspectives: Person, Process and Press.

2.1.1.3 | Person

Regarding the Person perspective, the study of the personality traits of creative people has unsurprisingly interested many psychologists: already in the first years after Guilford's speech much work emerged (an early review was made by Golann, 1963), and soon was found out that creativity is not directly related to intelligence (Getzels and Jackson, 1962), and a relationship between creativity and humor was also noted (Treadwell, 1970). Guilford himself underlined that creatives emerge for their sensitivity to problems, mental flexibility, and divergent thinking (Guilford, 1957, 1967). The importance of this last trait was exploited by Torrance, who designed the Tests of Creative Thinking (Torrance, 1965, 1974) that give an effective measure for the individuation of creative people (Torrance, 1988). Simonton (2000) gives a review of psychological studies on creativity in terms of personal and developmental traits, as well as the socio-cultural influence of creativity (connecting the Person and the Press perspectives).

Within the field of CC, one could argue that any Turing-complete machine is equivalent in what it can achieve, thus making every computer system equal under the Person perspective. Nonetheless, the Person remains an insightful perspective at a more abstract level, for example when a software system can be viewed as an agent or as a group of agents collaborating together. In this case, the (virtual) personality of each agent could give a different contribution to the system, making it useful to consider psychological personality aspects such as motivation (Guckelsberger et al., 2017) or curiosity (Schmidhuber, 2012), or to try and model in software cognitive aspects of creativity (Wiggins and Forth, 2015; Wiggins and Sanjekdar, 2019).

2.1.1.4 | Process

The Process perspective has interested CC the most, as someone who wishes to obtain a creative behavior from a computer must know how to describe creativity in algorithmic terms. While there is no such thing as a fixed procedure to obtain something creative, it is possible to gain insights on how to obtain creativity from the study of the creative processes of people that have shown great creativity throughout history (and wrote how they reached that idea). This is in part what Margaret Boden did in her book, The Creative Mind (Boden, 2004) (for a shorter introduction to the same ideas see Boden, 1998, 2009). The description of creativity she provides in that book has become extremely influential to the field of CC, also because she used computer models of creativity to discuss her ideas, explaining what was obtained and what was still to be achieved by machines. One of major contributions she gave was the introduction of the idea of "Conceptual Space", i.e. a space where the possible concepts exist, some of which have been explored and some are yet to be discovered. This basic idea allows the distinction of many levels of creativity:

- **Combinational Creativity**: two already explored ideas from a concept space are joined, thus creating an association that is novel;
- **Exploratory Creativity**: some kind of method for the free exploration of the concept space is used, to find regions in the space that are not explored yet but are valuable;
- **Transformational Creativity**: the highest level of creativity is reached when a new idea is found that was not part of the original conceptual space, thus changing the shape of the concept space itself.

The idea of obtaining creative ideas from the union of two known ideas, that Boden called Combinational Creativity, is at the basis of other theories of creativity, although with different names: Koestler (1964) called the same idea *Bisociation*, while Fauconnier and Turner (2008) used the term *Conceptual Blending*. The novelty of Boden's theory lies in the introduction of conceptual spaces, necessary for the definition of the other two levels of creativity. Wiggins (2006, 2019) mathematically formalized these ideas, also showing that Transformational Creativity is equal to Exploratory Creativity on a meta-level.

Another useful notion introduced by Boden is the distinction between *H-Creativity* (historical creativity) and *P-Creativity* (personal creativity). In order for something to be H-Creative, it must be the first time it has appeared in the history of mankind, while to be P-Creative it is enough to be new to the one creating it. As an example, Boden mentions that if a child can prove Pythagoras' theorem without any help, we would find this deed an impressive example of mathematical creativity even if that theorem was demonstrated millennia ago. H-Creativity is what is usually considered novel and/or creative, but Boden argues that P-Creativity is just as important as it originates from the same creative Process.

Another way to look at creativity as a Process is to study it from a cognitive and evolutionary perspective, addressing how creativity is useful and how it could have advantaged creative humans in the context of natural selection, possibly leading to a better understanding of the cognitive processes that guide creativity (Wiggins et al., 2015). Under this perspective, it is also possible to give an explanation to why creativity is highly valued in a social context, which leads us to the next perspective on creativity.

2.1.1.5 | Press

The Press perspective is most interesting to the evaluation and assessment of creativity. This is not just an appendix to the concept of creativity: the working definition for CC seeks behaviours that are deemed to be creative by an unbiased observer, making it necessary to have an external appraisal of the Product before calling something creative. The work of Amabile underlined both the importance of the environment for the development of creativity (Amabile, 1983b; Amabile et al., 1996) and the importance of the assessment of creativity, proposing one of the first formalized methods for the evaluation of creativity, using expert judges (Amabile, 1983a). I will discuss the problems relating to the evaluation of creativity later (see Section 2.1.3).

Even if someone tried to directly assess the creativity of a Product, of the Process behind it, or of the Person, he needs to pass through the lens of human perception (and thus the Press perspective) to be really uderstood (Colton, 2008), making the Press perspective the most ubiquitous. On the other hand, the Press perspective is not enough to give an

indication of creativity, since commercial success or reach of a Product is influenced by a variety of factors that go beyond creativity, or even just its Value (Fraiberger et al., 2018).

2.1.1.6 | Dimensions of Creativity

Another interesting contribution to giving a complete definition of Creativity comes from Jordanous and Keller (Jordanous, 2012, 2013; Jordanous and Keller, 2012, 2016; Jordanous, 2019), who used a statistical language processing techniques to identify fourteen main components of creativity, as described by scientific research on the topic. This study resulted in an unordered list of components, that should be seen as different dimensions of the concept of creativity rather than a systematic description (Jordanous and Keller, 2016):

- Active Involvement and Persistence;
- Dealing with Uncertainty;
- Domain Competence;
- General Intellectual Ability;
- Generation of Results;
- Independence and Freedom;
- Intention and Emotional Involvement;
- Originality;
- Progression and Development;
- Social Interaction and Communication;
- Spontaneity/Subconscious Processing;
- Thinking and Evaluation;
- Value;

- Variety, Divergence and Experimentation.

The notion of Novelty (here called Originality) and Value are kept, but using all 14 components gives a much broader definition of creativity, that considers all the four P's: for example *General Intellectual Ability* is related to the Person, *Progression and Development* to the Process, *Value* to the Product, and *Social Interaction and Communication* is connected to the Press perspective. Jordanous and Keller (2012) explain that not all the components listed above will be as important in all possible creative deeds, so this list also offers the possibility to categorize different kinds of creativity required by different activities.

To my knowledge, there is no work in literature that has given a short definition or a model of creativity based on these fourteen dimensions.

2.1.2 | Computers and Creativity

The above definitions of creativity were general enough to apply to both humans and machines alike (although the discussion sometimes focused on the implication of those theories on computers). It is now time to face the second question posed in the introduction: can computers be creative?

This is a question that seems to be as old as computer science: Lady Lovelace, while commenting the Analytical Engine, mentioned that computers do not have the ability to originate anything on their own (Lovelace, 1843). As paraphrased by Bringsjord et al. (2003, p. 4), her statement reads:

> "Computers can't create anything. For creation requires, minimally, *originating* something. But computers originate nothing; they merely do that which we order them, via programs, to do."

The Countess leaves no room whatsoever for creativity, but other important scientists disagreed with her. Alan Turing, who argued that artificial intelligence should have creative abilities, responded to Lady Lovelace's objection pointing out that she had no real experience in programming, while we now know that a computer can often surprise us by doing the exact opposite of what we intended, until a program is thoroughly checked for bugs (Turing,

1950). This response is somewhat unsatisfying, since it seems that the only accountability for creativity from computers would come from human errors, but in the rest of the article Turing argues that intelligent machines should be able to learn, thus gaining abilities beyond those envisioned by the original programmer.

Another strong argument against computer creativity is that of the "Chinese Room" introduced by Searle (1980). He argues against artificial intelligence in general, but the argument applies to creativity as well. He imagines to be locked inside a closed room, that can accept questions and give answers written on paper, either in English or in Chinese. For the English questions, he would answer normally using his own intelligence, while for the Chinese ones he would use a special script telling him, for any combination of Chinese symbols that he sees, what symbols to write as answer. Supposedly, the English answers would be as good as the Chinese ones to the eyes of the people outside the room (if the Chinese script is good enough), but the person inside would not gain any knowledge of Chinese in this way. Searle argues that computers work in this way, manipulating symbols without having a real understanding of those.

It is hard to argue with Searle's objection, unless we suppose that the manipulations of symbols that happen in computers are in reality not different from those that happen in our brains, if not because of less "computational power" (Minsky, 1982). This vision basically reduces human brains to extremely powerful computers, so that an artificial computer could recreate all of their functions. This is of course far from being a proven truth, and does not fully account for things we experience everyday, such as consciousness, free will, and subjectivity (Chalmers, 1995; Hameroff and Penrose, 2014; Ceroni and Prosperi, 2018).

There is room for a long lasting debate on the possibility of computers being "*really*" creative, but fortunately CC is not ultimately interested in this debate. According to the definition of CC, we want computer systems that have behaviours that an unbiased observer would deem to be creative, and not necessarily behaviours that are *actually* creative. This means that we aim at simulating creativity well enough to trick observers into thinking that the product they are seeing is actually creative.

It is nonetheless important to understand what creativity is, and possibly to incorporate

the definitions of creativity in the generation process, because the unbiased observer will judge creativity in the same way as it would with a human, thus implicitly applying some of the concepts relating to creativity that were illustrated above. The problem of the evaluation of creativity thus becomes central: if the goal is to recreate what an observer would deem creative, it is necessary to give metrics of how creative something would be perceived by an observer.

2.1.3 | Evaluating Creativity

Despite the importance of the evaluation of creativity, most of the scientific publications on evaluation only came about in the last twenty years (Jordanous, 2013). In this section some of the most common creativity evaluation methods are described. To read some more extensive reviews on this subject, I suggest the following: Jordanous (2012, 2013, 2014); Lamb et al. (2018); Pease and Corneli (2018); Ritchie (2019).

2.1.3.1 | Turing Test-Like Approaches

The definition of CC that we used suggests that creativity needs to be assessed via human judgement, leading to evaluation techniques based on the concept of "Turing Test" (Turing, 1950): ideally, if a human cannot distinguish computer creativity from human creativity, the computer has achieved a satisfying level of creativity.

Amabile (1983a) proposed the Consensual Assessment Technique (CAT), which has become the standard evaluation of human creativity (Baer and McKool, 2009). This technique requires a pool of experts independently evaluating a set of artefacts. An artefact can be considered creative if it receives good evaluations and the interrater reliability is high enough (for example having a Cronbach's alpha higher than 0.7). While this method was not originally conceived for CC, it is easy to insert one or more computer generated artefacts along some human made ones, to get a comparison between human and computer creativity. The judges only have access to the artefact, not knowing anything about its author or background (including whether the author is a computer). This means that CAT only evaluates the Product perspective in a non interactive way, making it rather different

from the original Turing Test, but it was included in this section because it operates a comparison between human and computers carried out by a human evaluator.

Pearce and Wiggins (2001) propose a machine composition framework that includes in its final phase an evaluation inspired by the Turing Test (although the authors underline the major difference of not having interaction). While it was initially defined for music generation, it can be applied to CC in general. This framework supposes that a corpus is available to the software, and that some sort of learning is applied to create a "critic" for that corpus. Once new compositions are generated that satisfy the learnt critic, some generated pieces are presented a group of subjects along with composition coming from the corpus. The evaluators are asked to tell whether the compositions they hear are human or machine made (similarly to Turing's imitation game). If their evaluation cannot be statistically distinguished from a random selection, the system is considered effective. This approach, being entirely based on learning a corpus, is arguably not really an evaluation of creativity but rather one of quality in imitating human products.

Ariza (2009) underlines this and other limitations of Turing Test approaches to the evaluation of creativity, showing how sometimes these tests are implemented in a way that he calls "toy Tests", failing to understand that interactivity between human and computers was the main feature of the "Imitation Game", as it was meant to assess intelligence, that is experienced through interaction.Another critic to this kind of tests comes from Soldier (2002), who raises a more fundamental doubt on the capability of non-experts to act as evaluators. This is not surprising (indeed, CAT requires experts), but often Turing Test approaches only require the evaluator to be human.

Bringsjord et al. (2003) propose to go beyond the Turing Test with the "Lovelace Test" (inspired by her statement reported in section 2.1.2). The authors argue that Turing's game could be beat with simple manipulation of symbols without the need of any intelligence (as Searle described with his Chinese Room example). On the contrary, an agent passes the Lovelace Test if and only if it is capable of creating an output of some kind through a repeatable process, and this output cannot be fully explained by the knowledge-base, the architecture, and the core functionalities of the agent. Unluckily, this test is not easy to

perform in real-life situations, and arguably a machine could never pass this test, as every output of a machine is the result of its architecture and functionalities. This might be a good abstract test for *real* creativity, but is not very useful to evaluate CC systems.

A more manageable version of the Lovelace Test was proposed by Riedl (2014), that requires the machine to be able to generate an output that satisfies a set of requirements chosen by a human. The generated output is then evaluated in terms of how well it meets the requirements and if it is "not unrealistic for an average human". This proposal is somewhat unsatisfying, because by losing the strong requirements of the original Lovelace Test it basically falls back to a standard Turing Test, in a way that Ariza (2009) described as "Directive toy Test", meaning a Turing Test where the interaction is only limited to giving initial directives for the generation.

2.1.3.2 | Self-Assessment Frameworks

Another popular approach is to have the author of the system describe the way it works and how it can be considered creative or not, and to what degree. These assessments try to frame the chosen Process in some kind of creativity scale, for example distinguishing if the used process is combinational, explorational or transformative, using Boden's categories. Indeed, this kind of evaluation is reminiscent of how Boden investigated creativity in her book (Boden, 2004).

Colton (2008) introduced these assessments with a reflection on how the evaluation of the Product alone is not enough to evaluate the creativity of a system. He proposes an example, where the same object is obtained through different processes. This can lead to different perceptions of creativity, but obviously only if the process is known to the observer. In that paper, he introduced the concept of the "Creative Tripod", a tripod having *Skill, Appreciation* and *Imagination* as legs, saying that all three must be extended to some degree in order for the tripod to stand.

The tripod framework had little success, possibly because it was not formalized enough, but it remained influential on literature on creativity evaluation. Colton, Pease and Charnley (Colton et al., 2011; Pease and Colton, 2011b,a) described another framework for self-

assessment: the FACE and IDEA models. The FACE model can be used to describe the creative capabilities of a system through a set of symbols that tell if the evaluated system *possesses* or is *capable of generating* Expressions (i.e. products), Concepts, Aesthetic measurements, and Framing information (read backwards, the initials spell FACE). The IDEA model describes instead the impact of the system during its lifecycle, starting from the developmental stage and ideally reaching a stage where it can perform some kind of transformational creative processes.

These assessment frameworks are limited in the possibilities they offer, and a common criticism is that the assessment comes from the author of the system, making it biased. Nonetheless it is useful to frame the capabilities of a system and to reflect on the degree of automation in creativity it has reached, even just for development purposes. Indeed, an extension to the FACE/IDEA framework was proposed to consider the creative abilities of different versions of a same software (Colton et al., 2014) to make it easier for a developer to understand how the creativity of the system is progressing.

2.1.3.3 | Quantitative Metrics

In order to compare the results of different systems in terms of creativity, and to give more scientific indications of the effectiveness of CC applications, it is desirable to have objective metrics that can indicate how creative a system is. Designing such metrics is no easy task, but many efforts have been made towards this goal.

Ritchie (2001) proposed a set of criteria for the evaluation of creativity based on the Product perspective, judged according to *Value* and *Typicality*. The latter is a concept strongly related to Novelty, but is based on the fact that an "inspiration set" (the corpus used by the system) is available, and used to define what is more or less typical. These two basic features must be measured according to some *rating scheme*, and can then be used to compute a set of parametrized criteria, that are basically functions over the the Value and Typicality. In his proposal Ritchie described these criteria as either satisfied or not satisfied (if a certain treshold is reached), but often these were applied as a continous scale rather than a boolean one. Extending this evaluation framework Pease et al. (2001)

focused on the measurement of Value and Typicality, while Colton et al. (2001) investigated the effects of fine-tuning the input knowledge. Ritchie (2007) presented an updated version of his criteria, commenting those researchers that have used it as a means of evaluation, but the presence of many parameters to be tuned makes it difficult to use for comparisons between different systems.

While Ritchie's criteria are the main metrics for the evaluation of creativity, there are other metrics in literature that can be relevant for CC systems, although they do not evaluate directly the creativity of the systems. Galanter (2012) made a review of metrics and methods to evaluate aesthetic value of computer generated artefacts, that is a vital part of many CC systems. Shaker et al. (2016) focuses on procedural content generation, and describes how it is possible to give a visual indication of the capabilities of a system in terms of the variety of products it can generate. To the best of my knowledge, this system has never been used for CC systems, despite the fact that the representation of the space of possible outputs generated by a system has strong links to Boden's theories (which in part inspired Shaker's work). Possibly, these graphical representations could give a good indication of whether a system uses mere combinatorial creativity or is capable of going beyond that limit.

2.1.3.4 | Evaluation of Generated Music

The evaluation methods that were presented in the previous paragraphs are general enough to be applied to musical generation as well as to other CC applications. The following methods focus instead solely on the evaluation of MGSs.

Eigenfeldt et al. (2012) used a concert setting to evaluate a variety of MGSs, and a similar event is described by Sturm et al. (2019). In both cases, the evaluation in itself was performed via a questionnaire given to the audience of the concert. This approach can be extended by turning the concerts into music competitions, as has been done for computer-generated expressive performances of human composed music (Katayose et al., 2012). If the program includes both human and computer generated music, this approach becomes similar to the ones inspired by the Turing Test, but a concert setting is one of the most

natural ways to experience music, and could fatigue the evaluators less than a laboratory setting. Two major limitations of this approach is that the audience will evaluate music according to their personal taste, rather than assessing creativity, and that this evaluation method can only be used to compare the pieces that are included in the concert: comparing different concerts could induce unwanted bias due to different performers, venue and setting in general.

Another useful contribution is that of Yang and Lerch (2018), who argue that while creativity cannot be assessed without a human evaluator, it is useful to use *formative* metrics to describe how well computer generated music fits a musical genre, in order to help the development of the system towards "human-like" music generation. To that goal, many quantitative metrics are presented, and data visualization techniques are suggested. While this does not solve the ultimate goal of the evaluation of creativity, it is nonetheless an useful addendum to the evaluation toolbox.

An overview of the current methods for the evaluation of MGSs was made by Agres et al. (2016). This contribution provides both motivations and tools to evaluate in different manners systems that are merely generative, systems that allow for feedback, and systems that are capable of some kind of self-reflection. Moreover, a distinction is presented between internal and external evaluation, the first being necessary for the functioning of the system and the latter being the usual *a posteriori* evaluation to understand the effectiveness of the system.

2.2 | Music Generation Systems

This section reviews some work describing Music Generation Systems (MGSs), trying to give a picture of what the current advancement of the state of the art is in creating systems that can creatively generate music.

2.2.1 | Methods for Music Generation

The many algorithms that can be applied to music generation can be grouped into some main categories, but the reviews available in literature give different categorization. The following are a summary of the categories found in literature:

- Markov Chains;
- Formal Grammars;
- Rule/Constraint based systems;
- Neural Networks/Deep Learning;
- Evolutionary/Genetic algorithms;
- Chaos/Self Similarity;
- Agents based system.

In the following sections, I will describe each of these approaches by citing work that implemented MGSs with these techniques.

2.2.1.1 | Markov Chains

A Markov Chain is a special Stochastic Process, i.e. a sequence of random events dependant on a time variable, that has a finite number of states, and the probability of the next state is only dependant on the current state (Brémaud, 2013). In practice, a Markov chain is described by a transition table, where rows and columns represent the states, and every cell (x, y) represents the probability of going from the state x to the state y. Since each row represents a probability distribution, the sum of all the cells in a row must be equal to 1.

If the last n states are used to determine the probability of the next state instead of just the last one, this is called n-th order Markov chain. These can be represented with a single transition matrix as well, by constructing an equivalent first order Markov chain having A^n rows, where A is the number of states in the n-th order chain.

Due to their sequential nature, Markov chains are well fit to describe melodies, seen as a sequence of notes. The simplest way to implement a melody-generating Markov chain is to use a set of notes as the possible states, and to compute the transition probabilities between these notes by counting the occurrences of each transition in a given corpus to create a first order Markov chain.

This is what was done in one of the first MGSs ever described. Pinkerton (1956) created the "Banal Tune Maker" by analyzing the transitions of 39 nursery tunes by hand to create a transition matrix. The states used were the seven notes of the diatonic scale of C major (only one octave was considered), plus one extra symbol to indicate rests or notes that are prolonged over a beat. In this case the states of the chain only contain pitch information, requiring the use of other strategies to implement the rhythm. In this case, all the notes were kept to the same duration, and the extra symbol was used to introduce rests in the generated music. Of course, other approaches are possible, including implementing another Markov chain to handle durations.

The basic assumption underlying this simple approach, i.e. that the next note is only dependant on the previous note, is very flawed and only lead to musical results of little interest. Pachet (2002) used a more refined approach in the "Continuator". He implemented a variable order Markov chains using prefix-trees to handle sequences of varying length (as opposed to n-th order Markov chains that will always consider n states) and also used a hierarchy of reductions: the system analyzed in a single chain pitch, duration and velocity, but was able to ignore some information when analyzing new input and comparing it to the learnt sequences. This was especially important in the Continuator because, as the name suggests, it was meant to listen to a musical input and continue it in real time. Being able to ignore part of the learnt information allowed the system to interact with previously unmet input, and to consider musical structures at various levels of detail.

Hiller Jr and Isaacson (1958) used a different approach in their "Illiac Suite". In their fourth experiment, they used Markov chains to generate sequences of motions and progressions rather than sequences of pitches and durations themselves, thus using the model to organize the notes at an higher level. The same idea of organizing higher structural levels

via Markov chains was used more recently in the GEDMAS system (Anderson et al., 2013), whose goal was to generate Electronic Dance Music. To do so, a series of Markov chains were used to choose the general form of the song (i.e. a sequence of sections, each section being 8 bars long), to fill each section with a chord sequence, and finally to generate melodic patterns.

2.2.1.2 | Formal Grammars

Chomsky (1957) introduced the concept of Generative Grammars, a tool for the analysis of natural language that became extremely influential in linguistic studies. The same idea was applied to musical studies, most notably by Lerdahl and Jackendoff (1985), who tried to design a Generative Grammar for the description of music starting from music analysis concepts introduced by Heinrich Schenker in his book "Free Composition" (Schenker, 1935), that well fit the concept of *rewriting rules*, that is at the basis of Chomsky's grammars.

A Generative Grammar is composed of two alphabets: terminal symbols and non-terminal symbols (or variables). A set of rewriting rules is given over the union of these two alphabets, that allow to transform variables into other symbols (both variables and terminals). The generated language is the set of all the strings of terminal symbols that can be obtained starting from a special variable chosen as starting point (usually called S) and applying any number of rewriting rules in sequence.

Grammars can be seen both as an analysis tool and as a generative tool. For example, Steedman (1984) compiled a Generative Grammar to describe Jazz chord sequences: Pachet (2000) describes a system that is in part inspired by Steedman's analysis to tell apart blues songs and non-blues songs, while Chemillier (2004) implemented Steedman's grammar creating a software for music generation.

Chord sequences can be very easily encoded as symbols, but, if an adequate alphabet is given, it is possible to use Grammars to generate any kind of musical information. Hamanaka et al. (2007) describe a system for the automatic analysis of scores based on Lerdhal and Jackendoff's Generative Theory of Tonal Music (GTTM), formalizing in details a grammar to describe musical material. This was then used to create variations on melodies by altering

the derivation trees (a graphical representation of the applied rewriting rules) (Hamanaka et al., 2008). Quick (2011) implemented a software to generate three voice harmonies using a Grammar derived from Schenkerian theory.

L-systems (Lindenmayer Systems) are a variant to Generative Grammars that has been used for music generation. Their main difference from Grammars is that they implement parallel rewriting, thus applying all the rewriting rules at once instead of only one at a time. This characteristic makes these system less apt to sequential data, like simple melodies, and have been used to generate stunning visual effects. When applied to music generation, the most common approach was to map visual data generated by L-systems either to score information (Prusinkiewicz, 1986; Nelson, 1996; Mason and Saffle, 1994) or to arrange a sequence of musical segments (Langston, 1989; Supper, 2001).

Another related approach is that of Transition Networks: finite state automaton that can parse languages similarly to what Generative Grammars do. The most notable example of Transition Networks applied to MGS is that of David Cope's Experiments in Musical Intelligence (EMI) (Cope, 1991, 1992). His approach was to use pattern-matching algorithms to analyze "signatures", short musical sequences that define the style being analyzed, and to determine when and how to use those signatures. After the analysis phase, the collected information is encoded in a Transition Network that is then used to generate new music in the style of the composer that was analysed. While the results are sometimes impressive, they are arguably not very creative, since they just reuse material taken from the learnt corpus (Wiggins, 2007a). Possibly, this is one of the reasons why there is not much research on Transition Networks for music generation beside Cope's work.

2.2.1.3 | Rule/Constraint based systems

Music theory traditionally describes rules that help that guide the compositional process. While composers regularly break those rules, it should come to no surprise that those rules have been used to implement MGSs since the early days of Algorithmic Composition, like in the first two movements of the Illiac Suite (Hiller Jr and Isaacson, 1958).

The inclusion of rules can be implemented in many ways, for example as a final validation

step, or to refine intermediate results. One natural way to implement rules in a MGSs is to use *Constraint Programming*, whose declarative nature is well fit to describe music theory rules. A survey on Constraint Programming used to model music theory (not only with the goal of generation) was made by Anders and Miranda (2011).

One of the most influential researchers within the scope of music generation through constraint is Ebcioğlu, who first implemented rules of fifth-species counterpoint into a Lisp program, and later implemented a custom logic language (BSL) that he used to create CHORAL, a system for the generation of Bach-like chorales that uses some 350 rules for the generation of melodies and harmonization (Ebcio\v{g}lu, 1988, 1990). The difficulty of designing such a system lies in the complexity of explicitly coding a sufficient amount of rules, many of which often do not have a formal definition in musicology literature. Moreover, there is a tradeoff between adding more rules to obtain results that better fit the style that is being modeled and leaving less constraints to be more open to different styles of music.

Constraints can be used to model more abstract features, rather than explicit music theory rules: Herremans and Chew (2016b) defines a way to describe tension in musical pieces based on a geometric model of tonality called the *Spiral Array* (Chew, 2014). Herremans and Chew (2017) used that tension model in a MGS that is capable of generating new music following the tension pattern of an input piece, by first generating random notes and then applying optimization methods (in particular, Variable Neighbourhood Search) to change the notes in order to satisfy constraints defined by the chosen tension model.

Techniques for optimization such as integer programming can be useful as a selection technique when more than one possibility is available. For example, Cunha et al. (2018) describe a MGS that creates guitar solos by concatenating guitar licks. This approach is somewhat similar to a transition network, but in their implementation the concatenation of any two licks had a defined transition cost, and through a branch-and-cut algorithm it was possible to compute the optimal solo. The computation of transition costs was in itself another example of integration of rules: in that work eight rules were described to assign the transition cost between licks.

2.2.1.4 | Neural Networks/Deep Learning

The increased computational power of computers, and the widespread of general purpose GPU programming, made deep learning techniques very popular in the latest years, with applications that span from natural language processing, to image and video editing, to, of course, music generation. The survey by Briot et al. (2020) is specifically focused on these techniques, and gives an exhaustive overview of how machine learning has been used in MGSs.

While the interest in these algorithms grew exponentially in the last decade, the first MGS to use Artificial Neural Networks is that of Todd (1989), who used a three-layered Recurrent Neural Network (RNN) to generate monophonic melodies. RNNs reuse the results of the computations from previous steps when new input is given, allowing them to encode temporal development. This is of vital importance when generating melodies, thus RNNs are a typical approach for MGSs that use deep learning. Nonetheless, there is also room for standard feed-forward networks: Lewis (1991) trained a network with musical patterns ranging from random to well-constructed, to learn a measure of "musicality" that is then used by his MGS to select pleasant compositions.

As already mentioned, RNNs are a popular choice for music generation. In particular, LSTM (Long-Short Term Memory) (Hochreiter and Schmidhuber, 1997) are a special variant of recurrent networks that use special gates to decide the amount of information that is taken from novel input and what is maintained from older inputs. This control over the data flow allowed LSTMs to be both more efficient and effective than standard RNNs in a wide range of applications, and have been used for music generation as well. The first music generation LSTM was applied to blues improvisation (Eck and Schmidhuber, 2002b,a). Traditional music was instead the focus of *folk-rnn* (Sturm et al., 2016), that analyzed over 20000 pieces in textual (abc) notation. A more advanced approach is used by DeepBach (Hadjeres et al., 2017), that generates chorales in the style of Bach (whose chorales made the training set for the software) using two LSTMs, one going forward and one going backwards in time, together with one feed-forward network to consider contemporaneous notes. The results of these networks is then handled by a final feed-forward network that joins the

results in final piece. The rationale behind this choice is explained by the goal of generating *counterpoint*, which requires knowledge of both the previous and the following notes. This gives an example of how it is possible to design complex architectures using many layers of Neural Networks, but the complexity Comes with a price in terms of computational time.

Another deep learning approach that is of great interest to CC is that of Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). The idea behind this method is to train two networks at the same time, one that generates artefacts imitating what is learnt from real-world examples, and the other trying to discriminate between real and imitated artefacts. As one gets better, the other must get better as well in order to "beat" the other network (thus making them "Adversarial"). The two networks can be simple feed-forward networks, but these are not the usual choice for music generation. For example, the eloquently called C-RNN-GAN (Mogren, 2016) uses RNNs (in particular LSTMs) in a GAN architecture to generate polyphonic music. MidiNet (Yang et al., 2017) uses convolutional layers instead: Convolutional Networks are trained to reduce the dimension of the input, usually starting from bidimensional input. This approach is often used on images, so when applied to MGSs the input of the Network is often some graphical representation of music, such as piano rolls. It is important to be aware that while images have two dimensions that are equivalent (both represent displacement in space), graphical representations of music show two non equivalent dimensions, usually pitch and time, possibly leading to less reasonable results (Briot et al., 2020).

2.2.1.5 | Genetic/Evolutionary algorithms

The general idea behind Genetic (or Evolutionary) Algorithms (GA) is that, starting from a population of random solutions to a problem, it is possible to combine those solutions to obtain new solutions, and by selecting the ones that better answer the problem it is possible to get closer and closer to the optimal solution to the original problem. Thus, to solve a problem via GA, it is necessary to have (Sivanandam and Deepa, 2008):

1. The ability to generate random but suitable solutions to the problem as a starting population;

2. A way to evaluate the "fitness" of a solution;

3. The ability to mutate and recombine those solutions.

In the field of music generation, the points 1 and 3 are for sure available (once a representation of musical material is chosen), but it is hard to evaluate how good a solution is (as already discussed in Section 2.1.3). It might be difficult even just giving a precise definition of what the problem is. Nonetheless, Genetic Algorithms have often been used to implement MGSs.

Possibly, the most famous Genetic MGS is GenJam, designed by Biles (1994). The system is meant for Jazz improvisation, where a human player interacts with the software that outputs both the pre-made musical base and solos generated on-the-fly by evolving the human improvisation it has just listened to. Originally, the fitness function was implemented by having a human decide if the output was good or bad, an approach that is usually referred to as "Interactive Genetic Algorithm" (IGA). This generates a bottleneck for the system, as a lot of human intervention is required. A successive version (Biles et al., 1996) used and Artificial Neural Network as a fitness function, but it lead to unsatisfactory results. In the end, the author resolved to completely eliminate the fitness function (Biles, 2001). Basically, the algorithm retains the ability to mutate and compose licks, an ability that is used to respond to musical input in a way that incorporates the human improvisation without being a mere copy, but since there is no more evaluation of the fitness, GenJam is no more a GA.

GenJam passed, through his versions, some of the most common approaches to the definition of a fitness function. Another approach is to use rules taken from music theory to design a fitness function. This is the approach chosen by Phon-Amnuaisuk et al. (1999). In that case the goal was the harmonization of a given melody, and the fitness function incorporated rules of harmony describing forbidden and preferred intervals and motions. In this case, the GA becomes a way to explore a space of possibilities described by the chosen rules. One might wonder if this is better or not than just generating samples following those rules, as described in the previous section. Indeed, Phon-Amnuaisuk and Wiggins (1999) found that their GA was outperformed by a rule-based system using the same set of

rules that were incorporated in the fitness function. The authors argue that having explicit control over a system's knowledge will lead to better results and more powerful means of exploration: while the authors do not scorn GAs in general, it seems that this approach cannot give such explicit control over the knowledge of the system, and thus other systems should be preferred when explicit knowledge is available.

GAs offer many other forms of hybridization, since the representation used by other algorithms can be evolved through a GA. For instance, it is possible to evolve the rules of a grammar (de la Puente et al., 2002), or to evolve the parameters of a Markov Chain (Werner and Todd, 1997; Bell, 2011) or of a Cellular Automaton (Lo, 2012). I already mentioned that rules, Neural Networks and human assessments can be incorporated in the fitness function for a Genetic algorithm. It is worth mentioning that Markov chains have been used for the same goal (Lo and Lucas, 2006). Markov chains can also generate the initial population, obtaining starting point that is better than random, possibly leading to convergence to good solutions with fewer generations (Manaris et al., 2011).

2.2.1.6 | Chaos/Self Similarity

Musical compositions show some degree of self similarity, both in the musical structures and in its spectral density (Hsü and Hsü, 1991), roughly following a $1/f$ distribution, at least for pieces that are deemed pleasant to listen to (as opposed to random compositions) (Voss and Clarke, 1978).

Starting from these considerations, fractals and other self-similar systems have been used to generate musical material. The results of such systems are usually not regarded as a final output, but rather as an inspiration for human composers (Bidlack, 1992). Another approach is to generate self similar structures rather than directly generating self similar melodies: Leach and Fitch (1995) generated tree structures like those described by Lerdahl and Jackendoff (1985), by tracing the orbit of a chaotic system, and mapping the computed values to different hierarchical levels of the tree.

Another approach is to use Cellular Automata (CA), dynamic systems composed of many cells, whose states are updated at discrete times using a set of transition rules. One

famous example of CA is "Game of Life" by Conway (1970). Like other fractal systems, CAs tend to generate melodies that are not too pleasing, and often need further human intervention. CAMUS is a MGS that is based on two different CA, whose cells were mapped to sequences of notes and to different instruments (Miranda, 1993). A later version used a Markov chain to specify rhythm, but despite the effort to create a full MGS, the authors still admit that the results can often be not very pleasing, but can become interesting "for the composer who is prepared to put a little effort into the system" (McAlpine et al., 1999). Miranda (2007) later argued that CA are more effective for sound synthesis, rather than for MGSs.

These systems are arguably not interesting to AI practitioners, since the decision making is based upon chaotic and random processes, but were included for completeness. Nierhaus (2009) provides a good review of these approaches, that are given less consideration by later surveys.

2.2.1.7 | Agents based systems

A software agent is an autonomous piece of software with perception and action capabilities. Any software with such capabilities can be seen as an agent (including many of the systems described in the previous sections), but the definition becomes especially interesting when multiple agents cooperate within a single software, that can be referred to as a Multi Agent System (MAS). This is not a specific algorithm for music generation, but rather a meta-technique that has gained popularity among researchers, as testified by Tatar and Pasquier (2019).

The use of agents in MGSs makes it easy to model certain musical behaviours. Voyager (Lewis, 2000) is a MAS that has 64 player agents that generate melodies according to one of various pitch generation algorithms written by the author, according to his own taste, and a behaviour model that describes the general timbre, tempo, pitch range and other features that regulate the development of the piece. This models a band where everybody is improvising, but still follows some general agreement. Lewis has played together with Voyager, both in recordings and live: in this setting one can also consider the human

performer as one additional agent of the system.

MASs are also useful to model social interactions: once each agent is given specific characteristics (one could say, a personality), the interaction between different agents can take into account the difference in their characteristics, either in a conflict or in an agreement. For example, Kirke and Miranda (2011) introduces a system later called MASC (Kirke and Miranda, 2015) where each agent has a specific "emotion" and the ability to express it by "singing" to another agent. The other agent will be affected by the mood expressed by the singer, adapting his own internal state. Moreover, their internal state also defines if the listener will "like" the song, incorporating it into his own song.

Taking further the same idea, the agents can implement cognitive models that regulates their interaction with the others. One such model is the Belief-Desire-Intention (BDI) Architecture. For instance, Navarro et al. (2014, 2016) describe a MAS with the goal of generating harmonic sequences, where two particular agent, the *composer* and the *evaluator*, have beliefs based on music theory and desires (one to compose and the other to evaluate the generated composition). The intentions are represented by the algorithms implemented to apply and verify the theoretic rules that form their beliefs, and are influenced by the communication between the two roles.

2.2.2 | Open Challenges in Music Generation

To define a list of the open challenges in Music Generation, the following reviews published in the last fifteen years were considered:

- Nierhaus (2009)
- Fernández and Vico (2013)
- Williams et al. (2015)
- Herremans et al. (2017)
- Lopez-Rincon et al. (2018)
- Tatar and Pasquier (2019)

- Briot et al. (2020)

The challenges and directions that were described by the authors were analyzed and groped as described below. For each of the identified challenge, some examples of how these problems can be faced are cited, discussing how these are related to Creativity in general, by categorizing these challenges using the dimensions of creativity described by Jordanous and Keller (see section 2.1.1.6).

2.2.2.1 | Control

Control refers to having the possibility to choose specific features that the output of the MGS will exhibit.

Having control over certain features of the output of a MGS can be, depending on the used algorithm, trivial. But, with more data-driven approaches like machine learning, it becomes less obvious what can be done to affect the output. It is not surprising that this issue was only mentioned in a review focused on deep learning.

Since data-driven approaches are meant to learn features from their input, one simple way to influence the features of the output is the selection of the training set. This approach is to some extent used by every corpus-based system, knowing that learning on Folk music will be very different from learning on Bach chorales. The problem with this approach is that it does not allow a good granularity of control, and any change on the input would require retraining the system, a task that can be very time consuming.

The same idea is applied in a slightly different fashion by Ekeus et al. (2012). Their approach was to generate a set of randomly sampled Markov chains, which were evaluated with an approach based on Information Theory. These were employed in a MGS that allows the users to select a point in a triangular space where the vertices represent periodicity, repetition, and noise. The chosen point is mapped to the features that were evaluated for each Markov chain, and the most appropriate one is selected and used for melody generation.

The same approach can be used in Neural Networks by altering the parameters that make up the network, but this can be much more intimidating, due to the excessive number of parameters involved and the difficulty of understanding their meaning (Sturm, 2018).

A way to obtain this is proposed by Kaliakatsos-Papakostas et al. (2018), who used a recurrent network trained on a small dataset (made of only three pieces) that was augmented specifically to address the features the authors wanted the user to be able to manipulate, in order to study how the parameters are affected, and to be able to alter them accordingly in the generation phase.

Control is related to the creative dimension of "Active involvement & persistence" which suggests that the creative agent is in control of the generation process. Using deep learning to achieve this can be extremely hard, although many advancements in this direction are being made. We suggest to use techniques that allow for simpler tuning over the features one wishes to control, by either using appropriate representations (see section 2.2.2.5) or by explicitly limiting those features with rules. Machine learning can be used in conjunction with these approaches to ensure other creative features, such as "Variety, divergence & experimentation".

2.2.2.2 | Narrative Adaptability and Emotion

Narrative Adaptability refers to the capability of the MGS to convey a sense of development (Narrative) in the generated music, giving a more complex meaning to the piece. *Emotion* refers to the capability of the MGS to convey specific emotions with the generated music.

These two are treated together because it is possible to convey different emotions in different sections of the piece, one of the main aspects of *Narrative Adaptability*. Both of these can be seen as a special instance of *Control*, where the features that are being controlled relate to emotional aspects or to specific events of the narration. This is especially relevant in non-linear media (like video games) where the Narrative must adapt in real-time to the events in the media.

The study of Emotion in music has a long history (Juslin, 2010) and, as can be seen in the review by Williams et al. (2015), has often been considered in MGSs. Narrative Adaptability is less commonly found, despite the fact that such adaptability is something that human composers could never achieve without the help of a computer, making it an interesting field of investigation. Ventura et al. (2009) present an installation implementing

a typical architecture for emotion-aware systems: an emotion (expressed as values in the valence/arousal plane (Hunter and Schellenberg, 2010)) is detected (in this case by analysing the movements of the users via webcam) and then used as the input for the MGS. To do so, some features that are known to be related to emotional expression are manipulated, such as tempo, pitch range and loudness (Oliveira and Cardoso, 2007). A similar architecture is used by Scirea et al. (2018) to add music to Checkers: the MGS analyzes the board to understand how risky the situation is for the player, and then generates music that emotionally expresses the level of risk.

Mezzo by Brown (2012a,b) uses a different approach that takes its roots in classical music: the use of *Leitmotifs*. In a video game setting, some characters and situations are given a theme (composed by a human), and when those are encountered in the game, a message is sent to *Mezzo*. This will use the themes triggered by the messages, blending them together to generate a music that expresses the current situation. Similarly, when music is used within human-computer interaction, is is useful to detect musical features in the human interaction to generate music that matches the emotional content as a feedback (Carnovalini and Rodà, 2019b; Carnovalini et al., 2019).

Another approach that does not necessarily involve a full MGS, but can be used to increase its adaptive capabilities, is that of automatically generating transitions between pre-composed sections, to be able to connect sections without knowing a priori when one will end and one will start (Gillespie and Bown, 2017; Horner and Goldberg, 1991). This is usually applied to human-made compositions, but it could be easily applied to a MGS, as long as it is capable of generating the next musical piece in advance, since it is needed for the transition generation.

These challenges are especially important for creativity, as they address "Intention & emotional involvement" and "Progression & development" and "Social interaction and communication", and in general are fundamental for the affective perception of the machine, which in turn is important to pass Turing-like tests. One research direction we suggest is to study how different expressive features can influence each other, and how to select one specific expressive technique to convey certain feelings rather than altering all the features

that are linked to that emotion, so that systems could be able to generate, for example, a sad piece which is also fast-paced. This can also improve "Variety, divergence & experimentation" of the generated pieces and possibly lead to more "Originality".

2.2.2.3 | Hybridization

Hybridization refers to the use of more than one technique for music generation in a single MGS.

This is the only point of this list that does not concern a quality of the output, but rather a characteristic of the system itself. The need to go beyond a single method for the generation of music was already noted twenty years ago (Papadopoulos and Wiggins, 1999), but the call for hybridization is relevant to this day. The rationale behind this idea is that since there is not a single method that has been proved to be more effective than the others, nor to be capable of addressing all the issues that a MGS must face, it is important to take advantage of different approaches. Nonetheless, using multiple algorithms is obviously expensive for the development, and in general it is hard to witness in the early stages of any project. Moreover, researchers are often more interested in applying a specific technique for music generation rather than creating a complete MGS that would benefit from hybridization.

Some approaches are more prone to being used in an hybrid context than others: we have already discussed how Genetic algorithms and rules or constraints are often coupled with other algorithms, but other approaches are possible. Eigenfeldt and Pasquier (2009) describes how the various versions of the *Kinetic Engine* have used different algorithms for designing agents capable of generating rhythm, melodies, and harmony. In the later versions, agents with different roles interacted with each other to generate both rhythm and melody, and also a Markov chain was used to influence the harmonic progression Eigenfeldt and Pasquier (2010). This gives an idea of how, in an agents-based system, it is possible to delegate different tasks to different agents, that can each implement a different strategy when generating their respective content.

A somewhat similar subdivision of tasks is proposed by Carnovalini and Rodà (2019a),

where the process of composing a melodic phrase is divided in successive steps: generation of pitch succession, generation of rhythm, and finally generation of expressive variations of intensity and timing. Each of these steps follows a different algorithm, but some information is passed on through each step. In particular, all of the steps use information about the importance of each of the generated notes, dividing them with a Schenkerian approach (Simonetta et al., 2018). The authors argue that this idea can be further extended to other tasks (such as form generation and harmony), including both deep learning and classical AI algorithms, trying to find the optimal combination for each task (Carnovalini, 2019).

The use of expressive musical performance generation systems Widmer and Goebl (2004); Canazza et al. (2012, 2015), that are sometimes embedded in MGSs as the one just cited above, can also be seen as a form of hybridization, but will be better discussed in the next section.

We believe that systematic use of Hybridization could be one of the most prolific research direction for CC, since it could help researchers expand different dimensions of creativity using different techniques for each. Moreover, giving a variety of compositional approaches to a software could be seen as giving it a better "Domain competence", and being able to choose between techniques can improve "Variety, divergence & experimentation". Explicitly modelling into the software what different techniques are more apt for and changing behaviour according to user's requests could be an interesting research direction.

2.2.2.4 | Rendering

Rendering refers to the quality of the audio (meant as waveform) that is generated by the MGS.

This might seem redundant, as the quality of the generation is obviously important in a MGS. But in many cases, MGSs only handle symbolic music generation, usually as MIDI or MusicXML files, and the audio is generated with simple software MIDI synthesizers, which are far from giving good renditions of any musical composition.

We already mentioned that it is possible to add expressive performance to a generated piece in order to improve its audio rendering. This is usually done through existing algorithms

that are applied to the music after it is generated. A review of existing algorithms can be found in Kirke and Miranda (2009).

Another way to improve the musical rendering is to use automatic orchestration techniques: rather than having a predetermined instrument to play the generated piece (piano seems to be a popular choice) it is possible to generate musical material that is then assigned to different instruments (Handelman et al., 2012) or to have a set of possible instruments from which to choose from and that can intervene at different moments of the composition (Anderson et al., 2013).

Brunner et al. (2018) describe a system that uses both expressivity and orchestration to perform Style Transfer through Variational Autoencoders (Kingma and Welling, 2013). Style Transfer tries to apply a certain style (for example, defined by a certain composer or genre) to an existing musical piece that was not originally meant for that style. This can of course be applied to computer-generated music as well, although we are not aware of any work in literature that has yet tried this approach.

All these approaches generate some sort of variations after the score generation process is over. This excludes any possibility to render audio in real-time, as the generation phase must be over for these algorithms to function. A different approach that has been less explored is to generate music and its expressive variation at the same moment: one example is PerformanceRNN (Oore et al., 2018).

A completely different approach is to model music directly at the audio level, thus implicitly generating the rendering as well. This approach is challenging for many reasons, including computational complexity and the difficulty to capture semantic structures from raw audio. Nonetheless, Dieleman et al. (2018) found that using Autoencoders it is possible to obtain realistic results that remain consistent for tens of seconds, meaning that local structures can be understood and modeled directly from the audio.

One could argue that the creative task we are interested in is the composition, while the rendering is delegated to musicians. While it is true that in some cases computers generate music that is meant to be played by humans, it is more often the case that computers directly play the generated music themselves. Moreover, in the context of evaluation of

CC, having a good audio rendering can influence human evaluators, so it should not be overlooked (Oore et al., 2018; Carnovalini and Rodà, 2019a). More generally, Rendering can be seen as part of "Generating results": while scores are results in themselves, the fruition of music is through sound. Therefore, to add "Value" to the output, Rendering must be considered. We are not aware of any research comparing user preference of computer generated music that is emotionally rendered versus 'deadpan' executions, but that would certainly be an useful contribution to CC research.

2.2.2.5 | Structure and Mapping

Structure refers to generating longer pieces, containing reasonable repetitions and subdivision of different sections, usually recreating some kind of musical form. *Mapping* refers to the problem of handling different representations of music and choosing the most appropriate one for the generation of musical content.

While the first is notoriously difficult for MGSs, the latter is an issue that is often not considered, as usually a certain representation is chosen a priori. There are instead notable proposals in literature that further the possibilities for MGSs using specific representations of music.

Herremans and Chew (2016a, 2017) used a specific data structure, the *Spiral Array* (Chew, 2014), to compute the *tension profile* of a musical piece. This profile is used to generate a new piece that follows the same profile, through constraint programming. Starting from a specific representation for tension structures, the MGS is able to create longer pieces with convincing structure. One might argue that the structure is simply being copied, but a possible extension to this work could possibly generate novel tension patterns using the same ideas.

Representations based on Schenker's or on Lerdahl and Jackendoff's theories are studied, since these can capture different levels of structural information. Most works only have the aim of automatic analysis of musical pieces (Marsden, 2010; Marsden et al., 2013; Hamanaka et al., 2016, 2017), but others have used this approach to generate music that follows a defined structure (Groves, 2016; Carnovalini and Rodà, 2019a)

Other systems approached the problem of structure without any specific representation. GEDMAS (Anderson et al., 2013) explicitly generates structure, seen as successions of 8-bar segments, through a Markov chain. Medeot et al. (2018) describe StructureNet, a neural network that studies occurrences of repeats (either of rhythm or of interval sequences), and that can be be embedded in a larger MGS influencing the generation process according to the learnt structures of repeats.

Structure is strongly linked with the creative dimension of "Progression & development", and can be linked to the challenge of Narrative Adaptability as well. Once again, we suggest to hybridize different approaches, possibly using different techniques at every level of representation to consider the development of a piece at a macro level before considering the local melodic and harmonic content.

2.2.2.6 | Playing Difficulty

Playing Difficulty refers to the ability of an MGS to regulate the difficulty for a human to play the generated music.

This can be seen as a specific instance of *Control*, where the feature that must be controlled is the technical difficulty of the output. This problem only becomes relevant when the output of the MGS is meant to be played by a human, which is often not the case: this might be the reason why this issue has hardly been acknowledged in literature.

The review by Herremans et al. (2017), that is the only one to mention Playing Difficulty, only cites a couple recent works that have considered the issue. One is Sébastien et al. (2012), that designed seven criteria used to estimate the difficulty of a piano piece, in order to suggest pieces to learn to students. A similar approach for guitar is presented in Xambó et al. (2018), based on known chords. Both these systems are not MGSs, but could be implemented as a constraint or as a fitness function in a MGS. Another work is that of McVicar et al. (2014), that generates guitar solos in tablature form. This is indeed a MGS, but it does not really consider the difficulty of the generated solo, but rather uses an algorithm to minimize fingering difficulty, without affecting the generation of the piece. Extending on the same idea, Ariga et al. (2017) created a guitar solo generator that

considers the fingerings as a way to measure and control the difficulty of the generated solos. Nakamura and Yoshii (2018) describes a system that creates piano reduction of ensemble scores, capable of generating reductions with different levels of difficulty based on fingering and tempo information.

On the opposite side of the difficulty spectrum, Pachet (2012) describes a system that can generate virtuoso solos, using variable-order Markov chains trained on a dataset of virtuoso solos. Arguably, increasing the Playing Difficulty of a generated piece is easier than lowering it (without losing musicality), but the work by Pachet was motivated by a study of creativity in solos and not of difficulty itself.

To be able to change the playing difficulty of a piece, one needs to increase the "Domain competence" considering for example the physical characteristics of the instruments that will be used to perform the piece, making this challenge also relevant to the perception of CC. Choosing an appropriate level of difficulty can also improve the "Social interaction and communication", since if one wishes to create a MGS to interact with humans, it is important to tune the difficulty to the end user's ability. One possibility to deepen this relatively unexplored branch is to use published exercises books for the learning of instruments to extract features correlated to the difficulty level.

2.2.3 | Discussion

The sections above describe some open problems in music generation that were identified by other researchers reviewing the state of the art. In addition to the ones listed above, most researchers also include Evaluation and Creativity as open problems that are not completely addressed yet, but these were excluded from this part of the chapter as those topics were already widely discussed in the first half of the chapter.

It is intersting to notice that most of these open problems deal with making generated music more human-like or more useful for humans. Narrative Adaptability, Emotional value, Structure, Mapping, and Rendering all are arguably motivated by trying to make generated music be more similar to music composed by humans, which usually has some kind of emotional value or content, has a reasoned long term structure, and is performed or produced

to sound as good as possible. Control and Playing Difficulty are related to generating music under conditions dictated by the human user, so that the generated music can be most useful to them and according to the motivations they had for wanting generated music in the first place. Hybridization is an outlier in this context, as it regards technical details of the implementation of music generated systems rather than being a feature of the output, but one can easily make the case that hybridization is desirable because it can potentially allow to address other problems as well.

As a result of this study of the state of the art, I decided to focus on the general aspect that encompasses these open problems, that is trying to make generated music as human-like and as useful for humans as possible. This is explored in the following chapters first by implementing an application where human social interaction is enhanced by generated music, and then by studying structural representations of music to obtain better long term generation and thus better imitating human-composed music.

3

Social Interaction and Music Generation

In this chapter, I explore one application where computer-generated music is crucial to allow for new modalities for interaction, giving one example of how Computational Creativity can be useful for human in a human-centric social experience.

3.1 | The Social Nature of Music

Music can be played alone, and solo pieces are often valuable parts of any virtuoso's repertoire. Despite this, music playing is mainly a collaborative activity. Music students spend a lot of time exercising in groups and orchestras, knowing that keeping a shared tempo and shared musical expression is a fundamental skill for a professional musician. In classical music coordination between different players can become so complex that it requires a director to ensure that the musical interactions are effective and aesthetically pleasing.

When there is no written score, as with jazz improvisation, the interactivity of musical playing reaches another dimension. While there may be some agreement before the improvisation, such as choosing an approximate tempo or a fixed chord progression, each musicians needs to pay close attention to what the others are playing to keep the tempo and make meaningful contributions to the session.

Musical and social interactions are fundamental in music-making for non-professionals as well. Even a garage band of teenagers requires the members to collaborate both when

they play and when they stop if they want to make music effectively (Kokotsaki and Hallam, 2007). Interaction also leads to educational benefits, strengthening social skills and self-esteem, and giving musicians a sense of belonging as well as higher satisfaction (Hallam, 2010). The emotional value of music is also a critical factor in the emergence of social benefits (Koelsch, 2015) and music can be helpful even with subjects with social difficulties, for example those suffering from autism spectrum disorders (ASD), that can express feelings through music more effectively than with words (Allen and Heaton, 2010; Quintin et al., 2011).

While playing in groups can realize these social benefits even if the involved players are not professional-grade musicians, reaching an effective interaction and pleasing musical results still requires months (and sometimes years) of practice with an instrument. Not everybody wishes to spend that much time learning how to play, and some cannot even hope to achieve such technical abilities, for example due to motor impairments. Professional music therapists can help their clients obtain these benefits despite the lack of training (Bruscia, 1987). One of the tools available to music therapists is that of improvisation. Even if the client is not a musician, therapists can share musical instruments and create music and sounds together. Alternatively, the client could play a simpler instrument, like a drum, that can be played by anybody to some extent, while the therapist contributes to the improvisation using more complex instruments, such as a piano or a guitar, creating music that follows what the client is doing but is more elaborate and pleasing. Doing so requires the help of a highly trained professional and is therefore hard to practice as a daily and inexpensive activity. Moreover, when applied to children, these improvisation sessions are usually a dialogue between the (adult) therapist and a single child. The therapist could interact with two or more children, but it would be impossible to have a "peer to peer" improvisation between children alone. One option towards this goal would be reducing the quality of the music, which is normally controlled by the therapist. This would mean, for example, using only drums, which are less demanding in terms of musical training. The result would be a rhythmic improvisation, devoid of melodic and harmonic material, but an improvisation nonetheless. This could seemingly be a good compromise since the interac-

tivity is maintained, which is one of the principal features of music therapy. However, the aesthetic value of the interaction is a crucial factor for the therapeutic effects of a musical dialogue (Aigen, 2007; Stige, 1998), and should therefore not be overlooked.

But what are the social implications of music making when this activity becomes shared with computers?

In this chapter, I describe a gaming system inspired by this kind of musical interaction: it requires two players that collaboratively create a rhythm, having the experience of creating a musical interaction, while the system 'augments' the interaction by adding a chord progression and melody. The game is meant to recreate the social benefits of group music-making and music therapy sessions, even for people that do not have musical expertise nor training. I propose this as a serious game since the goal of the game goes beyond mere entertainment (Ritterfeld et al., 2009). In this sense, this contribution falls within the intersection of technology, music, and health/well-being. Using digital technologies to facilitate music therapy interactions and to create personally-tailored experiences is indeed a field which has seen a rise in interest lately, but still has lots of unexplored territory and potentially profitable interdisciplinary approaches (Agres et al., 2021). In our case, the game can be seen as a way in which technology can use music to empower patients to reach therapeutic goals. While these aspects are related to a variety of fields, this thesis is about Computational Creativity. The motivation for studying this specific application of musical creativity is to showcase how automatic music generation, while being performed by a machine, can be extremely human-centric. While some would argue that in general the task of music composition is better left to humans (yet many others would argue against such position), this application shows one example where computer-generated music is vital to open new possibilities for interaction that would not be possible with pre-made human-authored compositions.

The rest of the chapter, after a brief discussion on related work, will first introduce the technical implementation of the tempo tracking system which is at the basis of the game in Section 3.3, and then the gaming and therapeutic aspects will be described in Section 3.4.

3.2 | Related Work

Music is a fundamental element of multimedia applications, but systems that use music and rhythm as the main element of interaction are rarer. Some notable examples are commercially acclaimed video games that focus on rhythm.

The most common way in which rhythm is used as a game mechanic is to require the player to perform a series of actions at the right moment, following a predetermined sequence that is shown to the player aligned with a musical soundtrack. The actions can vary from dance moves (as in *Dance Dance Revolution* (Konami, 1998) or *Just Dance* (Ubisoft, 2009) or clicking buttons on a keyboard or touchscreen (*Frets on Fire* (Unreal Vodoo, 2002) or on special controllers (usually shaped like musical instruments, like guitars in *Guitar Hero* (Harmonix Music Systems/RedOctane, 2005), drums in *Rock Band* (Harmonix Music Systems/MTV Games, 2007) or *Taiko no Tatsujin* (Namco, 2001) or even DJ consoles in *Beatmania* (Konami, 1997). In all of these games, the rhythm is predefined and follows the chosen songs. The player can influence the musical output as making correct moves or mistakes can trigger visual and aural feedback, but the song does not account for improvisation nor adapts to the player's performance. Other games have made different use of rhythm by forcing a rhythm on otherwise "normal" game mechanics. *Otocki* (SEDIC/ASCII Corporation, 1987) is a shooter game and one of the first games to have a procedurally generated soundtrack. Each time the player shoots a bullet, a note is randomly generated using rules that ensure the note fits the background accompaniment. Since the notes must fall on beats of the background music, the bullets are also shot only on those beats, resulting in delayed shots if the player does not follow the rhythm. An even stricter imposition of rhythm is found in *Crypt of the NecroDancer* (Brace Yourself Games/Klei Entertainment, 2015), a fast-paced turn-based RPG where turns are tied to the soundtrack's beats, meaning that if the player fails to act within a beat, the turn is lost. An interesting feature of this game is that the soundtrack is fully customizable by the user: a tempo and beat detection algorithm is used to align the gameplay to any music file the user selects from his personal library. This feature makes the game adaptive to the user selection, but once the music is selected it is still the player that must follow the rhythm of

the game and not vice-versa as I wish to do with the proposed system.

Rhythm is also used in academic research to create serious games. One application is to use rhythm games to teach the sense of rhythm, which is usually learned early in infancy and strongly influences auditory processing and speech learning (Santolin et al., 2019). In particular, infants with hearing impairments or neurological disorders do not learn rhythm as effectively as others. This makes the use of serious games that help to build a rhythmic sense useful to support their educational and neurological development. One such game is *Rhythm Workers* (Bégel et al., 2018), where the player must either tap a touchscreen following the beat of a song or recognize whether a percussive sound is aligned to the background music's beat. This game is designed for children with neurological impairments but requires good hearing since the rhythm is presented through sound. Another mobile game called *El Misterio de Armonisia* (Pérez-Arévalo et al., 2017) uses both sound and visual cues as it is intended for children who have hearing impairments. The player must tap certain areas in the touchscreen when a symbol enters those areas, and the symbol's movements are aligned to the musical cues, similarly to more famous games like *Piano Tiles* (Umoni Studio, 2014).

Another context in which rhythm games were used in serious context is that of motor rehabilitation. Rehabilitation of fine movements requires repeating certain movements many times, a task that can quickly become tiring making the patients less willing to do the prescribed exercises, hindering the benefits of the therapy (Fujioka et al., 2018). One serious game with the goal of overcoming this difficulty (Shah et al., 2019) uses the Leap Motion Controller, a sensor made of two infrared cameras, to detect hand gestures and use them as input for a rhythm game that is similar to many games described above, but in which the notes are triggered by gestures. The presence of the music (to which the notes are aligned) and the challenge of the game makes the experience more pleasing and less tiresome. Other similar proposals using other motion capture systems exist in the literature (Agres and Herremans, 2017).

All the applications described above are games that use some kind of prerecorded music to which the gameplay is aligned, rather than allowing the players to improvise and use

their own pace. One application that is more similar to ours is *D-Jogger* (Moens et al., 2014), which detects the user's walking or running pace and warps the music they are listening to make the musical tempo match the movement pace. This application has the goal of making an exercise session more pleasant by matching the tempo of the music with the user's pace, making it similar to the therapeutic goal proposed here. Despite the similarity, the strong simplification of the rhythmic features considered (i.e., time between steps) in their system makes the tempo detection algorithm they implemented not useful for a music improvisation application. One application that deals with full improvisation instead is *Wiimprovisation* (Benveniste et al., 2009), which uses Wii controllers to allow the players to create music collaboratively. In this case, there is no background music, but the players can move their controllers as if they were drumsticks to produce sounds. A few different sounds are available by pressing different buttons on the controllers, and different instruments can be selected to allow the players to use different sets of sounds. The serious goal of this game is to provide mediation and improve social interactions in children suffering from behavioral disorders, making this proposal the most similar to ours. The main difference, in this case, is that the music is entirely generated by the players, who have access to very limited sounds and expressive potentialities, making the resulting music arguably less aesthetically pleasing than the augmentation proposed in this paper. I am not aware of other work in literature that have the aim of augmenting a rhythmic improvisation in the context of a serious game to make it more aesthetically pleasing.

Without considering the gaming aspects, the main technical requirement of the proposed system is being able to understand and follow a rhythm established by its users, synchronizing musical events generated by the software with those performed by the players.

Much work within Music Information Retrieval has focused on detecting rhythm meant as tempo and meter. In that case, there is no interest in retrieving such information in real-time, but is rather used to label musical data offline, for use in retrieval tasks at a later time. Despite the abundance of works addressing this task, It is worth noting that this task is usually applied to audio files and audio datasets (Jia et al., 2019; Quinton, 2017; Quinton et al., 2016) and, more recently, to multimodal files including audio and video (Itohara et al.,

2012; Ohkita et al., 2015) or other additional data (Benning et al., 2007). Far less research has been dedicated to symbolic music. This is partly due to the fact that symbolic music often incorporates tempo and meter markings within the files, making the task trivial. On the contraty, in the application that is the core of this chapter, we find ourselves in the situation where symbolic music is being generated in real time, and thus meter and tempo annotations are not available, meaning that we must focus mainly on the relatively little relevant literature that covers tempo and meter estimation in a symbolic context.

The earlier methods dealt with non-expressive quantized tempo files, which is common for scores and computer-generated files, but does not reflect actual human performances. Steedman (1977) analyzed inter-onset intervals to find most plausible beat durations, and analyzed accents to detect meter. Dixon (2001) applied similar ideas to recorded human performances in symbolic format, a context more similar to ours, by clustering groups of similar inter-onset intervals to account for variations due to human expressivity. A different approach is taken by Frieler (2004), who analyzes the discrete temporal events that constitute symbolic files by transforming them into a continuous function, and then analyzing the autocorrelation of the signal to find viable beat candidates. Whiteley et al. (2006) takes the same approach further by adding the concept of a bar pointer, an ideal cursor over the musical data which enables a bayesian approach to represent metrical and temporal expectations. A Bayesian approach was also used by Cemgil et al. (2000), but by estimating tempo through Kalman filters operating on a Tempogram, a wavelet based representation of the performance.

For completeness, the following are some methods used for tempo estimation in audio. When dealing with aural features different challanges emerge, mostly due to the fact that musical events have no explicit temporal labeling in audio files. One approach would thus be the use of techniques for onset detection by using spectral features, followed by the study of the periodicity in the onsets following some of the approaches outlined above, such as autocorrelation of the signals (Gouyon and Herrera, 2003; Alonso et al., 2004). This approach can be taken a step further by using specialized regression approaches to determine the tempo based on the found autocorrelation values (Eronen and Klapuri, 2010).

The current state of the art for audio tempo estimation is less adherent to the techniques which could be used for symbolic music, leveraging deep learning techniques. In particular, Convolutional Neural Networks are employed in this field to deal with the high dimensionality of an audio signal to infer the tempo information with high success rate (Schreiber and Müller, 2018; Böck and Davies, 2020; Durand et al., 2017).

As stated above, those methods were not meant to work in real time. Requiring the tempo estimation system to work in real-time is more frequently considered by research relating to score-following and automatic accompaniment (Muller et al., 2004; Dannenberg, 1984; Raphael, 2002), but once again these often consider aural features that are not available in the context considered here. Those works consider MIDI instruments instead only have access to symbolic information, as in the case of the system presented here. Those implementations that focus on following a human improvisation best cover the requirements, but often make assumptions on the input performance, requiring some prior knowledge before the improvisation starts. One example is the MIDI Accompanist by Toiviainen (1998) that uses oscillators to adapt to a non-predefined beat. This work has a complicated mathematical basis, and it only focuses on the adaptation to a beat, not considering the meter. Beatback (Hawryshkewich et al., 2010) offers another approach to follow drummers in their improvisation, but the synchronization of the system is based on having the tempo decided before the improvisation, thus making it not adaptive. A similar approach is that of B-keeper (Robertson and Plumbley, 2007), which analyzes the audio of a kick drum to perform synchronization, but again requires the prior knowledge of the approximate tempo and does not consider meter. The improvisation follower by Xia and Dannenberg (Xia and Dannenberg, 2017) is instead based on melody but is limited to improvisations over a given melody, and the system needs to be trained over examples of the target accompaniment. Other famous systems that are capable of augmenting human improvisation include Pachet's *Continuator* (Pachet, 2002) and Biles' *GenJam* (Biles, 2013b), but both systems require a fixed tempo (GenJam also requires a chord progression and a scaffold of the song structure), and the musical augmentation works by exchanging musical solos with the user, and not giving a contemporary musical background to the user improvisation.

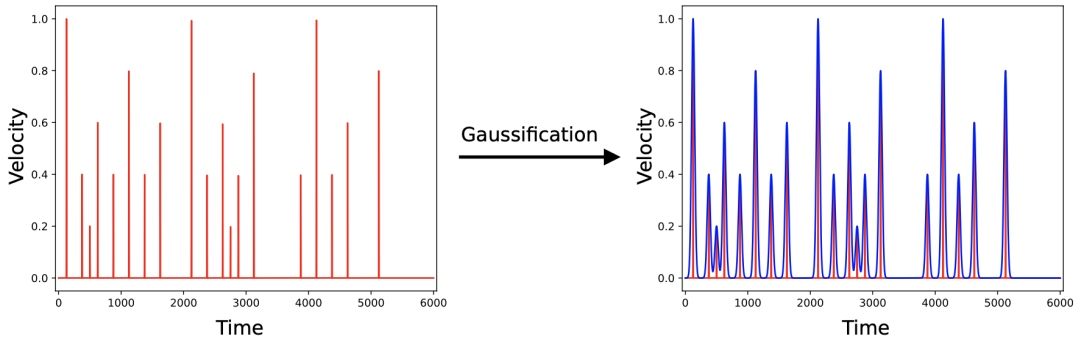


Figure 3.1: Example of Gaussification. On the left, a set of time points with velocities are represented as spikes over time. On the left, a Gaussification is applied to the same points, to obtain a integrable function.

3.3 | Real Time Tempo and Meter Tracking

This section describes the algorithm for the tempo and meter detection that is used within the Serious Game described in Section 3.4. Due to the length of its description, its importance within the system, and the fact that it can be potentially be applied to different applications, it is discussed separately.

3.3.1 | Theoretical Basis

The proposed algorithm is based on the concept of Gaussification, as introduced by Frieler (Frieler, 2004). The algorithm's input is a list R of N time points (onsets of notes) and a list V of N coefficients (velocities of the same notes). These could be received through MIDI protocol, for example by playing on a velocity-sensitive MIDI pad, or could be inferred from an audio source extracting energy peaks. The system then constructs the linear combination of Gaussians centered at the time points in R, each multiplied by the corresponding coefficient in V (as an example, see Figure 3.1). To define Gaussification more precisely:

Let $R = \{t_i\}_{1 \leq i \leq N}$ and $V = \{v_i\}_{1 \leq i \leq N}$, and σ the standard deviation of the Gaussian kernel.

Then

$$G_{R,V}(t) = \sum_{i=1}^{N} v_i e^{-\frac{(t-t_i)^2}{2\sigma^2}}$$

is the Gaussification of R, V. Throughout the work, σ was fixed to 25 ms, as suggested by Frieler (Frieler, 2004).

The idea behind this representation of the input is that human players do not follow precisely the beat as a metronome would do, but the timing of each human-played note is an approximation of that regular beat. The Gaussification allows the algorithm to consider this approximation when computing other features.

The Gaussification has the advantage of being an integrable function. Thanks to that, Frieler (Frieler, 2004) was able to define the correlation between two Gaussifications as follows:

$$CG_{(R_1,V_1),(R_2,V_2)}(t) = \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} v_i^1 v_j^2 e^{-\frac{(t-(t_i^1-t_j^2))^2}{2\sigma^2}}$$

Similarly, one can compute the autocorrelation of a single Gaussification by computing its correlation with a time-shifted version of itself:

$$AG_{R,V}(t) = \sum_{i,j=1}^{N} v_i v_j e^{-\frac{(t-(t_i-t_j))^2}{2\sigma^2}} \tag{3.1}$$

The main difference between the formulations that are reported here and the original ones by Frieler (Frieler, 2004) is that the formula for the Gaussian used here is not the usual probability density function. This variant was chosen so that the peak of each Gaussian centered in t_i is exactly v_i, giving a more precise representation of the velocity of each note.

Another important function needed for the meter and beat detection algorithm is one that allows choosing the most likely tempo when more than one is possible. If a song has a tempo of 100 bpm, it is possible to perceive it as 50 bpm or 200 bpm: if one makes such a mistake tapping along the song, they would correctly tap to beats of the song, but either skipping one every two beats or by tapping twice every beat. Since this does not result in a loss of synchronization this is not to be considered a major mistake, but it is nonetheless necessary to choose the most likely tempo in these ambiguous situations. Psychology of rhythm comes in handy for this task. In particular, Frieler (Frieler, 2004) used a function

derived from Parncutt's pulse-period salience (Parncutt, 1994):

$$P(t) = e^{-2\log_2^2 t/t_s} \tag{3.2}$$

where t_s is the "spontaneous tempo" expressed in milliseconds, that is the tempo that humans are more likely to tap to if instructed to regularly tap to a tempo of their own choice. Throughout the research, t_s was fixed to 500 ms (120 bpm).

3.3.2 | Algorithm

This section describes the procedures used to estimate the tempo and meter, and also to predict the onset of measures needed to effectively follow the rhythm established by an improvising human.

$events \leftarrow$ list of all the rhythmic events. Each element of this list contains two fields: $timestamp$, $velocity$;
$window \leftarrow$ the width of the time window in milliseconds;
Input: $events, window$
for $ainevents$ **do**
  if $a.timestamp < now() - window$ **then**
    remove a from events;
  else
    $a.timestamp \leftarrow a.timestamp - (now() - window)$;
  end
end
$N \leftarrow$ length($events$);
for $i \leftarrow$ *to* N **do**
  $R[i] \leftarrow events[i].timestamp$;
  $V[i] \leftarrow events[i].velocity$;
end
$norm \leftarrow AG_{R,V}(0)$;
for $i \leftarrow$ 100 **to** 2000 **do**
  $A[i] \leftarrow (AG_{R,V}(i)/norm) * P(i)$;
end
$beat \leftarrow argmax(A)$;
$clarity \leftarrow max(A)/P[beat]$;
return $beat, clarity$;

Algorithm 1: Estimation of the beat duration, along with its clarity.

3.3.2.1 | Beat Estimate

Algorithm 1 shows how the tempo is estimated. Of the input set of time points, only the ones played in the last *window* ms are kept. The autocorrelation of the gaussification of the points within this time window is computed as described in the previous section. The rationale behind this is that a musical rhythm will be highly similar to itself when shifted by the duration of a beat, thus the autocorrelation of the rhythm will peak on the most reasonable beat candidates. The possible beat durations considered are between 100 ms and 2000 ms, corresponding to tempos between 600 bpm and 30 bpm. For each possible tempo, the autocorrelation is normalized by dividing the result with the autocorrelation of the signal with itself, that is greater or equal to any other possible autocorrelation value. This gives a result in the [0,1] interval, that is then multiplied by the value of function (3.2) for the same tempo. This weighting makes the algorithm prefer tempos that are closer to the perceived preferred tempo. The amount of time that gives the maximum result is the chosen duration of a beat. To obtain a value in bpm one simply needs to use the following equation:

$$bpm = \frac{60000}{beat\ duration}$$

The algorithm also computes the clarity, that is simply the value of the autocorrelation of the chosen tempo before the application of (3.2). This is a value in the [0,1] interval and can be seen as the confidence of the algorithm in stating that the chosen beat is the correct one.

For simplicity, Algorithm 1 shows how the autocorrelation is computed with the formula introduced above. In practice, this is rather inefficient. Algorithm 2 shows how to incrementally compute the autocorrelation every time a new rhythmic event is added, by saving the partial sums in the events themselves. By doing this, the partial autocorrelation sums of events that are outside the window are discarded and only those that are still within the window are considered, along with the new event. The Add_Event function returns the autocorrelation at a time shift t. To compute the beat and clarity, this function must be inserted at the beginning of Algorithm 1 substituting the first two *for* cycles.

$events \leftarrow$ list of all prior events, added to the list with the following function. Each element of this list contains three fields: $timestamp$, $velocity$ and sum_t;
$window \leftarrow$ the width of the time window in milliseconds;
Function `Add_Event`($timestamp, velocity, t$)
  Output: $AG_{R,V}(t)$
  for *a in events* **do**
    $a.sum_t \leftarrow$
    $a.sum_t + velocity * a.velocity * e^{-\frac{(t-(timestamp-a.timestamp))^2}{2\sigma^2}}$;
  end
  $events \leftarrow$ events.append(event(timestamp, velocity, 0));
  $result \leftarrow 0$;
  for *a in events* **do**
    if *a.timestamp* $>$ *now() - window* **then**
      $result \leftarrow$ result + a.sum$_t$;
    else
      remove a from events;
    end
  end
  return *result*;
end

Algorithm 2: The algorithm for incrementally computing the autocorrelation of the input signal at a given time-shift t. The call to function $now()$ should output the current timestamp.

3.3.2.2 | Meter Estimate

Once the beat is estimated, the meter is computed based on the assumption that meter emerges as a pattern of accents (Povel and Essens, 1985). A series of rhythmic patterns are predefined, and a "prototypical" signal is constructed for each by generating time points distanced by the estimated beat, having a velocity defined by the rhythmic pattern, following the procedure in Algorithm 3. After these prototypes are generated, the meter is estimated by choosing the prototype that has the highest correlation to the input time points set. In this implementation only $\frac{3}{4}$ and $\frac{4}{4}$ were considered as possible meters, but it would be easy to extend the procedure to other meters using additional prototypes. Exactly nine time points are generated so that the sum of the velocities of all the points are the same between the two prototypes, meaning that the correlations computed for the two meters are comparable. Since there is no guarantee that the beginning of the window coincides with the beginning

of a measure, the the correlation is computed giving a variable time shift as input, that "moves" the prototype along the input points. The time shift that results in the highest correlation is the "phase" of the signal, i.e. the beginning of a measure in the window, as shown in Figure 3.2. The meter whose prototype has the highest correlation (considered the phase) is the chosen estimate for meter. The complete procedure is reported in Algorithm 4, that uses the function $generatePrototype()$ described by Algorithm 3.

3.3.2.3 | Estimation of Next Measure

From the other procedures, the system already has an estimate of the duration of a beat and of the number of beats in a measure, thus can easily obtain the duration of a measure:

$$measure = beat \cdot meter$$

In order to be synchronized with the human improvisation, the system needs a way to determine when to expect the beginning of a measure, just like a human joining an improvisation

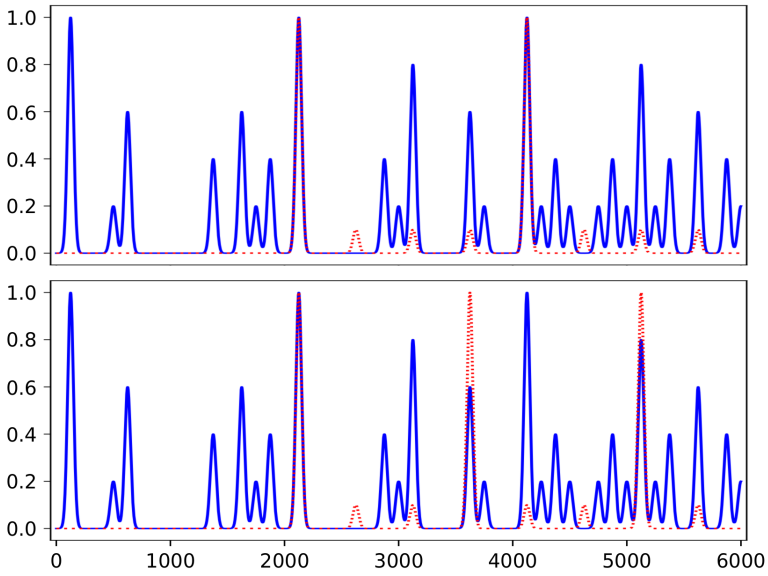


Figure 3.2: The gaussification of an input set of time points (solid blue line) is compared to the prototypes for $\frac{4}{4}$ (top) and $\frac{3}{4}$ (bottom) meters (red dotted lines). The prototype's phases computed by the system translated them to maximize the correlation with the input, but $\frac{4}{4}$ was (correctly) chosen in this case as it scores a higher correlation.

Input: $meter, beat$
$R_{meter} \leftarrow [], V_{meter} \leftarrow []$;
for $i \leftarrow 0$ **to** 8 **do**
  $R_{meter}.append(i * beat)$;
  if $modulo(i, meter) == 0$ **then**
    $V_{meter}.append(1)$;
  else
    $V_{meter}.append(0.1)$;
  end
end
return R_{meter}, V_{meter};

Algorithm 3: Generation of a prototype.

Input: $beat, R, V, window, current\_time$
$R_3, V_3 \leftarrow generatePrototype(3, beat)$;
$R_4, V_4 \leftarrow generatePrototype(4, beat)$;
for $i \leftarrow 0$ **to** $window$ **do**
  $corr_3[i] \leftarrow CG_{(R,V),(R_3,V_3)}(i)$;
  $corr_4[i] \leftarrow CG_{(R,V),(R_4,V_4)}(i)$;
end
if $max(corr_3) > max(corr_4)$ **then**
  $meter \leftarrow 3$;
else
  $meter \leftarrow 4$;
end
$phase \leftarrow argmax(corr_{meter})$;
return $meter, phase$;

Algorithm 4: Estimation of the meter, along with the phase.

would wait for the beginning of a measure. It is possible to forecast the beginning of first measure outside the considered window with the following:

$$onset = current\_time + measure - ((window - phase)\%measure)$$

Where % is the modulo operation, and $window$ must be the same time window used in the above procedures. Ideally, the start of the next measure is obtained by adding the duration of a measure until the result is beyond the window. In practice, the same result is achieved implicitly via the modulo operation.

3.3.3 | Evaluation

The evaluation of the system was carried out through a variety of simulations, trying to quantitatively assess if and how well the requirements described in the Introduction are met by the system. Since the literature is scarce on systems for the estimation of tempo and meter in a symbolic context, there are not many available datasets to test the effectiveness of the system. Even less datasets are available that account for human deviations from a perfect tempo. For this reason, most of the experiments described below are simulation based. For the same reason, it is not easy to compare the results of this system with our state-of-the-art approaches. Both real human-performances and a comparison to a state-of-the-art algorithm can be found in section 3.3.3.5, but some concessions had to be made and the dataset had to be adapted in order to compare the results of this system with a system which is meant to work with audio files.

3.3.3.1 | Experiment 1: Real-Time

Before describing the experiment in itself, it is useful to make a few considerations on the implementation of the system that are relevant to real-time computation. Envisioning an application that needs to synchronize with an input rhythm, the rhythmic analysis process should be called at constant time intervals. For example, two or three times every second (every 500 or 333 ms). To consider the requirement satisfied the computation needs to be carried out before another call is made (i.e., it must last less than 333 or 500 ms).

Method The computational time of the algorithms described above grows as $O(N^2)$, where N is the size of the input arrays of time points and velocities, or more simply put the number of input notes. This depends on the user improvisation and on the size of the window. For this simulation, the size of the window was fixed to 6000 ms, and generated regular rhythms having from 30 to 80 notes (with a step of 5) inside the 6000 ms time window, and run the algorithm for the estimate of tempo, meter and onset of next measure 50 times per each setting on a 2013 Macbook Pro (2.4GHz Intel i5 processor, 4GB Ram). The results are shown in Table 3.1.

Notes	Avg. Time [s]	SD	% SD
30	162.62	2.17	1.34
35	215.32	3.15	1.46
40	253.98	4.37	1.72
45	315.04	5.30	1.68
50	360.70	6.20	1.72
55	432.86	11.37	2.63
60	486.32	11.03	2.27
65	568.56	3.44	0.61
70	640.88	4.15	0.65
75	705.24	12.18	1.73
80	788.92	15.30	1.94

Table 3.1: Average execution time (in seconds) for the extraction of tempo, meter and beginning of next measure, along with absolute and percent standard deviation (SD). The timings were measured on a 2013 Macbook Pro (2.4GHz Intel i5 processor, 4GB Ram)

Discussion Despite the low computational power of the used machine, the algorithm is able to analyze a set of 45 notes in less than 333 ms, and up to 60 notes in less than 500 ms. With the use of the incremental algorithm for the computation of the autocorrelation, the only notes that must be considered are those played since the last calculation. In practice, this means that the system can follow the execution as long as the users play less than 45 notes in 333 ms, or one note every 7.4 ms. Given the fact that this is borderline impossible, I consider the system to be completely capable of functioning in real-time.

3.3.3.2 | Experiment 2: Steady Tempo

The second experiment aims to compute metrics relating to the ability of the system to predict the tempo and meter, assuming a quasi-steady rhythmic production. The performance is evaluated using as input the rhythms generated by a simulator, able to create random event, characterized by reasonable (in the sense of human-like) constant tempo and meter, given as parameters.

Simulations Setup The simulation used does *not* work in real-time, but it simulates the passing of time via an internal metronome implemented as an integer counter. Iteratively,

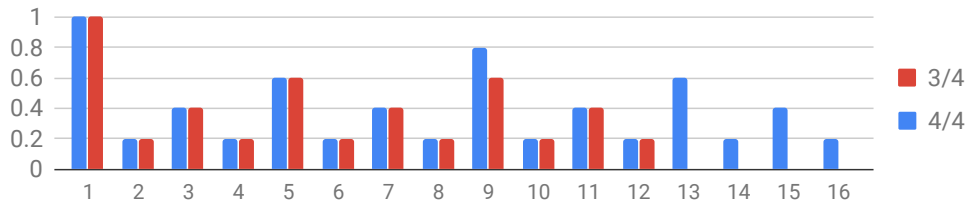


Figure 3.3: The importance assigned by the simulator to each of the 16th notes positions in a measure, both for $\frac{4}{4}$ and $\frac{3}{4}$ meters.

the simulator consults, for every 16th note, a predefined table determined by the chosen meter, whose cells represent the importance of each metric position in the measure. The used weights are reported in Figure 3.3. Algorithm 5 shows how the table is used to generate a rhythm in $\frac{4}{4}$ meter (but is easy to adapt to $\frac{3}{4}$): for every 16th note the content of the table represents both the probability of generating a note corresponding to that time point and the velocity associated to that note if generated. Both Figure 3.1 and 3.2 are examples of $\frac{4}{4}$ rhythms generated in this way. The algorithm includes the possibility of adding a random error to the generated time, to simulate the human imprecision in keeping a rhythm, that depends on the σ_{err} parameter.

Input: *beat*
$metro \leftarrow 0$ $time \leftarrow 0$;
for $i \leftarrow 0$ **to** 160 **do**
  $importance \leftarrow table[metro]$;
  if $random(0,1) < importance$ **then**
    $R.append(time + error(\sigma_{err}))$;
    $V.append(importance)$;
  end
  $metro \leftarrow modulo((metro + 1), 16)$;
  $time \leftarrow (time + beat)$;
end
return R, V;

Algorithm 5: Generation of a simulated rhythm in $\frac{4}{4}$.

Metrics Every time a note is generated, an estimate for tempo, meter, and onset of next measure is done and compared to the correct value, to compute the following metrics:

Tempo Accuracy (T-AC) : a score of 100 is assigned to each correct tempo estimate, a score of 75 is instead given if the estimate is half or double the real tempo, as this is considered a minor mistake. A tolerance of 10 ms is considered in both cases. A score of 0 is given otherwise. The final value is the average score;

Meter Accuracy (M-AC) : the percent of times the estimated meter coincides with the real meter;

Precision (P) : computed as the ratio of estimates that are within 50 ms from an actual onset of a measure over the total amount of estimates;

Recall (R) : computed as the ratio of actual onsets of measures that are estimated within 50 ms over the total number of measures minus one (as the very first measure is impossible to forecast).

Precision and Recall are computed only when the simulation is over, while Tempo and Meter Accuracy are computed progressively during the simulation. This is important because the "real" tempo and meter are the ones that the simulator is considering at the moment of the estimate, and it could change over the simulation for some experiments.

Experiment Setup For this experiment the simulator module was instantiated varying two parameters. The first was the tempo, that varied from 60 bpm to 200 bpm with 20 bpm intervals. The second parameter is σ_{err}: when generating the onsets of the rhythmic events, a random error based on the Normal distribution was added. σ_{err} is the standard deviation of the distribution, that imitates different rhythmic precision levels of the simulated human player (Repp, 2005). For each setting, the simulator ran 50 times, each on a freshly-generated random rhythm, and the metrics were averaged over these executions. The tempo did not change during an execution, and the meter was fixed to $\frac{4}{4}$ for all the experiment.

The goal of this experiment was to test how well the algorithm responded to rhythmic imprecision, and if the chosen tempo was relevant to the effectiveness of the algorithm. The results of these experiments are reported in Tables 3.2 and 3.3.

σ_{err} [ms]	T-AC (SD)	M-AC (SD)	P (SD)	R (SD)
0	79.6 (19.1)	87.8 (10.3)	54.8 (19.6)	86.4 (14.5)
2.5	79.5 (17.2)	87.4 (10.5)	53.3 (19.0)	85.8 (13.9)
5	76.0 (18.6)	87.6 (11.5)	49.9 (20.0)	83.1 (15.9)
7.5	70.5 (20.4)	86.1 (11.5)	45.6 (19.7)	80.1 (15.9)
10	66.7 (21.2)	86.0 (12.2)	42.5 (19.3)	77.0 (17.7)
15	53.9 (23.3)	84.8 (11.7)	34.5 (17.5)	70.3 (18.8)
20	41.2 (22.5)	83.0 (11.8)	25.8 (15.3)	59.7 (19.8)
25	33.4 (20.5)	81.1 (12.1)	21.2 (14.3)	53.0 (20.4)

Table 3.2: The four metrics (with standard deviation) computed over the various trials for each of the σ_{err} settings, averaged across all tempo settings.

Beat [ms]([bpm])	T-AC (SD)	M-AC (SD)	P (SD)	R (SD)
1000 (60)	56.9 (18.2)	93.5 (5.6)	30.1 (12.5)	76.8 (18.3)
750 (80)	58.0 (24.6)	88.4 (7.2)	32.5 (15.0)	72.8 (19.1)
600 (100)	78.1 (22.6)	83.2 (6.9)	45.5 (17.0)	83.5 (17.6)
500 (120)	81.9 (19.9)	94.7 (5.8)	66.2 (21.0)	88.5 (13.7)
428 (∼140)	78.1 (23.0)	92.9 (7.1)	52.0 (20.5)	86.1 (16.7)
375 (160)	45.8 (28.5)	83.4 (13.2)	19.3 (9.6)	49.8 (16.1)
333 (∼180)	44.3 (21.8)	74.0 (9.5)	34.5 (15.8)	66.7 (18.2)
300 (200)	57.6 (18.99	73.8 (10.9)	47.3 (19.1)	71.0 (17.3)

Table 3.3: The four metrics (with standard deviation) computed over the various trials for each of the tempo settings, averaged across all σ_{err} settings.

Discussion In the best-performing conditions, i.e. either having very low σ_{err} or having a tempo of 120 bpm, the system performs really well: both the tempo and meter accuracy are near or above 80. The recall in estimating the beginning of new measures is generally much higher than the precision, and is generally lower than what one could expect from the value of the tempo and meter accuracy: if the tempo and meter are correctly identified it should not be difficult to foresee a new measure. This difference can be explained in two ways. First, the measure onset determination procedure may not be precise enough, meaning that sometimes the beginning of a measure is estimated to be one of the weaker beats and not the first one. Second, the fact that a double tempo is considered as nearly correct means that the prediction might fall on half a measure rather than a full one. By looking more qualitatively at some of the generated rhythms where there are errors, it seems that the

sum of these two factors makes it happen that the prediction often falls on the third and sometimes on the second or fourth beat.

When deviating from the best settings, the performances decrease, especially concerning the tempo accuracy and the precision. Changing the tempo from the ideal one, the performance of the tempo accuracy degrades, probably because of the Parncutt function (3.2) the double or half tempo estimates become more frequent as the tempo becomes more extreme. Yet, the change is different for slow and fast tempos: faster tempos perform worse. This is probably due to the fact that the error on the input given by σ_{err} becomes more evident as the intervals between notes become shorter.

As σ_{err} grows, the performances worsen (although the meter accuracy remains over 80), but this comes to little surprise. With σ_{err} $\leq$15 ms (as expected from a musically trained human, although this depends on many factors (Repp, 2005)), the tempo estimate remains correct more than half of the times, but having σ_{err} $>$20 ms (which is expected from a non-musician playing a steady rhythm) is below that threshold. This, coupled with the fact that the precision is rather low, means that additional controls might be needed to keep the estimate and the ability to follow the improvisation consistent, for example taking the median of a series of consecutive estimates.

3.3.3.3 | Experiment 3: Sudden Changes

The above experiment only evaluated the general effectiveness of the proposed method when a fixed tempo and meter is kept throughout the execution. This experiment tries to check how quickly the system can adapt if those values suddenly change.

Experimental Setup This experiment uses two simulations: one in which the meter changes, and one where the tempo changes. In both scenarios, the simulation performs five measures without any change, and then suddenly changes setting. From that moment, the time that passes until the system has predicted the new tempo or meter correctly ten times is measured, meaning that the system has successfully adapted.

When changing the meter, the simulation was run with three tempo settings to see how

	Tempo [ms]	Avg. Time [ms] (SD)	Measures
$\frac{4}{4} \rightarrow \frac{3}{4}$	375	6598 (2355)	5.87
	500	4625 (1488)	3.08
	750	5662 (2013)	2.52
$\frac{3}{4} \rightarrow \frac{4}{4}$	375	6257 (2376)	4.17
	500	5047 (976)	2.52
	750	3465 (924)	1.16

Table 3.4: The time needed by the algorithm to detect a change of meter, in milliseconds and in number of measures.

Δ Beat [ms]	Avg. Time [ms] (SD)	Measures
-50	7463 (1158)	4.15
50	7711 (1775)	3.51
-100	6557 (1778)	4.10
100	7995 (1918)	3.33
-150	7136 (1315)	5.10
150	8866 (2347)	3.41
-200	5537 (1213)	4.61
200	9971 (3048)	3.56

Table 3.5: The time needed by the algorithm to detect a change of tempo (millisecond added or subtracted from each beat), in milliseconds and in number of measures.

the estimate was affected, both going from a $\frac{4}{4}$ to a $\frac{3}{4}$ meter and vice-versa. The results, averaged over 50 runs for each setting, are shown on Table 3.4. When changing the tempo instead, the initial tempo was fixed to 500 ms (120 bpm), but the amount of change varied from 50 to 200 ms (both increasing and decreasing the speed). Each setting was tested on 50 runs. The results are reported on Table 3.5. The σ_{err} parameter was kept to 10 ms. Since the reported timings are highly dependent on the tempo, the results are also reported as number of measures.

Discussion As evident from the results, sudden changes do not lead to immediate adapting. Even for a human it is not easy to immediately detect a change in meter before listening to at least one full measure to realize that the metric accents have changed. That considered, the results for meter changes are satisfactory.

For a human, noticing that the tempo has changed is usually more immediate, but

Step	T-AC (SD)	M-AC (SD)	P (SD)	R (SD)
0	87.1 (10.7)	97.4 (2.8)	69.3 (12.3)	90.2 (9.6)
1	60.0 (22.3)	92.7 (6.6)	45.7 (22.6)	78.0 (19.2)
2	49.3 (22.9)	92.4 (6.4)	36.9 (20.8)	68.3 (22.0)
3	40.8 (26.0)	90.2 (8.6)	31.4 (19.5)	61.4 (22.3)
4	35.7 (27.5)	87.8 (9.8)	27.6 (19.6)	55.8 (21.9)
5	34.3 (28.19	86.8 (9.2)	26.3 (19.7)	54.5 (23.9)

Table 3.6: The evaluation of the estimates when the tempo grows (or decreases) by Step milliseconds every 16th note.

instead the system performs worse on this task, especially when the tempo becomes faster (denoted on the tables by tempo intervals with the minus sign, as the bpm value is inversely related to the beat duration). The difficulty of the adaptation to tempo changes is probably due to the fact that the time window of considered notes is fixed to 6000 ms, and thus will consider the old tempo until the notes relating to that tempo are outside the window. Windows that vary in size could possibly be helpful to this task.

3.3.3.4 | Experiment 4: Gradual Tempo Change

The final experiment tests how the system reacts when the tempo grows not in a sudden way but gradually over time.

Experimental Setup The setup for this experiment was similar to that of Experiment 2, but here instead of having a fixed tempo, there were five measures with a fixed tempo of 500 ms, then a measure where the tempo incremented or decremented by a fixed amount every 16th note, followed by four measures where the reached tempo remained unchanged (the final tempo is 500 $ms \pm increment \cdot 16$). The possible values for the tempo change were 1 to 5 ms, and for each 50 runs were made increasing the tempo and 50 runs were made decreasing the tempo. The σ_{err} parameter was kept to 10 ms. The averaged results, along with a baseline where the tempo does not change, are reported in Table 3.6. The results are averaged over over 50 runs for each setting. There was no significant difference between the increasing tempo versus the decreasing one, so the two cases were joined.

Discussion The results show that the system cannot maintain its stability even if the tempo change is performed over a period of time rather than immediately. If the change is limited, we can expect the performance to degrade in a limited way for what concerns the tempo estimate. The estimate of the next measure's onset becomes less and less effective, because the previsions that are made before the tempo change cannot account for the final tempo change. This is not really surprising: even for a human musician it is hard to follow a crescendo/rallentando only by hearing another musician perform it, without having practiced it before or without having a visual cue (like a director).

3.3.3.5 | Experiment 5: Human Performance

Setup To verify the tempo tracking abilities of the system, a corpus of rhythmic performances is needed, having a known ground truth tempo. Moreover, since the system uses symbolic information (timing and velocity) based on the MIDI protocol, a corpus of MIDI files (or other symbolic formats) must be used for the assessment, rather than a corpus of audio files (such as `.mp3` or `.wav` files). Many such corpora exist and are available freely on the internet since such datasets are also used for music production. However, most datasets are mostly (if not entirely) comprised of beats in $\frac{4}{4}$ meter, and generated from scores or other sources that use quantized timings, representing an ideal perfect tempo. Instead, real human performances (the ones that would be most interesting here) include subtle variations from the established tempo due to human expressiveness or to imprecisions and other physical constraints.

The chosen dataset is the Groove Dataset (Gillick et al., 2019), created as part of Google's Magenta Project[1]. This dataset includes more than thirteen hours of drum MIDI files, recorded directly from human performances. The problem of not having many samples in meters other than $\frac{4}{4}$ is present in this dataset as well: of its 1150 files, only twelve are not in $\frac{4}{4}$ meter. Therefore, I decided to eliminate those twelve files and focus on $\frac{4}{4}$ for this evaluation. This dataset also includes many "fills" samples, i.e., short phrases meant to join different parts of a song. These are not fit for tempo tracking because of the short length

[1] `https://magenta.tensorflow.org/datasets/groove`

that would not allow the algorithm to find a rhythmic similarity. Those were thus excluded, along with all the files that lasted less than two measures, leaving a total of 448 files that were used for the evaluation of the system.

Since the proposed system is not designed to simply tell the tempo of a given MIDI file, for the goal of this evaluation each input file was processed as follows:

- all MIDI note_on messages were considered a single hit on one pad of the system;
- notes that happened at the same moment (with a 25 milliseconds tolerance) were joined to be considered as a single event, summing their velocity and keeping the timestamp of the first note;
- only the first 15 seconds of each file were kept: the system considers only the notes within a defined time window, which was less than 15 seconds in all experiments, so the remaining part was disregarded for this evaluation.

The tempo output given by the system represents what the system estimates as the tempo for that particular moment, as the system never considers the file as a whole. This means that it uses less information than what is available, but this is a better simulation of what happens during the actual improvisation game, where the system must evaluate the tempo only based on the last few seconds of improvisation. Two metrics were used in this evaluation: the percentage of files for which the tempo was correctly identified, and the percentage of files for which the tempo estimate is double or half the ground truth tempo. Since playing double or half tempo still allows synchronization (while the "feel" is different, the beats of a halved tempo always fall on a beat of the regular tempo), this is considered a minor mistake, and thus the sum of these percentages can be considered as an indication of the precision of the system. These metrics were also used as a way to tune the three parameters of the tempo tracking system: σ, used to compute the autocorrelation of the signal (see Equation 3.1 and Algorithm 2), β, a damping factor for the Parncutt function (Equation 3.2), and the time window of events to be considered when computing the tempo.

Window (ms)	Correct	Imprecise	Total
2500	58.80%	15.80%	74.60%
5000	66.33%	14.71%	81.05%
7500	69.47%	13.65%	83.12%
10000	70.31%	13.27%	83.58%

Table 3.7: The results obtained by the system with various window settings, averaged across all β and σ values tested.

β	Correct	Imprecise	Total
0.25	63.77%	21.91%	85.68%
0.5	70.15%	15.80%	85.95%
0.75	71.82%	13.07%	84.89%
1	71.80%	11.84%	83.63%
1.5	70.85%	10.13%	80.98%
2	68.40%	9.19%	77.59%

Table 3.8: The results obtained by the system with various β settings, with a time window fixed to 7500 ms, averaged across all σ values tested.

σ	Correct	Imprecise	Total
7.5	66.16%	17.36%	83.53%
10	68.15%	16.32%	84.47%
12.5	68.92%	15.59%	84.51%
15	69.73%	14.79%	84.51%
17.5	69.75%	14.43%	84.18%
20	70.53%	13.84%	84.37%
22.5	70.66%	13.67%	84.33%
25	70.95%	13.42%	84.37%
27.5	71.24%	13.05%	84.29%
30	71.11%	12.76%	83.87%
32.5	70.78%	12.34%	83.13%
35	69.66%	11.80%	81.46%
37.5	68.20%	11.22%	79.43%
40	66.67%	10.56%	77.22%

Table 3.9: The results obtained by the system with various σ settings, with a time window fixed to 7500 ms, averaged across all β values tested.

Discussion Various values for those parameters were tested, as reported in Tables 3.7-3.9, testing all the combinations between these parameter settings. The tables report the percentages of estimates that were correct, as well as the percentage of estimates that were double or half the correct tempo ("Imprecise" guesses), and the total of times which the system guessed either correctly or imprecisely. Table 3.7 shows a first result which was to be expected: lengthening the window makes the estimates more accurate. These results would make us choose the longest window possible, but, in this context, the tempo does not vary from the established one. Having longer windows also means that the system will be slower reacting to tempo changes, a feature that is not tested here but is considered important to the system. Therefore, the window was fixed to 7500 ms in the following tables. Tables 3.8 and 3.9 show the results when changing the β and σ parameters, which are useful for tuning the system. From both tables, we can see correct estimates are not directly correlated to imprecise estimates. This means that tuning the parameters can make a difference in how the system interprets different tempo candidates (especially the β, used in 3.2 to distinguish between promising tempo candidates given by the autocorrelation computation). When choosing the parameters, a tradeoff must be found between favoring the correct estimate (that will result in a better experience) and tolerating more imprecise estimates that allow for synchronization even if the algorithm is not accurate. I fixed $\beta = 0.75$ and $\sigma = 25$, a combination that gave correct results in 74.06% of the cases and an additional 12.47% of imprecise results, for a total of 86.53% of "acceptable" estimates.

Comparison It would be useful to compare these results to those obtained by algorithms for tempo tracking to give this data more meaning. Within the field of Music Information Retrieval (MIR), MIREX[2] is the primary collection of challenges and datasets relating to MIR tasks, so it is customary to compare new results to those obtained in those challenges. The challenge that comes closest to my goal is that of "Audio Tempo Estimation", which, as the name suggests, involves detecting the tempo of audio files. While being the most similar, being geared towards audio files makes this challenge very different from my goal of detecting tempo in a symbolic setting, but MIREX does not include any challenge or dataset

[2]Music Information Retrieval Evaluation eXchange, `www.music-ir.org`

	Correct	Imprecise	Total
Proposal	74.06%	12.47%	86.53%
Tempo-CNN	70.31%	21.43%	91.74%

Table 3.10: Comparison of the results of the proposed system (window=7500, $\beta = 0.75$, $\sigma = 25$) and those obtained by Tempo-CNN.

for the estimation of tempo in symbolic files. This may be because, in most scenarios, a symbolic file will come with a tempo annotation (which is sometimes necessary to play the file) making it useless to use algorithms to infer the tempo, although in our situation as well as in other interactive applications it makes completely sense to estimate the tempo of a symbolic sequence. To have a benchmark, a comparison with Tempo-CNN (Schreiber and Müller, 2018) is reported below. Tempo-CNN is an open-source algorithm that is freely available online[3] and that performed well on the 2018 MIREX Audio Tempo Estimation challenge, which is the latest at the time of writing as the challenge was not repeated in later years. Instead of using the MIREX dataset for the comparison, which is composed of audio files and cannot function with the proposed system, I rendered the MIDI files from the Groove dataset into audio files. The Groove dataset already includes an audio rendering of all its files, but using a standard rendering would not have allowed a fair comparison with my system, which only uses timestamps and velocity. To avoid including too much information (mainly timbre variation, due to using a full drumkit rather than a single pad), I rendered the MIDI using only a single sound for all the note_on events in the files. I chose the Hi Bongo sound (note 60 on General Midi channel 10) because it is a hand-played drum, making it most similar to the playing style I would expect on the proposed system. The conversion from MIDI to .wav was performed using the Command Line tool TiMidity++[4]. For each of the 448 files in the corpus, Tempo-CNN estimated the tempo, and once again, the number of correct guesses and also guesses that are half or double the correct tempo wes computed.

Table 3.10 shows the results obtained by the system in the ideal setting, compared to

[3] https://github.com/hendriks73/tempo-cnn

[4] http://timidity.sourceforge.net

those of Tempo-CNN. The two systems performed similarly: my system guessed correctly more often, but Tempo-CNN had more acceptable guesses in total. It is also interesting comparing Tempo-CNN's result with the MIREX one, even if they are not directly comparable. MIREX requires submitted algorithms to guess two tempos rather than one, as the dataset includes two tempo annotations for each song. These two annotations are almost always one the double of the other, making the metric used by MIREX "At least one tempo correct" similar to "Total" guesses in this evaluation, which accounts for double/half tempo estimates. Tempo-CNN guessed more than 97% of the times at least one tempo correctly on both the datasets used by MIREX. This suggests that there is a performance drop between what registered at MIREX and the new evaluation due to the lack of timbrical information in the dataset, which highlights that it is harder to guess the tempo using the limited information available to the proposed system. It is also to be noted that the proposed system is extremely efficient: using Algorithm 2 for the computation of the autocorrelation incrementally, it is possible to calculate the autocorrelation at every new note in a couple of milliseconds, making this system extremely apt for real-time applications like the one described here. It is not immediately clear if Tempo-CNN could be adapted to operate in real-time.

3.4 | A Social Musical Game

This section describes the Serious Game that was implemented using the tempo detection system described above. The introduction to this chapter briefly discussed the potential therapeutic benefits of such an interaction between players. Regardless of these potential benefits, from the player's perspective, the game's objective is the interaction itself and the two players must collaborate rather than compete to obtain more rewarding results. The game starts without any music, and the two players must start playing the MIDI pads that constitute the gaming system's input. In the beginning, they can only hear the sound they make with the percussion. As the game goes on, the system evaluates how well the two players are interacting and assigns a score to their playing. This score is presented to the players in two ways: on-screen, as "points" collected by the pair by keeping

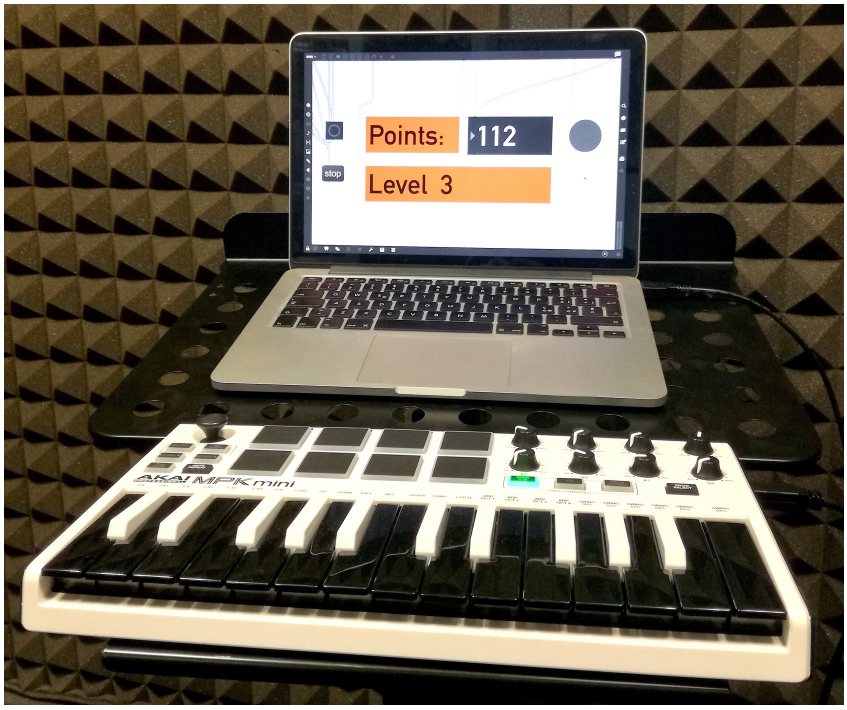


Figure 3.4: The interface of the system: a computer running the software connected via USB to a MIDI keyboard with drum pads.

a good interaction over time, and through "musical" feedback. This feedback is the musical augmentation itself, which follows the tempo and rhythm given by the two players but adds synthesized instruments to the musical production. Having a higher aesthetic value in the musical output should give a more engaging experience to users and incentivize better interactions between the players.

3.4.1 | Architecture

The system's architecture is divided into three main parts. The "Listener" module collects the inputs from the users and uses those to infer low-level features like tempo and measure duration. The second module is the "Scorer", which uses these low-level features to evaluate the interaction between the players and assign them a score. Finally, the "Generator" uses all the available information from the other two modules to generate the augmented music and synchronizes it to what the users are playing. These modules will be better explained in

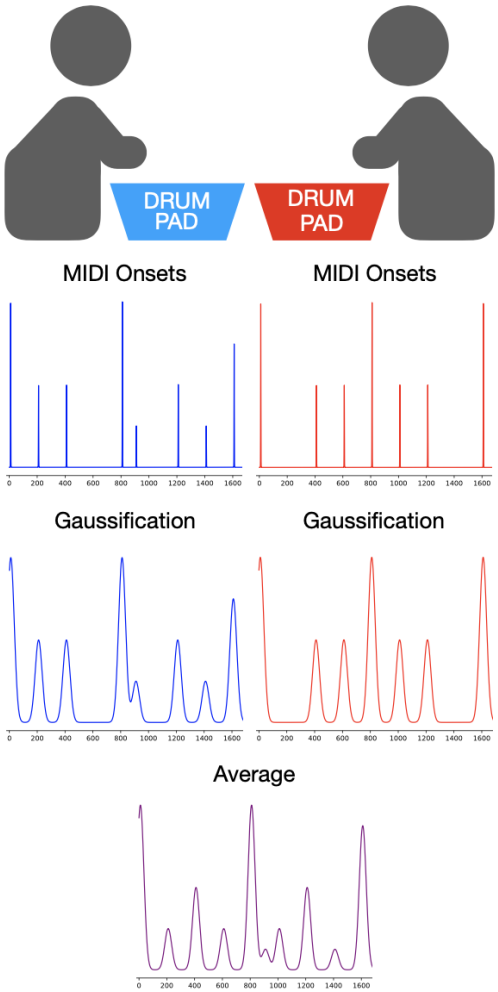


Figure 3.5: Data flow in the Listener module: the onsets and velocities are collected from the drum pads and then gaussified, and an average of the two signals is computed.

the following three subsections. Before reading the details of each module, the reader might find it useful to watch a video example execution of the game, available at the following URL: `https://mediaspace.unipd.it/media/0_g90zoo2n`

3.4.1.1 | Listener

The MIDI pads (the black squares in Figure 3.4) represent the only input interface of the system used by the users. In the current implementation, each player uses a single pad,

but this could easily be changed in the future. Every time someone hits a pad, it sends a note_on message to the computer running the gaming software, which saves a timestamp when the message is received, along with the MIDI velocity included in the message, which represents the force with which the user struck the pad. Being a strongly time-dependent application, the system currently does not allow for network play and the pads must be connected to the same computer. The rhythm played by each user is saved as a list of events, which can then be represented as the topmost plots in Figure 3.5 show.

The players are not required to follow a precise tempo, and their rhythm is completely improvised: at first, they can only hear the percussive sounds they generate by hitting the pads. The system must then infer the tempo they produce to add music that is synchronized with what the users play. To do so, the system uses the algorithm described in Section 3.3. That algorithm functions using a single set of time points, while in this setting each player produces one. The Listener thus calculates the tempo on an additional set of points which is the average of the two lists of events registered from the two users. Since the two players perceive the tempo as a feature emerging from the sounds produced by both, this approach is more appropriate for the estimation of the tempo the two players perceive, rather than computing the autocorrelation on both the signals and keeping a different tempo estimation for each user.

This module outputs the estimated tempo and meter of the average signal and a timestamp representing the moment in which the system estimates the next measure will begin. The beginning-of-measure prediction is fundamental for the Generator module, which uses it as a synchronization point between the music generated by the system and the sounds produced by the pads played by the users.

3.4.1.2 | Scorer

The features computed by the Listener module are needed for the correct generation of music and to ensure the synchronization of the generated music with the rhythm played by the users. The Scorer module instead considers features related to the gaming aspects of the system to give feedback to the users about the quality of their interaction. The

theoretical basis for this module derives from music therapy, and in particular from the "Improvisation Techniques for Music Therapy" devised by Kenneth Bruscia (Bruscia, 1987). Bruscia's book describes a wide set of techniques available to music therapists to improve their interaction with the client. These are divided into different categories and do not focus solely on the music production but also on the physical/visual interaction with the client, which is a kind of information that is not available in the proposed setting. I selected five of the main basic features that only rely on the rhythm produced by the players. Here are the definitions given by Perret in his comment on Bruscia's work (Perret, 2005) for the chosen techniques:

Imitation: Echoing or reproducing a client's response after the response has been completed;

Synchronization: Doing what the client is doing at the same time;

Incorporating: Using a musical motif or behavior of the client as a theme for improvising or composing, and elaborating it;

Pacing: Matching the client's energy level (i.e., intensity and speed);

Rhythmic grounding: Keeping a basic beat or providing a rhythmic foundation for the client's improvisation.

These techniques are meant to be used by the music therapist to help the client during the improvisation and do not immediately fit the situation where two peers are playing together and must be evaluated by a computer system. Nonetheless, they were useful in designing more precise and measurable features for the system. In particular, the system distinguishes four possible levels, describing the quality of the interaction:

Level 0: The system is incapable of clearly following the users, as they are not making a clear enough beat (no Synchronization or Rhythmic grounding);

Level 1: The system is capable of following the users, but one is dominating the rhythm and the other is not contributing (no Pacing);

Level 2: The interaction is considered normal: the two players have established a rhythm together;

Level 3: The interaction also includes imitations between the two players (Imitation and Incorporation).

The algorithm computes the level of interaction according to Algorithm 6, that requires as input the gaussified signals (the ones of the two players as well as the average one), the duration of beat and measure in milliseconds, and the list of the timestamps of the notes produced by the two players. The algorithm uses three functions: *correlation(a,b)*, that computes the correlation of the signal *a* with the signal *b*; *now()*, that returns the current timestamp; and *shift(a,b)* that moves along the x-axis all the points of the signal *a* by *b* units.

Input: $signal_1, signal_2, signal_a, beat, measure, notes_1, notes_2$
Output: Level of interaction
$clarity \leftarrow$ correlation($signal_a$, shift($signal_a, beat$))/ correlation($signal_a, signal_a$);
if *clarity* < 0.4 **then**
  return *0*;
end
$density_1 \leftarrow |\text{el} : \text{el} > \text{now()-10000} \ \& \ \text{el} \in notes_1|/10$;
$density_2 \leftarrow |\text{el} : \text{el} > \text{now()-10000} \ \& \ \text{el} \in notes_2|/10$;
if $density_1 < 0.5 \ || \ density_2 < 0.5$ **then**
  return *1*;
end
$crossCorr \leftarrow$ (correlation($signal_1$, shift($signal_2$,measure)) + correlation($signal_2$, shift($signal_1$,measure))) /2;
if *crossCorr* < 15 **then**
  return *2*;
else
  return *3*;
end

Algorithm 6: The algorithm for the computation of the current interaction level.

The algorithm computes three significant features to distinguish the levels. The $clarity$ represents how confident the system is in estimating the current beat. The two $density$ values represent the notes per second each user plays. Finally, $crossCorr$ is how similar the two signals are at the distance of one measure, i.e., how much the users imitate each other

measure per measure. The threshold values for the various levels were chosen empirically by computing the average values obtained by a "metronome" interaction, i.e. where the two users were substituted by a software sending a beat at regular intervals. It would be possible to use machine learning to determine better thresholds if enough data is collected from real users' interactions.

The final score is given to the users by adding a number of points each second depending on the current level. Notice that, since the goal of the game is the interaction between the two players rather than a challenge, the score is shared. The values of these features are not directly transformed into points given to the user, but rather to compute the current multiplication level. To ensure that the scoring system is stable, the level shown in the interface is not the latest computed level, but the median of the fifteen last computed levels. As well as being necessary for the visual feedback, the levels also influence the musical output as the music generation method is the same at every level (as described in the next section), but the volume of the generated instruments is proportional to the level reached by the players.

3.4.1.3 | Generator

In this section, the algorithms used to generate music that is then synchronized with the rhythm established by the users are described. However, there are potentially no constraints on the music that can be added to the system since the Listener module computes all that is necessary to synchronize music to the beat produced by the users. For example, it would be possible to playback a pre-selected audio file (e.g. the users' favorite song) and synchronize it via time warping (Dannenberg, 1984; Robertson and Plumbley, 2007; Moens et al., 2014). I decided to use simpler generated music to avoid taking the focus away from the interaction between the users. If the players are too captured by the music, they might start following it by tapping on its beats, becoming a sort of 'human metronome'. This would strongly reduce the interaction between the players and distract them from each other's rhythm. The Generator receives as input all the results of the other modules' computations. The most important information is the prediction of the beginning of the

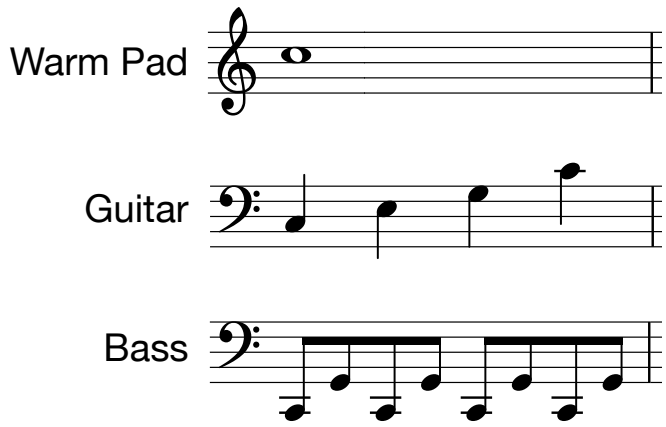


Figure 3.6: A generated measure in $\frac{4}{4}$ for each of the accompaniment instruments, based on a C major chord.

next measure. The Generator saves all the predictions obtained from the Listener, and each time a pad is struck the current time is compared with the saved predictions. If one is compatible with the current time (with a 50 milliseconds tolerance), the system estimates that the received input is the beginning of a new measure. The time tolerance is based on Parncutt's studies (Parncutt, 1994). Every time this happens, the tempo and the meter of the Generator are updated to match the ones computed by the Listener, and the internal metronome of the Generator is reset and started. The Generator could potentially start a measure at the predicted moment without waiting for input from the user, but the above approach was preferred since it increases the feeling of control over the output. Having the system react to specific actions performed by the user is important to obtain the feeling of "I made this happen", which is considered crucial for the effectiveness of music therapy (Swingler, 1998).

Harmonic Accompaniment Generation At the beginning of each measure, a chord is selected using a first-order Markov chain. This chain can be manually crafted or can be generated via a Python script that uses the music21 library[5] to read a lead sheet in MusicXML format. The chain is constructed by taking note of the frequency of moving from one chord to another in the input file. The script is written so that the chords are analyzed

[5] `https://web.mit.edu/music21/`

in a way that does not depend on the key, i.e. by using roman numerals instead of actual chord names. The software only generates chord progressions in the key of C major, but it would be easy to transpose the generated music to other tonalities.

Three accompaniment instruments are used to play the chords: an example is shown in Figure 3.6. The actual notes depend on the current chord, but the warm pad (General MIDI instrument 90) always plays the tonic while a guitar (General MIDI instrument 25) plays the chord arpeggio, and a bass (General MIDI instrument 34) alternates between the tonic and the fifth of the chord every eighth note. A new chord is selected each time there is synchronization between a note_on event and the beginning of a measure predicted by the Listener, but also when the internal metronome of the Generator reaches the 1st beat. This means that the chord changes with every measure unless the same chord is selected again.

Melody Generation To generate melody, a second Markov chain was constructed using the same Python script. Since the generated melodies should depend on the underlying chord being played at the moment, the script saves intervals between the tonic of the chord and the note being considered rather than the name of the note itself. The chain is restarted every time a chord changes, both in the learning and in the generation phase. Moreover, since some chords are major and others are minor, the intervals are saved as diatonic intervals and become a chromatic interval depending on the current chord during the generation phase. This process is briefly illustrated in Figure 3.7.

A chromatic percussion instrument is associated with each player: a Music Box (General MIDI instrument 11) and a Vibraphone (General MIDI instrument 12). Chromatic percussions are used to keep the direct interaction of users focused on the rhythmic aspect, but these instruments can add pitch to the played notes. The two players follow two parallel chains that are both aware of the chord being played by the system. The Markov chain only controls the pitch of the generated melody, while the player determines the rhythm. Moreover, the Markov chain does not output the final pitch but rather an interval relative to the current chord. Therefore, each time a player hits his pad, the actual new note is

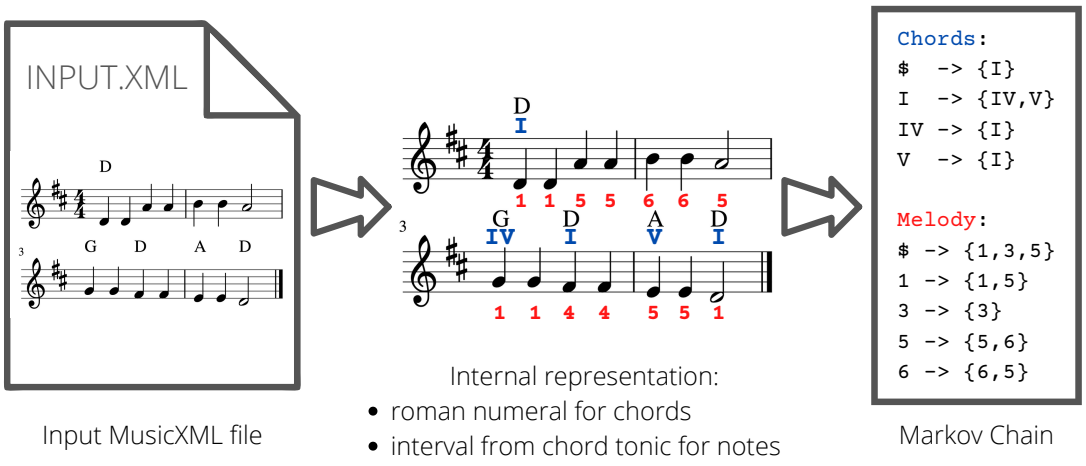


Figure 3.7: Data flow for the creation of the Markov chains from an input lead sheet. In the example chains, the likelihood of each transition is not represented.

generated following these steps:

1. Current chord is retrieved;

2. If the chord has changed, the seed is set to "$" (empty seed), else, the seed is the last generated interval;

3. The seed is used on the chain to generate the next interval;

4. The generated note is computed by following the interval chosen by the chain starting from the tonic of the current chord.

There is no control of the harmonicity of the notes generated by the two chains, but the use of diatonic intervals converted to chromatic intervals based on the chord ensures that it is impossible to generate strong dissonances. This approach of using intervals from the chord tonic to limit the output to "safe" notes was used by other notable work in Music Generation, such as Biles' GenJam (Biles, 2013b).

3.4.1.4 | Implementation

The modules described above represent the abstract functionalities of the system rather than actual software modules. The real implementation used for this system uses a Max/MSP[6] patch for the collection of the input from the user and the generation of the music. The patch communicates via Open Sound Control with a Python script that receives the list of inputs from the patch and computes both the features described in the Listener module and those related to the Scorer module. Practically, the Generator is implemented in Max/MSP and the Scorer in Python, but the Listener is shared between the two systems. This subdivision was chosen because Max/MSP is not fit for computations like those needed for the autocorrelation, but on the other hand Python is not ideal for real-time computing. Having a server invoked by the patch ensures that the computations done by Python are not critically dependent on timing (all the time labeling is handled by Max/MSP) while keeping the advantage of using Python and NumPy for the computation of the correlation of signals.

3.4.2 | Evaluation

The main goal of this study was to create a system that would help the players develop a rhythmic improvisation with music added by the software. The added music should help increase interpersonal synchronization, resulting in an engaging experience that can lead to better affiliation (Hove and Risen, 2009). The evaluation of the system's capabilities of following a tempo was described in the previous section. Here, I include a questionnaire-based evaluation of the user experience to measure the general appreciation and engagement of the users with the system.

Twenty-four participants were collected among university students who volunteered to test a musical game. Nine were females and fifteen males with an average age of 24.17 years (standard deviation 3.48 years). In pairs, the participants who agreed to take part in the study and signed the informed consent were told they would use a game to create a "rhythmic

[6] `https://cycling74.com/products/max/`

Question	Avg. Response	Std. Dev.
I found the game experience pleasant.	5.92	1.00
I think the music was aesthetically pleasing.	5.04	1.17
I wished to keep playing.	5.58	1.22
I wish to play again with this system in the future.	5.25	1.20
I was actively interacting with the other participant.	5.33	1.72
I was actively interacting with the system.	5.08	1.32
The system was interacting actively with me and the other participant.	5.50	1.29
I felt I had control over what was happening musically.	4.17	1.34
The system reacted to what I and the other participant were playing.	5.25	1.27
The music produced by the system was a stimulus to interact with the other player.	5.58	1.32

Table 3.11: The questions posed to the participants of the evaluation of the system with their average response and standard deviation on a scale from 1 (Completely Disagree) to 7 (Completely Agree).

interaction". Before starting the experiment, they were shown the drum pads and how hitting them would produce a percussive sound. They were instructed to collaboratively create a rhythm, only using the pads and not talking or communicating if not through the percussions of the pads. They were also told that the system would add music to their rhythmic performance based on how they interacted. Immediately after the explanation, the users were left alone, while the researcher listened to the interaction from the next room, and the users could start playing as soon as the researcher had left. For this evaluation, the system for the generation of melodies was left out, leaving only the harmonic accompaniment described in Section 3.4.1.3. This was done because while informally testing the system, I found that the melody can distract the users from the rhythmic interaction, which I wanted to be the main focus in this evaluation.

After playing with the system for three minutes, the researcher returned to the room and each of the participants was asked to fill a questionnaire consisting of ten 7-points Likert items asking their level of agreement with the presented sentences, on a scale from 1 (Completely Disagree) to 7 (Completely Agree). The questionnaire also asked what their level of musical expertise was using a single question (Zhang and Schubert, 2019), but except for one classically trained musician, all reported little amateur experience or no experience at all. All the questions were posed in Italian, as it was the native language of all

the participants. The questions in English, as well as the results of the questionnaire, are reported in Table 3.11. The Cronbach's alpha of the collected data is 0.83, showing that the questionnaire has reasonable reliability. The first four questions were meant to assess whether the participants liked to play with the system, while the other questions were more intended to assess if they felt the system helped them interact musically. These questions were chosen to test whether the players considered the game engaging, as this was the main aspect to be assessed via user-testing since it could not be directly assessed via quantitative measures by the researchers.

On average, the rating for the first questions was between 5 (Somewhat Agree) and 6 (Agree), indicating that the participants found the system to be pleasing to play with and would have liked to play again, meaning that the system is considered engaging by the participants. The results regarding the interactivity of the system are also positive, with averages between 5 and 6 except for question eight. The participants felt there was an interaction both between the players and with the system, and that the generated music was helpful to their interaction. Despite the perceived sense that the system reacted to their inputs, the participants did not feel like having full control over the produced music (the average response is around 4, Neither Agree nor Disagree). This is reasonable, since the rhythmic dimension of the music, the one that is directly controlled by the users, is only a fraction of the whole musical output.

Aside from the questionnaire, qualitative observations collected during the experiment by listening to the players' interactions show that while the system is both capable of following an established rhythm and quickly adapting to changes in tempo, the users are not driven to experiment with more complex interactions. This is probably because the system takes a few seconds to adapt to quick and abrupt changes (this is to be expected since the system needs at least a complete repetition of a period before being able to adapt), and the immediate feedback (before the system adapts) is negative. This induces the players to stick to the first common rhythm they can establish, which is not necessarily the best situation. Moreover, it was noticed that while the system is capable of handling meters different from $\frac{4}{4}$, all the participants only used this meter. The trained musician seemed

to try exploring more complex rhythms at times but settled for simpler ones possibly to accommodate the other player as well as not having immediate positive feedback from the system. This suggests that further studies only using musicians might show a different usage of the system. In general, the results collected from the questionnaire are not very strongly detached from the neutral response, showing that there is still much room for improvement in the system. Nonetheless, considering that the evaluation was carried out on what is the very first version of the system, the results are very encouraging. Further developments could improve the interaction to give a better feeling of control over the produced music, for example by adding more complex musical variations to the output mimicking the users' input more closely.

3.5 | Discussion

3.5.1 | Main Findings

This chapter described in detail how a Serious Game to improve sociality and interaction was developed. While this did not immediately relate to music generation per se, it was a useful to explore novel ways of music-based interaction, both between the players using the system, and between the human and the machine.

To create this system, the main effort was dedicated to the creation of the Listener module, that in more general terms is a system for the detection of meter and tempo in real time, based solely on symbolic information on rhythm. The system accepts a set of tuples describing the timing and the force (velocity) of played notes, and uses this information to determine the meter and tempo. The latter is estimated by calculating the autocorrelation of a signal built from the input. The meter estimation uses the same signal and compares it to pre-built signals that represent typical accents in different meters. Despite the need to calculate an autocorrelation at different time shifts, which in general could be an expensive operation, the system can leverage the discrete nature of the signal to perform this operation incrementally.

While tempo and meter detection are problems that are generally considered as solved

by the MIR community, this application had more stringent requirements than usual (i.e., having only access to timing and velocity), and having obtained results that are comparable to other state of the art approaches (when given similarly limited information) while working in real-time is extremely satisfactory. The need to detect tempo and meter in a symbolic context is uncommon (usually symbolic notation explicitly states this information) and for this reason the MIR community focuses more on audio-based detection of these features. Nonetheless, in the context of live music and live electronics, being able to detect the tempo played by a musical instrument capable of outputting midi or other symbolic information can be useful to a large number of artistic applications where some output from a machine should be aligned with a musical performance (the output can be additional musical sounds like in this case, but possibly also images, videos, or even actions performed by robots). In this context the algorithm presented here becomes a relevant proposal, since it is capable of functioning in real-time and makes extremely little assumptions on the input, making it capable of working in any context where a discrete/symbolic rhythmic information is available.

To complete the presented Serious Game, additional research and development were needed. The game was framed within the context of Music Therapy, by researching the principles of this discipline that would apply to the use case. Based on those principles, a game that evaluates how well two persons are interacting was developed, using Python and Max/MSP. This serves both as a way to illustrate how the real-time tempo detection system can be useful to develop interactive experiences, and as a way to show how computer-based music generation can be an extremely human-centric activity. In this case, the computer-generated music depends on and favors the interaction between the players, giving them part of the agency. In the context of Computational Creativity, this can be seen as a co-creative system, where the creative product is both the interaction between the players and the final musical product created in the interaction. Instead of being something that tries to substitute human composers (as some seem to view Computational Creativity applied to music in general, as reported by Sturm and Ben-Tal, 2021) it is something that empowers humans in creating something that is directly a product of their actions, even if they would

not have been capable of producing it on their own. This sense of agency and the creativity of the act are also important to the therapeutic aspects of the experience (Swingler, 1998; Dai et al., 2018). These considerations make the presented application a perfect example of how Computational Creativity can sometimes be needed to achieve things that human artists could not possibly do as effectively, as efficiently, or as inexpensively. Artificial music must not necessarily be a substitute for human musicians (as the above cited people seem to fear) but can instead be a powerful tool at the service of humans. While further work is required to fully assess the therapeutic usefulness of the proposed game, it already shows many potentialities in itself and in the development of the related human-computer interaction aspects.

3.5.2 | Limitations and Future Work

This work has primarily focused on the technical implementation of the rhythm following system and of the related Serious Game. While this in itself can be a valuable contribution, the therapeutic aspects of the serious game need to be further assessed.

Primarily, a long-term evaluation on the effects of using the serious game with continuity should be considered, to see if the desired therapeutic effects on sociality are achieved. More fine-tuned evaluations on the effects of different kinds of music and on the use of the melodic module described above could be implemented as well. These kind of evaluations were considered and planned, but the rise of the COVID-19 pandemic halted this study, as it required the players to be in close proximity. A networked version of this application was considered, but discarded since it would decrease the social effects that we wish to study.

The exploration of a social application for music generation was could also lead to the exploration of how social and emotional aspects of music impact on the perception of creativity. While this was not directly tested, this work tried to explore novel ways in which generated music can play a role into co-creative applications, where computational composers are not an alternative to the human, but rather create a joint musical effort. In this case, the effort also has the goal of assisting the human and helping the players socialize through non-verbal interaction. This can open many possibilities for Computational

Creativity applications, both with a therapeutic goal and without any goal beyond the generation of co-creative artifacts. Still, if one wishes to evaluate the creativity of this social game, the usual evaluation criteria for musical artifacts would be hard to apply. Different studies with other approaches should be considered in order to evaluate the direct impact of emotional value on the perceived musical creativity.

4

Structure in Music Generation

In this chapter some ad-hoc tree-based representations of musical structures are presented, with the goal of giving tools to allow structure-aware music generation. A system for music generation is also presented, which leverages the above structures and information theory concepts to impose structured generation based on the style of a corpus represented through these representations.

4.1 | The Hierarchical Nature of Music

Several studies concerning music cognition prove that our music perception uses some level of *abstraction* (Aiello, 1994; Deutsch et al., 2013; Schön et al., 2007). This perceptual abstraction can be retrieved from symbolic representation of music since, as Vinet (2003, p. 194) points out:

> "The symbolic representation is content-aware and describes events in relation to formalized concepts of music (music theory)."

This suggests that the symbolic level could include information about our music cognition and not just about music notation, which can be retrieved through the help of music theory.

Generally speaking, the operation of retrieving perceptual abstractions with the help of music theory concepts often leads to tree-based representations of music, where the

nodes describe notes or grouping of notes, which are linked based on respective importance according to music theory. Musicologists have used such representations to describe music on multiple levels at least since the works of Heinrich Schenker (1935).

In his influential books, he theorized that music is imagined by the composers in a hierarchical fashion, roughly divided in *Ursatz* (fundamental structure), *Mittelgrund* (middle ground), and *Vordergrund* (foreground). According to Schenker the *Ursatz* should always result in a scale descending to the tonic over an arpeggiation of the tonic chord, and the other levels represent successive extensions of this structure. In particular, the *Mittelgrund* includes all the most relevant notes and chords of the piece, and the *Vordergrund* represents instead the complete composition.

The influence of this theory is evident in the well known book by Lerdhal and Jackendoff, *A Generative Theory of Tonal Music* (GTTM) (Lerdahl and Jackendoff, 1985), where the authors also grouped notes in a composition to form different hierarchical levels of reductions. Differently from Schenker, these authors have provided an extensive rule-set to define how such reductions work, inspired by Chomsky's work on language (Chomsky, 1957).

These theories make it possible to describe different levels of importance of notes in a score that, according to musicologist David Temperley, should reflect both the composer's ideas in the generation of a composition and the listener perception of the same piece (Temperley, 2011), naturally incorporating other musical cognition concepts like tension and realization.

4.2 | Related Work

Structure can have different meanings, even within the context of music, so it is convenient to consider what different accounts of musical structure entail for the computational analysis and representability of such structures. Traditionally, music theory deals with the concept of structure either talking about how melodies can be divided into periods, phrases and motifs, or by analyzing how a piece can be divided into sections according to form

theory, for example recognizing "Exposition", "Development" and "Recapitulation" within a Sonata form. More advanced theories of music describe the hierarchical nature of musical pieces, such a Schenkerian Analysis (Schenker, 1935) or the Generative Theory of Tonal Music (Lerdahl and Jackendoff, 1985). These kinds of approaches form tree representations of the analyzed musical pieces, describing how different parts are linked together thorugh reductions. Other theories, such as the one proposed by Nattiez (1975), follow a syntagmatic approach, where the piece is seen as a series of syntagms, basilar melodic units that can be grouped according to their resemblance. A commonality among all these approaches is the fact that the musical piece is divided into smaller segments, which can be categorized and/or linked based on their content and some similarity measure, making it possible to group similar segments and differentiate segments that have different roles within the piece (Abdallah et al., 2016).

Since the first common step to this kind of structural considerations requires segmenting a piece into smaller parts, it is not surprising that a large amount of research has been devoted to automatic segmentation of melodies, a (sub-)task that is also useful to a variety of other musicological and music information retrieval applications.

The algorithms for segmenting melodies can be based on music theory principles, such as those inspired by Grouping Preference Rules from GTTM (Frankland and Cohen, 2004) which are mainly based on distance between notes and changes in duration or articulation within the local melodic context, or other theories like Temperley's Grouper model (Temperley, 2004) which also incorporates the concept of parallelism and considers that phrases within a certain piece should have roughly the same length.

One especially successful model for segmentation is the Local Boundary Detection Model (LBDM) by Cambouropoulos (2001), which is based on the degree of variation between features like pitch, duration and inter-onset-interval of notes in a local context. The basic idea is that the comparison between the end of a phrase and the beginning of the next one will result in a higher degree of difference than the comparisons operated within a single phrase.

Other models follow statistical and bottom-up approaches rather than rule-based ones.

For example, Hidden Markov Models have been applied to this task (Batlle and Cano, 2000), using aural features as the observations and the segmentation as hidden states. Others have implemented segmentation methods based on Information Theory concepts, where changes in Information Content denote melodic boundaries (Pearce and Wiggins, 2008). Another more advanced bottom-up approach that is not based on music theory leverages Haar Wavelets to filter the pitch contour at different time scales and then examining the filtered results to infer relevant segmentation points (Velarde et al., 2013). More recently, with the growing popularity of deep learning algorithms, the task of segmentation is also tackled with such algorithms, leading to satisfactory results (Guan et al., 2018; Zhang and Xia, 2020).

The task of identifying meaningful segments is not the only aspect to structural analysis: finding significant relationships between the segments is just as important. The first and foremost relationship between melodic segments is that of similarity. This means that melodic patterns can be exactly repeated within a musical piece, but they can also have variations that still represent a significant link between the segments. The concept of similarity itself is not devoid of ambiguity: ideally, we would like to capture as similar all those melodies which are perceived as similar by listeners. This perceptual definition requires to take into account cognitive aspects, making the detection of similarity not a trivial task (Wiggins, 2007b).

The easiest approach is that of finding exact repetitions. This can be achieved by means of efficient string-based algorithms applied to textual encodings of melodies (Cambouropoulos, 1998; Hsu et al., 1998). It is worth noting that even if only exact repetitions are considered, depending on the employed encoding certain melodic variations can be detected as well. For example, transposition invariance can be achieved by using an interval-based representation (Conklin and Anagnostopoulou, 2001). String based-pattern matching can also be applied to find approximate repetitions, i.e., repetitions that include a certain amount of editing operations, such as insertion, substitution or deletion (Rolland, 1999) but this generally requires less efficient algorithms. Another way to consider non-exact repetitions is to take into account sub-segments. One way to achieve this is by using prefix-trees which

allow to describe different hierarchical levels of repetition (Lartillot, 2016). Another method is to use geometric representations of musical content, and use geometric tools to describe possible transformations between melodic segments. This approach is at the base of SIA and SIATEC (Meredith et al., 2002). which can also deal elegantly with poliphonic music, unlike most other algorithms covered above. Given the success of these algorithms, other algorithms that derive from these two can be found in literature (Meredith, 2013; Forth and Wiggins, 2009). A deep architecture can also be applied to the task of pattern discovery: Pesek et al. (2017) applied a Compositional Hierarchical Model inspired by similar models used for Computer Vision for unsupervised pattern discovery in symbolic music, allowing to give a description of repeated patterns at different hierarchical levels, although the hierarchy described in this context is a hierarchy between the detected patterns, rather than a hierarchy of the parts of a musical piece as usually meant in this context.

Instead of focusing on the segments of a musical piece, others can directly implement structural and hierarchical theories, such as those described in section 4.1, trying to give appropriate representations with ad-hoc algorithms and data structures. While the theoretical works cited in the previous section do not have the aim of describing algorithmic procedures to compute the hierarchies and the structures they describe, there is a variety of more computer science oriented work that try to formalize those ideas in a more machine-friendly way, or that implement similar ideas for automating structural analysis of music (Marsden et al., 2013).

Some researchers have applied grammar-based approaches to the computation of Schenkerian reductions, as grammars naturally allow the description of tree-like structures (Kassler, 1975; Mavromatis and Brown, 2004). One limit to this approach is the complexity and aboundance of the rules needed to guide the process, which results in non-feasible computations (Marsden, 2007; Marsden et al., 2013). Some researchers overcome this problem by emplying markovian approaches to filter the rules (Gilbert and Conklin, 2007), dynamic programming (Marsden, 2010), or other kinds or pre-processing of the melodic material (Kirlin and Utgoff, 2008). Another approach to making reduction more manageable is to use heuristic approaches, relaxing the goal of performing Schenkerian analysis and instead per-

forming more generic but more efficient melodic reductions. Orio and Roda (2009) performs melodic reductions by assigning weights to notes in a melody based on melodic and harmonic features and iteratively eliminating the notes with the lower weights. While the results cannot claim to be actual Schenkerian analyses, the obtained reductions can be useful to other applications.

GTTM is another well-known hierarchical theory of music. One early example of using some of the rules from GTTM in a computational manner is the software Melisma Music Analyzer (Temperley, 2004). In order to implement the entirety of GTTM instead, it is necessary to resolve some ambiguities that are reasonable for a musicological theory but cannot be allowed in a computational system. Hamanaka et al. (2006) proposed an extended version of GTTM (exGTTM) which was later implemented in software for computer assisted analysis (ATTA) or even for fully automated analysis (FATTA) (Hamanaka et al., 2007). More recently, deep learning was applied to learn the rules for grouping required by GTTM rather than explicitly implementing them (Hamanaka et al., 2017).

Since the goal in this chapter is describing structure for music generation, I'll close this discussion with some relevant proposals that come from that field. GEDMAS (Anderson et al., 2013) uses a top-down approach for structured generation, using Markov chains to generate the overall structure of a piece, but the melodic content itself is not part of this hierarchical structure and is generated at a later time. MorpheuS (Herremans and Chew, 2017) applies a structure to imitate a given piece, but there, structure is related to perceived tension, rather than repetition and reuse of melodic content. Deep learning techniques are often applied to melodic generation, and some researchers have given proposals on specific network topologies that can allow for more structured output (Chen et al., 2019; Zixun et al., 2021), but the main downside of these approaches, besides the high computational requirements, is the fact that the learnt structures are not easily interpretable by humans and reusable in other contexts. Herremans et al. (2015) propose explicit modelling of structural patterns for Ethiopian bagana songs (although the method could certainly be applied to other musical styles), and use markov chains along with optimization techniques to generate music that follows the given structural patterns. This proposal is the most similar to the

one described in this chapter, but the main difference is that structure is imposed with top-down patterns instead of emerging from the analysis of a corpus, as I propose here. Finally, Wiggins (2021b) provides an in-depth theoretical base for the relevance of this approach to music analysis and generation, but does not provide any practical approach to perform the proposed analyses.

4.3 | Three Tree-Based Representations for Music

4.3.1 | Corpus Description

Along with this work on music representations, I present a small corpus made of twenty-four baroque pieces that exhibit strong structural regularities. Though the size of the corpus is relatively small, it is ideal for top-down approaches to musical analysis due to its structural regularity.

The corpus is comprised of twenty-four allemandes (dance music originating from Germany, usually possessing even meter), written in 1768 by Gabriele Leone (sometimes referred to as Pietro Leone), a mandolin virtuoso from Naples. These pieces were originally included in a method for teaching mandolin to violin players. As such, despite the great technical ability of the author, the pieces are extremely simple and can be played by a novice. All of the allemandes are written for a mandolin duo, ideally having the first part played by the student and the second by the teacher, so all these pieces are polyphonic. In addition, a single instrument often plays chords, so either part may polyphonic on its own. Since the Neapolitan mandolin has only four strings, neither part ever has more than four simultaneous notes. The corpus is released under a Creative Commons license, in MusicXML format[1].

4.3.1.1 | Structure

Since I propose this corpus as a useful tool to study musical structures, it is worth describing what kind of structural regularities this corpus offers.

At a high level, there are some evident regularities, as follows:

[1] https://doi.org/10.5281/zenodo.5145348

- All of the pieces are divided into two sections, which I call the A section and B section.
- In 20 out of 24 pieces, both the A and B sections are eight bars long.
- In 22 pieces, the A section is repeated at the end, thus obtaining an "A-B-A" structure.
- 20 allemandes have $\frac{2}{4}$ meter.
- All pieces are in a major key and modulate in the B section either to a close tonality, or in only four of the allemandes, to the minor mode.

The following are the exceptions to above regularities:

- IV has each section repeated (A-A-B-B).
- XI has a 12 bars long A section.
- XIV has both A and B section last only 4 bars.
- XVIII has each section repeated and then the A section again (A-A-B-B-A).
- XIX has a 16 bar A section and a 24 bar B section.
- XXI has a 4 bar B section.
- VII, IX, XVIII, and XIX have $\frac{3}{8}$ meter.
- VIII is unique in the corpus in that it has anacrusis.

Besides the regularities in the macro-level form, within each of the sections there is frequent use of repetitions, transpositions, and imitation, both within a single part and between parts of the two instruments. These make it simple to distinguish four-measure long phrases that can be further divided into two sub-phrases. The relative simplicity of the melodic material makes it easy to distinguish the use of techniques such as repetition, transposition or inversion, making this corpus useful for algorithms that wish to detect such techniques. Moreover, the corpus has two voices, so there is still enough variance and description to extract meaningful information relating to harmony. One final peculiarity of this corpus

not strictly related to musical structure is that each piece was named by the author with an adjective describing the "feel" of the piece, like "The Joyful", "The Grumpy" or "The Fickle", and therefore could be used to research if specific musical techniques relate to the proposed emotions and expressions. Finally, the corpus is enriched with manually added chord annotations. While there are many MusicXML corpora available, very few present all the above characteristics. For this reason, I believe this corpus can represent a useful tool for many researchers, despite its small size.

4.3.2 | Representations

4.3.2.1 | Schenkerian Trees

The first representation describes hierarchical reductions of a melody. In this tree representation, each node is a note, and the children of a node are the notes that. can be reduced to the note in the node. Another way to see this, is to think of nodes as groupings of notes. We can call this tree a Schenkerian tree (Sk_tree), as it represents a set of iterated reductions of the given piece, inspired by what is traditionally done in Schenkerian analysis (Schenker, 1935) or in the Generative Theory of Tonal Music (Lerdahl and Jackendoff, 1985).

The algorithm used to obtain these trees from a input lead sheet was described in previous related work (Orio and Roda, 2009; Simonetta et al., 2018; Carnovalini and Rodà, 2019a), but it is worth describing here how the algorithm functions at a high-level.

First, a sliding window is created, twice as long as the shortest note duration in the input. That window passes over the melody, and whenever there are two or more notes present in window, the "more important" note is selected, given the harmonic context, tonality, and metric position of the notes. For example, in the context of a C major piece, over a G chord, a note G will be considered more important than a note F. The winning note is made as long as the window, and the other notes are discarded. When all the notes in the window have passed through the window, a new melody can be formed by all the notes that were not discarded. This new melody will have the same length as the original melody, but less

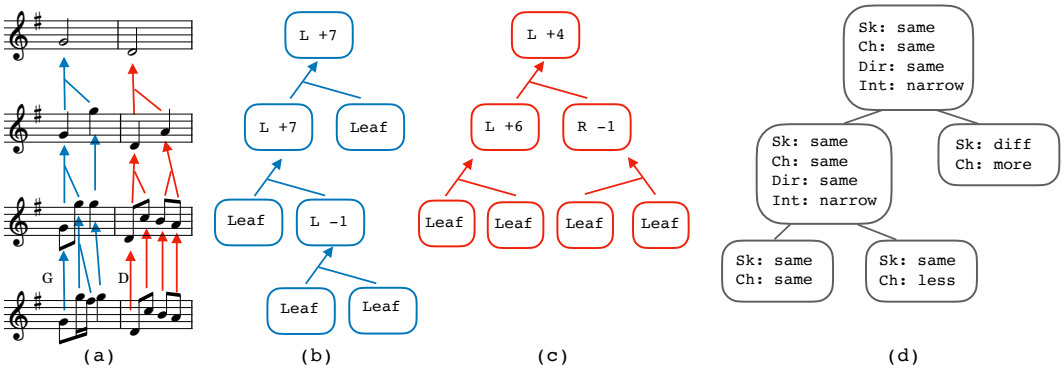


Figure 4.1: Example of the process building Sk_trees and a Diff_tree. (a) shows two bars of music being reduced with the Schenkerian approach. The arrow shows the notes that are kept in the successive reduction, and the line joining the arrow represents the reduced note. (b) shows the Sk_tree obtained from the first measure. Notice that the leaves correspond to the notes that are present with that duration in the surface melody, regardless of the level. (c) shows the Sk_tree obtained from the second measure. Notice that there is one node reporting `R`, as the reduction kept the note on the right, and not the one on the left as usually occurs. (d) shows the Diff_tree obtained by comparing (a) and (b). Notice that a leaf in this tree occurs whenever a leaf on either Sk_tree is found.

notes overall, and the shortest notes should now be longer than what previously was the shortest note. For instance, if the shortest note was a quarter note, the window would be as long as an half note. If such window passed over two quarter notes, one of the two would be eliminated and the other would become an half note: in this way, the shortest note would now be an half note. The size of the window is then increased to twice the shortest duration of this new melody, and the process is iterated until only one note remains.

These iterated reductions naturally form a tree (see Figure 4.1 over the letter a) where the nodes of the tree correspond to the notes of the melody, and each level of the tree represents a level of reduction. For the following algorithms, the tree was further transformed into a more compact representation by annotating in each node how a note is expanded in the lower layer, describing the interval between the children (over the letters b and c in Figure 4.1).

4.3.2.2 | Difference Trees

The above algorithm serves as a way to simplify the melodic material in piece to make it easier to find regularities, but does not actually compare different segments of a piece to find such regularities. The Difference Trees (diff_trees) serve this purpose: they compare the reductions made in the Sk_trees and annotate the actions required to transform from the first tree to the second. The comparisons can be made between any segment of the original piece, but should only be made going forward, i.e. comparing one segment only with segments that come after that, due to the fact that this operation is not commutative.

The algorithm takes as input two Sk_trees, and proceeds as follows: the root nodes of the two trees are selected (called R_1 and R_2), and a new node is constructed to be the root of the output Diff_tree. Considering the direct children of R_1 and R_2 in their respective trees, in the node of the Diff_tree the following features are annotated:

- `Sk`: Schenkerian direction. If R_1 and R_2 come from the same position in the children notes (left or right note), annotate `same`, otherwise `diff`. A leaf note counts as expanded to the left.

- `Ch`: Number of children. Annotate if R_2 has the `same` number of children as R_1, or if it has `more` or `less`. If the result is same, compare the following features regarding the children:

- `Dir`: Interval direction. Annotate if the interval described in R_1 has the `same` direction as the one described by R_2 or not (`diff`).

- `Int`: Interval width. Regardless of the direction, annotate if R_2's interval is more `narrow`, more `wide` or the `same` as R_1's interval.

Then, the first child of both is selected, and the algorithm recursively repeats on all children until there are no more children or if they have a different number of children (a leaf is found), in which case the recursion stops as it is not possible to operate the comparison anymore. Figure 4.1 shows the result of this process.

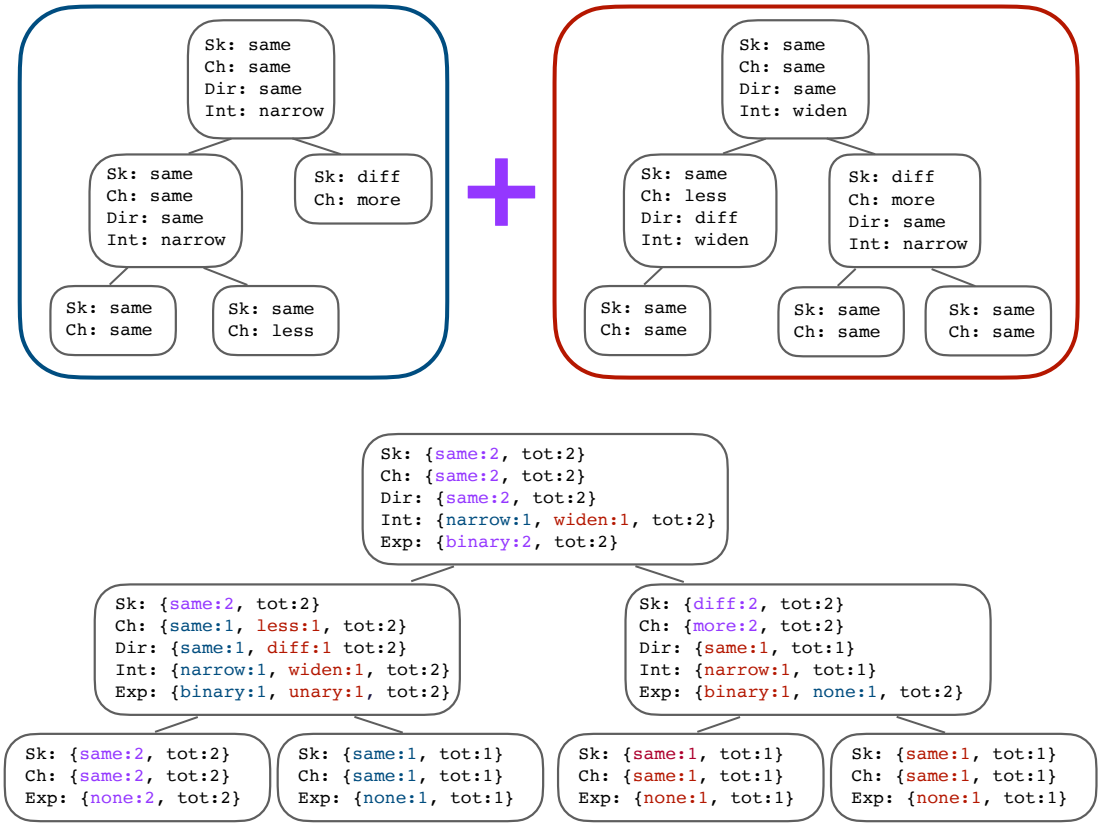


Figure 4.2: A simplified Abs_tree built from the two Diff_trees on top. For readability, the tree reports frequencies of occurence and the total number of observation rather than the probabilities that need to be computed with the Bayesian Estimator. The colors represent the tree from which each value for each feature comes from: blue for left-side tree, red for the right-side one, and purple for those values that are found in both.

4.3.2.3 | Abstraction Trees

For each input piece, a set of $\frac{(n-1)(n)}{2}$ Diff_trees are produced, where n is the number of segments the input piece is divided into. This number is due to the fact that the segments are not compared with themselves nor the segments prior to them, but only with segments that come later in the musical piece, so not all the n^2 possible comparisons are performed. Each tree is then labelled to indicate which segments are compared in that tree. Once the Diff_trees for a set of pieces are produced, the Diff_trees that share the same label

(i.e., that refer to the same segments of different pieces) can be joined together into a single tree, that abstracts the general development of that particular comparison. For this reason, we can call this representation an *Abstraction Tree* or *Abs_tree*. The procedure works as follows: a new node which will be the root of the Abs_tree is created. Starting from the root of all considered Diff_trees, for each of the possible features, the new node annotates all the possible values that the feature assumes in all the given Diff_trees and the number of occurrences of those values. The new node also annotates how many children the roots of the given Diff_trees have, as a new separate feature. Then the algorithm repeats recursively as long as at least one Diff_tree node has at least one child node. Once the recursion process is complete, it is possible to compute the probability of each value v for each feature in each node, using the following Bayes Estimator (Schürmann and Grassberger, 1996):

$$p(v) = \frac{occ(v) + \beta}{tot + \beta * size} \tag{4.1}$$

where $occ(v)$ is the number of occurrences of the value v, while tot and $size$ are respectively the total of samples for that feature and the number of possible distinct values for that feature. β was set to 1. All the features for which tot is less than a certain threshold (I used 5 in the experiments described below) can be removed as those features would entail too little information about the corpus. The nodes that remain empty because of this can be removed as well. Figure 4.2 shows a simplified example of an Abs_tree built from two Diff_trees. This process basically creates a probability distribution for each node, consisting of a set of stochastic variables depending on the features present in the Diff_trees. Because of this the Abs_tree is similar to a Markov Model, but rather than having the probability distributions vary in time depending only on the previous state, the only feature that determines the distribution is the position on the tree. For this reason, it would be misleading to think of an Abs_tree as a chain or an automaton, and it is best to only view it as a static probabilistic description of a musical corpus.

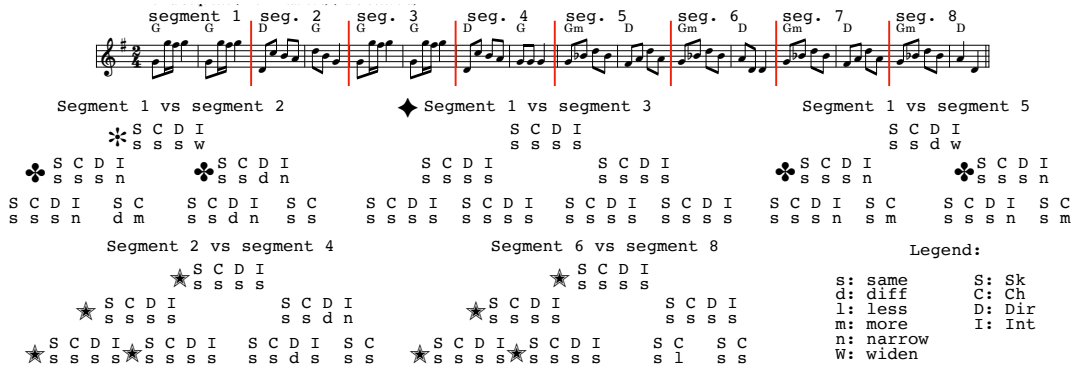


Figure 4.3: Diffrence trees computed on allemande V, in compact form. The nodes below the third level were omitted.

4.3.3 | Applications

In this section, I will demonstrate, through example applications, the utility of the proposed abstract representations. To do so, I will apply the explained structures to tasks relevant to computational musicology and music information retrieval. It is worth noting that while these applications were used to test the effectiveness of the proposed representations, these do not represent the actual goal of this work. Indeed, the analysis performed by this system is not actually a complete musicological analysis, but rather a way to extract meaningful information capable of helping a music generation system.

4.3.3.1 | Musical Analysis

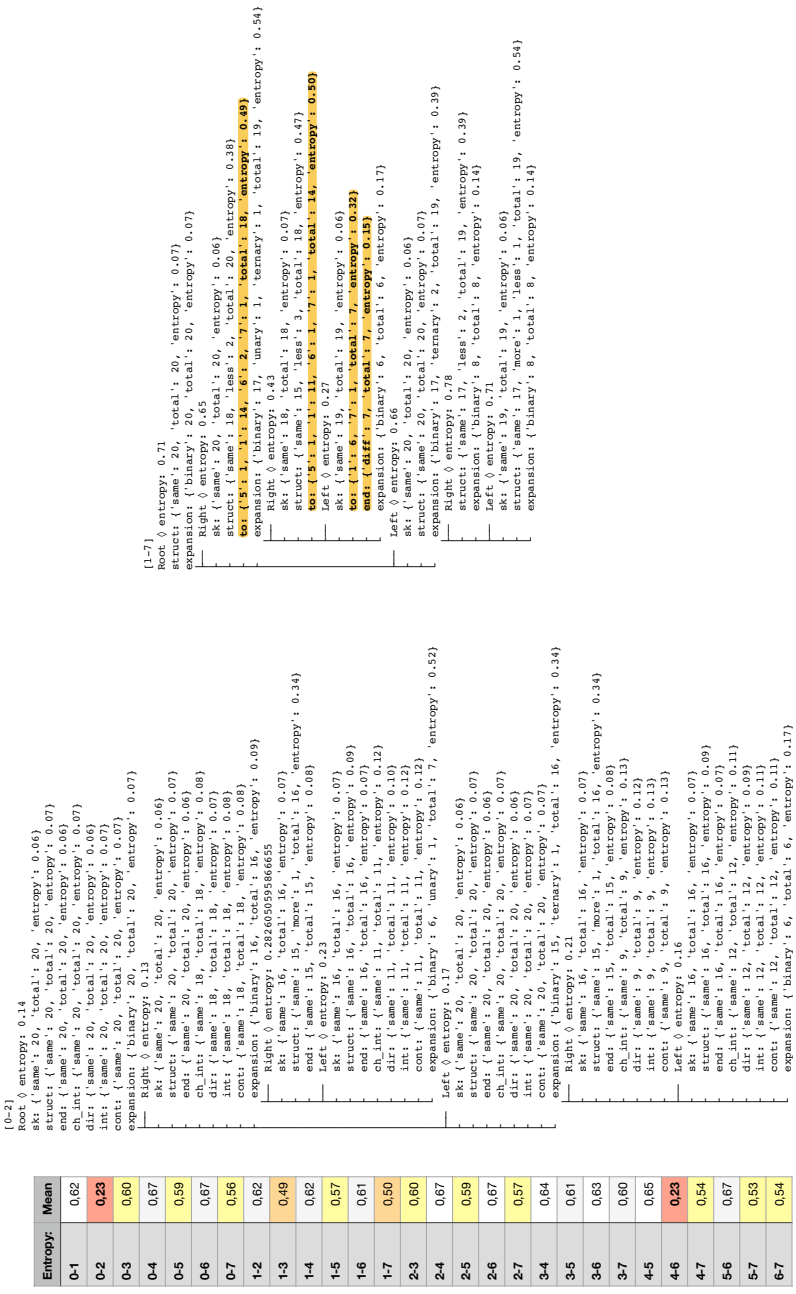
Figure 4.3 shows some Diff_trees computed from one allemande taken from the corpus: following are some information that can be inferred by the encoding. In the first tree, comparing segments 1 and 2, the highest level shows an interval that is widened (see ✻ in the figure). The highest level of reduction is strongly reliant on the harmonic development. This widening is connected to the fact that the second segment is more harmonically diverse than the first. Conversely, the second level of both that tree and the one comparing segments 1 and 5 (✤) sees only narrowing intervals. This level is less reliant on the harmony but rather gives an indication of the general melodic contour, and indeed segment 1 (and it's repetition segment 3) is the one with the widest extension in the piece. The mentioned

repetition is captured by the tree comparing segments 1 and 3 (✦), that shows no difference at all across all features. The comparison between segments 2 and 4 and the one between segments 6 and 8 (★) also show a repetition (of the first measures of these segments), but also shows the ending variation, that is to be expected from the ending of a phrase/section.

4.3.3.2 | Regularity Detection within a Corpus

The above introduced Abstraction Trees can be inspected to find regularities within a given corpus. Of the twenty-four that make up our corpus, the twenty that have sixteen measures in total were considered here, to make comparisons easier thanks to the equal length. All the pieces were divided into two-bars long segments and the abstraction trees pairwise comparing the segments were built as described above. The procedure produces a large amount of data which is difficult to interpret on its own, but the tools of information theory can help find the most relevant features. For each feature in each node, it is possible to compute the normalized entropy (efficiency) of the probability distribution it describes, which gives an useful indication of the importance of that feature within the corpus. The lower the entropy, the more strictly that feature describes a recurring element in the corpus.

For example, as can be seen in Figure 4.4, looking at the mean normalized entropy of all the constructed abstraction trees, it becomes evident that the tree comparing segments 0 and 2 (shown in figure) and the one comparing segments 4 and 6 are the most regular ones. In those tree, almost each feature is set to "same", meaning that there are little or no differences between the above mentioned pairs of segments. Indeed, the phrases in this corpus tend to repeat after 4 measures (the distance between the start of segments 0 and 2), and that is captured by the abstraction tree. Moreover, while the phrases repeat, their ending is varied to make for more definitive phrase endings. This is captured in the abstraction tree comparing segments 1 and 7 (shown in figure) where the left side of the tree shows a repetition like the one described above, but in the right side of the tree the most relevant feature is the one describing the ending grade, which is usually the tonic, as expected from the closing of a musical period.



Entropy:	Mean
0-1	0.62
0-2	**0.23**
0-3	0.60
0-4	0.67
0-5	0.59
0-6	0.67
0-7	0.56
1-2	0.62
1-3	0.49
1-4	0.62
1-5	0.57
1-6	0.61
1-7	0.50
2-3	0.60
2-4	0.67
2-5	0.59
2-6	0.67
2-7	0.57
3-4	0.64
3-5	0.61
3-6	0.63
3-7	0.60
4-5	0.65
4-6	**0.23**
4-7	0.54
5-6	0.67
5-7	0.53
6-7	0.54

Figure 4.4: A table summing up the mean entropy of each abstraction tree derived from the corpus, and some examples of abstraction trees as a text output of the software. Only the first three levels were kept for readability. The labels of the tree represent the compared segments: for example "0-1" means that the first two bars of a piece are compared to measures 3 and 4, since in this case each segment was two measures long.

4.3.3.3 | Genre Distinction

The following example shows that our system is capable of distinguishing different corpora based on their structural aspects. While this is not a genre detection system, it shows that structural information can vary between different genres and that the abstraction trees can capture these differences. To do so, this application leverages another metric commonly used in Information Theory. While entropy is related to regularity in a probability distribution, Information Content gives an indication of how unexpected a certain outcome is with respect to a given probability distribution. Since musical cognition is strongly related to expectation (Huron, 2008), this metric becomes a relevant indicator when analyzing musical pieces (Pearce and Wiggins, 2006). In this experiment, a set of abstraction trees from the 20 allemandes taken from our corpus were built, along with the difference trees from a set of 20 reels from the Nottingham Dataset (Foxley, 2011), and from 20 jazz pieces composed between 1921 and 1930 taken from the EWLD corpus (Simonetta et al., 2018). The abstraction trees contain probability distributions for each feature in each node, while difference trees can be considered as outcomes for the same features. This means that for each feature it is possible to compute the information content. To give single measure of the total information content of a difference tree compared to an abstraction tree, the mean of all the features in a node is computed to give the information content of a single node, and the mean of all the information contents across nodes is computed to give the general information content of a tree. This latter mean is also weighted by the mean entropy of the nodes, and by an added coefficient that makes nodes lower in the tree less important than nodes in the upper part of the tree ($depth\_k$ in the formula below). The total formula is described below, where $p(diff\_tree\_feature)$ represents the probability $p(v)$ (computed according to the estimator 4.1) of the value v found for the considered feature f in the diff_tree node.

$$ic(tree) = \frac{\sum_{node \in tree} ic(node) \frac{1}{ent(node)} * depth\_k(node)}{\sum_{node \in tree} \frac{1}{ent(node)} * depth\_k(node)} \tag{4.2}$$

$$ic(node) = \sum_{f \in node} \frac{-\log_2(p(diff\_tree\_feature))}{ent(f)} \tag{4.3}$$

$$ent(node) = \frac{\sum_{f \in node} ent(f)}{number\_of\_features\_in\_node} \tag{4.4}$$

$$ent(f) = \sum_{v \in alphabet(f)} -\log_2(p(v))p(v) \tag{4.5}$$

Figure 4.5 shows the results of the comparisons. Since computing the information content of a piece included in the abstraction tree would be an unfair advantage, an approach similar to a k-fold validation was used: the allemande corpus was split into four parts of 5 pieces, and the information content of each piece was computed with respect to the abstraction trees built solely on the 15 pieces outside the considered allemande's group. This means that there were actually four sets of abstraction trees built each on a different subset of 15 pieces: the values for the other two groups (Jazz and Nottingham) were computed on all the four sets and the mean is reported.

The results clearly show that this approach is capable of detecting the structural differences between the corpora. The allemandes show a strong structural regularity, that is not found in the other pieces. As expected, the reels from the Nottingham dataset are less unexpected than the jazz pieces, since they too have some structural regularities that are not always found in jazz pieces. It is worth noting that being based on difference trees, what this system captures is the general structure of the piece and how much reuse of melodic material is present, rather than comparing for instance the regularities in the melodies and how typical they are for each genre. For this reason, the system in itself could not form an actual genre detection system, although this metric can possibly constitute a complemen-

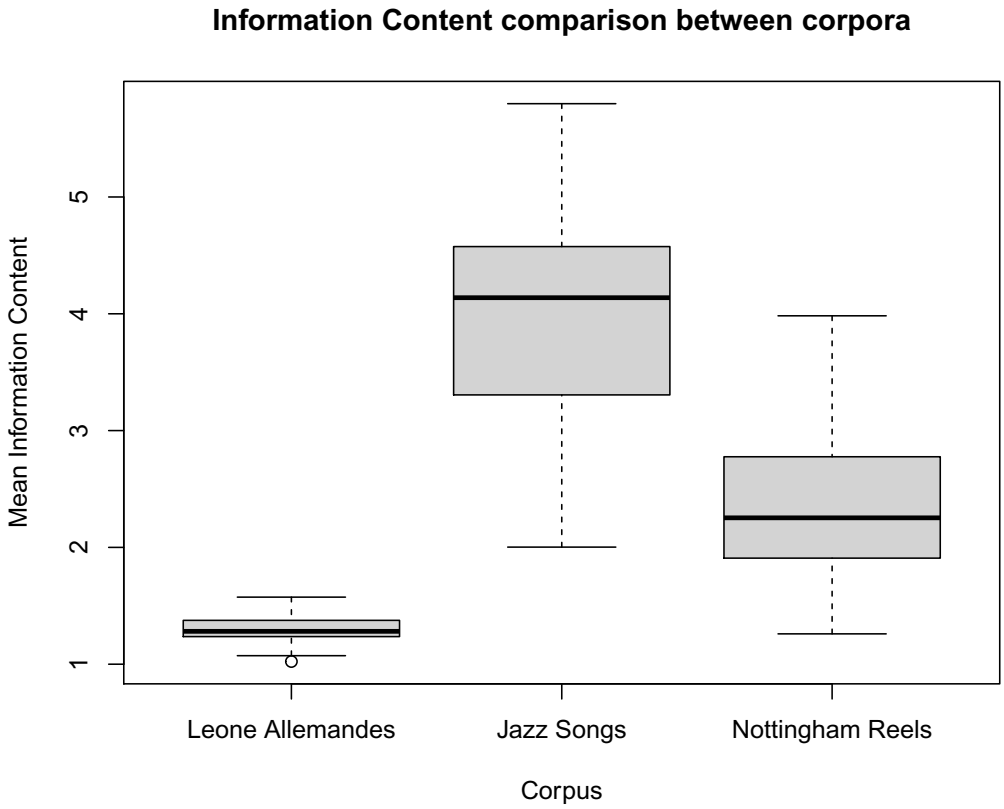


Figure 4.5: Comparison of the mean information content computed from each of the three sets of twenty musical pieces. Mean refers to the mean of the trees of a single piece, rather than the mean of an entire corpus.

tary indicator that could be used in combination with other approaches in a genre detection system.

4.4 | Structure-Aware Style Imitation

The representations described above only serve as an analysis tool, and not as a generation tool. Moreover, most of the information that is represented is related to structure, and do not prescribe ways in which a melody can be constructed. Yet, its analytical power can be used to discriminate typical structures within a certain genre, as shown in the Genre

Distinction task explained above. The music generation approach I describe here uses the same idea of using Information Content to find typicality of structures, but embedded within a genetic algorithm. This algorithm first generates some melodies, and then uses Information Content as the fitness function to choose those melodies that are more fit to be a continuation of the starting melodies given the genre of the analyzed corpus, so that the final generated piece will show a satisfactory long-term structure.

The generation process functions as follows:

Step Zero: Construct the required data structures from a given corpus.

Step One: One starting segment lasting two bars is generated.

Other segments are generated with the following steps, until the specified length is reached.

Step Two: Generate some segments using the same procedure as the one above, that constitute a pool of candidates. Add to the candidates pool all the segments that constitute the piece so far.

Step Three: Mutate the pieces in the candidates pool. For each of the candidates, 5 mutations are generated, by selecting at random the operations: note deletion, note insertion, change of rhythm, change of pitch.

Step Four: Evaluate the Information Content of all the candidates. If for one continuation the information content is below a certain threshold, a winner is found. Otherwise, the pool is reinitialized with the 5 best results and 5 new segments, and the process repeats from Step Three.

The following sections will describe the above points in more detail.

4.4.1 | Corpus Analysis and Generation of Melodic Beginning

Step Zero

For this process to function, it is necessary to have a input corpus that must be analyzed following the process described in the prior sections of this chapter. Each piece in the given corpus is divided into segments lasting two measures, and a Sk_tree is built for each segment. For each piece a set of Diff_trees are constructed, and finally the Abs_trees that represent the entire corpus can be computed. In addition to those structures, that only deal with structural representations, this algorithm extracts some additional data for the construction of melodies.

The first of such structures is a first-order Markov chain learnt from the notes at the topmost layer of the reductions operated on the segmented corpus. After the reduction process for the construction of Sk_trees, each segment is reduced to only one note, and thus the piece is reduced to a sequence of N notes where N is the number of segments in the piece. A first order Markov chain is built to replicate this sequence, that is needed as input for the following operations. We can call this chain the `Top_chain`. The Top_chain also saves the probability for each of the symbols in the chain to be found as the reduction of the first segment of a piece, by counting how many times each symbol is found in the first segment's reduction.

Additionally, a set of first-order Markov chains is constructed by analyzing the pitch sequences in each of the segments. Instead of learning just one chain for these melodies, a chain for each of the symbols found in the Top_chain is built. This is done to preserve the fact that melodies can have different developments depending on the underlying structure of the piece, represented here by the reduced note. Notice that prior to the segmentation, as required by the algorithms for the construction of the tree structures, all the pieces are transposed to the key of C Major/A minor, so that the reduced note is significantly comparable across all pieces. Additionally, the surface chains also save the probability for each symbol in the chain to be found at the start of the melody, by simply counting the occurrences of each note at the beginning of a segment. We can call these chains `Surface_chain`, and, for example, we can refer to the one built for those segments that are reduced to the note C by writing `Surface_chain[C]`.

The above Markov chains only consider pitch in their alphabets. This is fine for the Top_chain, that describes abstract notes, but for the construction of melodies it is necessary to consider rhythm as well. In this work, a list of all the rhythmic sequences found in the segments of the corpus were collected, so that it is possible to select one at random when adding a rhythm to a melody.

Step One

All of the above can be computed beforehand, and the resulting model can be used to generate any number of pieces. Once such a model is available, the generation process starts by creating a sequence of notes via the Top_chain. The resulting piece will have twice as many measures as the notes generated in this sequence. The resulting sequence is called the Top_sequence The first note from which to start the Top_sequence can be chosen beforehand (C is a reasonable choice) or it can be chosen at random using the saved probability for the beginning symbol.

The first segment is then generated. To do so, a rhythmic pattern is chosen at random from the rhythmic list. Then the surface chain is used to generate an appropriate amount of notes, starting from a note chosen at random using the probabilities saved for the beginning note. For this operation the surface chain labelled with the first note of the Top_sequence is used (`Surface_chain[Top_sequence[0]]`).

4.4.2 | Generation of a Pool of Continuations

Step Two

Each of the segments beyond the first one are generated via a Genetic algorithm. A pool of candidates is built for this process, and a winner is selected within this pool. The pool is initialised with five segments that are generated using the same process described above, i.e., by choosing a rhythm from the model's rhythmic list and generating pitches selected with the appropriate Surface_chain. Notice that the selection of the chain depends on the note found in the Top_sequence in the position that corresponds to the segment that is

being generated. Additionally, all of the segments that were already included in the piece are added to the pool, as well as a transposed version of the same in the case where the note in the Top_sequence differs. For example, if the third segment is being generated, and the Top_sequence looks like this: [C4, D4, C4], both the first and the second segments are added to the candidates pool, but a copy of the second segment transposed down by 2 semitones (the interval between D4 and C4) is added as well. This allows the algorithm to create melodic progressions (should this copy be chosen as a winner).

Step Three

For each of the candidates in the pool, 5 mutated segments are generated. The possible operations performed to mutate the segments are:

Note deletion: randomly select one note from the segment and delete it. The length of the previous note (or the following, if the deleted note was the first) increases by the length of the deleted note.

Note Insertion: randomly select one note from the segment. That note's length is halved, and a new note is generated (using the appropriate Surface_chain) to fill the void.

Change of Pitch: randomly select one note from the segment and delete it. A new note is generated to fill the void, using the appropriate Surface_chain if there is a preceding note or using the probability for the beginning note if the deleted note was the first.

Change of Rhythm: a new rhythm is chosen from the rhythmic list. If the previous rhythm had more notes, delete the last notes that would not be used. If the previous rhythm had less notes, add notes to fill the void by using the appropriate Surface_chain.

The amount of mutation operations applied to one segments varies with the number of the mutations that were already generated. That means that the first mutation for the segment has one operation, the following two, and so on until the last one that undergoes five operations.

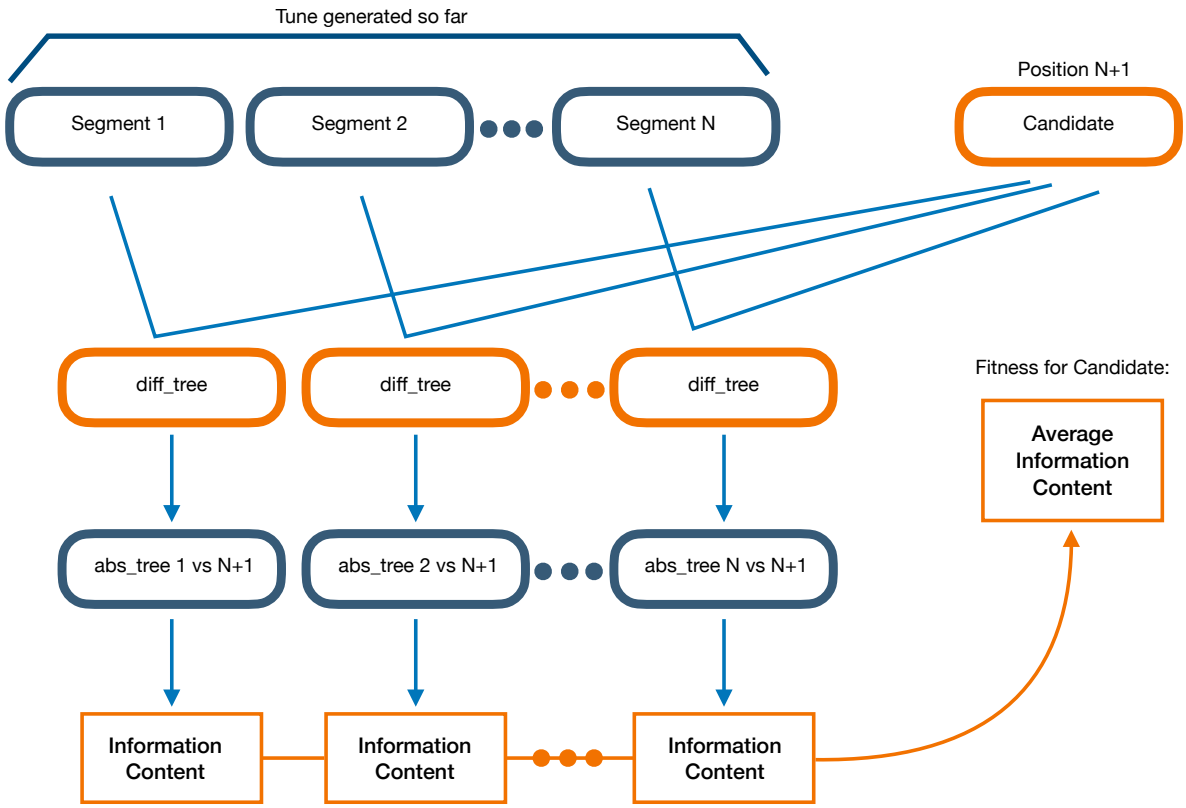


Figure 4.6: The process of the construction of the fitness for a new candidate. For each of the already generated segments, a diff_tree must be constructed which will be used to compute the information content with respect to the relative abstraction tree.

4.4.3 | Genetic Approach to Select Continuation

Step Four

Evaluate the Information Content of all the candidates. To do so, a diff_tree between the candidate and all the segments in the piece so far must be constructed, and those diff_trees are evaluated against the abs_trees learnt on the corpus (see Figure 4.6). If the average information content is below a certain threshold, a winner is found.

Otherwise, the 5 best results are kept in the candidate pool and the others are eliminated. Five additional new segments are generated with the process described above, and are added to the pool. The process repeats from Step Three, meaning that each of the candidates in the new pool gets mutated five times before the evaluation restarts.

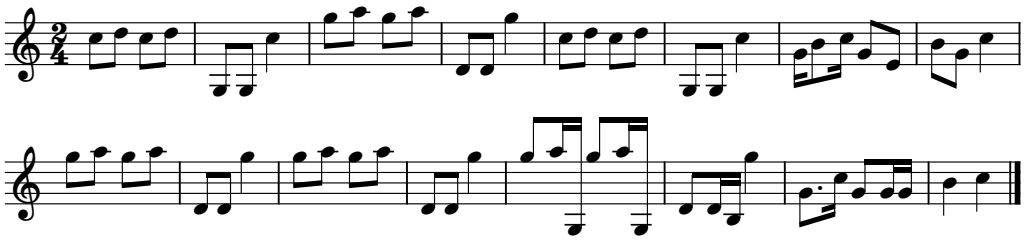


Figure 4.7: One allemande generated with the system.

After ten generations, if the desired threshold is not reached, the computation is halted to avoid plateaus.

4.4.4 | Example Result

Figure 4.7 shows one example of what the generation system described above is capable of creating. It is possible to see that the system managed to create a tune with a similar structure to the corpus of allemandes. For example, the first two measures are repeated in the fifth and sixth measures (segment 3), as was the case for the allemande reported in Figure 4.3. Similarly to that allemande, the ending of measure 8 and 16 end on the tonic, variating from the previous segments. We can also see that the second part of the allemande is somewhat more free than the first part, with measure thirteen and fourteen being rather different from the rest. This reflects the fact that in the corpus the second section of the tunes is less regular than the first half.

The main shortcoming of this generated allemande is the presence of large melodic intervals, which are indeed present in the corpus (otherwise those could not have been generated by the Markovian approach), but are usually justified by melodic progressions leading to those intervals, or happen with longer notes that allow the player to move the hand the required amount in time. These intervals make the generated allemande rather unfriendly for a beginner player, which is not the case for the ones in the corpus. The use of an extremely simple algorithm for melodic generation (first order Markov chains) may be the cause of the unfitting melodies, and this could possibly be solved by using more refined

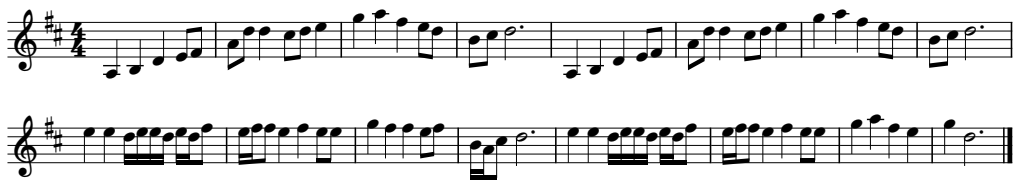


Figure 4.8: One reel generated with the system.

algorithm in the segment generation phase, as well as in the mutation phase. On the bright side, the fact that the final output is reasonable despite the use of such a simple approach to melodic generation shows the potential of the structural fitness system.

Figure 4.8 shows one additional example. In this case, the system was primed with reels from the Nottingham Dataset. In this example it is possible to clearly distinguish and A and a B sections. As customary in this style, the B section shows more embellishments than the A section, but remains coherent with the first section especially in how the period ends: measures 15 and 16 are a variation of measures 7 and 8, tying together the two sections in their endings, allowing repetition of either or both sections in any order. As in the previous example, the phrases end on strong grades, in this case always on the tonic.

4.5 | Discussion

4.5.1 | Main Findings

This chapter was devoted to representations of music that consider its structural aspects, and to music generation based on such representations.

The first contribution of this part is the implementation of an algorithm for the reduction of melodies inspired by Schenkerian Analysis, based upon work present in the scientific literature. While this does not represent innovative contributions, it is the main building block for the following representations, and already follows the principle of being an algorithm based on cognition of music. The general consensus is that the idea of grouping musical passage in a hierarchical fashion is indeed how we perceive melodies in our brain (Temperley,

2011).

The main contribution is the three-tree based representation system, which uses the trees built with the above mentioned algorithm as the input for further computations. The relatively simple approach described in this thesis allows to give probabilistic descriptions of how a piece, or even a set of pieces, unfold over time, rather than giving a static description of the entire pieces similarly to what Schenkerian analysis did. Moreover, given the use of probability distributions in the Abstraction trees, it was possible to use Information Theory concepts to pragmatically analyze the pieces, in a way that is once again rooted in human cognition. The use of information content is especially related to unexpectedness, which is one of the main aspects of music cognition (Pearce and Wiggins, 2006; Pearce, 2018; Wiggins and Sanjekdar, 2019). While this representation cannot capture the entirety of information included in a musical piece (nor it is meant to), it offers a way to deal with the structural aspects of melodic formation, which are sometimes overlooked by research in Computational Creativity and is generally considered an open problem.

The representation system was applied to the goal of Music Generation, to create folk music showing convincing structure. The limits of this representation, that does not consider harmony, rhythm, and other key features of music, required the use of additional algorithms and knowledge representations, taken from literature. While these additional methods were not particularly innovative nor sophisticated, the use of information content applied to the structural representations allowed for convincing results. A genetic approach to generation was used, where the information content was at the basis of the fitness function, to select continuations to a generated melody that resulted in a plausible structure given the learnt corpus. Genetic algorithms are not commonly used in recent literature for music generation, and some question their usefulness within this context (Phon-Amnuaisuk et al., 1999; Wiggins et al., 1998; Carnovalini and Rodà, 2020). Yet, in this work I gave an example of how this approach can be useful when dealing with complex features (such as information content) that cannot be easily imposed by construction on a given melody.

4.5.2 | Scope of the Representation

While explaining and exemplifying the representation system that constitutes the main proposal of this chapter, a series of design choices were made, that influence what kinds of music are fit to be represented with this system, and what kinds of uses can be made of these data structures. It is worth discussing here what currently can be effectively represented and what would instead require different approaches.

The first and foremost limitation is that this representation system, being based on a Schenkerian-like approach, is limited to the representation of tonal music. Any genre of music that makes strong use of atonality cannot be meaninfully analyzed and represented in this manner. Going even further, the system requires the musical corpus to be represented in symbolic systems based on western score notation, so any kind of electroacoustic music, microtonal music, certain kinds of ethnic music, or any kind of music that cannot be represented via score notation cannot be represented, even if it still was tonal music. While these are serious limitations, tonal music still represents a vast majority of the western music production and being capable of only representing tonal music is a reasonable limitation.

Even within the scope of tonal music, it is possible to further characterize the limits of this representation system. As stated above, this system is strongly rooted in the concept of tonality, so music genres that use tonality but also modulate often, as modern jazz, might be less efficiently represented. More importantly, this system is based on structural aspects of music, and regularities within the melodic content of a corpus. If a musical genre has little structural regularities in itself, it becomes harder for this system to infer useful information.

The examples provided in this chapter deal with baroque and folk dance music. These are perfect examples of music that presents strong structural regularities and lies within the context of tonal music. Yet, it would be wrong to assume that this is the only context where the presented representation can funciton. The allemande corpus chosen for this study as its structural regularities made it easier to design the representation system, but the algorithm in itself can be applied to any situation where melodic reuse is relevant, which is true in most popular and baroque music, but also in many other contexts where standard

forms are used.

If one were to try and overcome these limitations, the general approach of building difference trees and abstraction trees could still be useful. More specifically, if one were to substitute the Schenkerian-like analysis of the corpus with other kinds of analyses, possibly to better describe different kinds of music, the hierarchical comparison between analyses and the subsequent statistical study of the comparisons can lead to similar structural representations, but for different musical contexts.

Another limit to the presented approach is that it uses a fixed segmentation approach, which can be set by the user and that we set to two measures for the generation. While this is surely a limit, one might think that this implies that only repetitions that have that exact length can be detected, where instead it is possible to find both longer and shorter relationships, since the difference trees will compare schenkerian trees across the entire piece, and can find relationships at different levels in the reductions meaning that different lengths can be considered. On the other hand, this fixed segmentation is still limiting in the fact that a piece could be more reasonably divided into variable-length segments, and most importantly the "phase" might be different across pieces. The system still functions regardless of this limitation, but further work in this direction could lead to better results in the future. Related to this, this study was limited to pieces having the same duration (expressed in measures). Since the systems mainly deals with period-level structures, it can be appropriate to detect periods and apply the proposed structures to periods, while larger-scale aspects could be dealt with another abstraction tree built to function with larger segments, which could be adapted to allow for different lengths.

One final limitation of the abstraction trees is that they require to be built starting from a corpus of relatively coherent pieces, as it will try to find regularities within the corpus. If the pieces differ too much, the result might lead to distributions too close to the uniform to be useful for style imitation. Further research might be able to allow clustering based on the difference trees, so that different abstraction trees can be built to describe the different cluters.

To conclude, our system for music generation based on style imitation also has some

limitations that were describe above. Most importantly, the melodic material it generates is sometimes unsatisfactory. The first step towards a better system would be the implementation of more advanced methods for melody generation, possibly integrating neural networks and deep learning in this phase.

5

Conclusions

5.1 | Summary and Discussion

In this thesis I presented the results of my research on some aspects of music generation, related to the more general field of Computational Creativity.

I introduced the field in Chapter 2, presenting a review of the state of the art on both Computational Creativity and on Music Generation Systems. In the final part of that chapter, I discussed some open challenges of the field that need more consideration from researchers. The rest of the thesis focuses on two of those challenges, namely Social Interaction and Structure in music generation.

Chapter 3 illustrates a serious game for social interaction between two players inspired by Music Therapy. The game requires two users to improvise a rhythm on MIDI pads, and adds a musical augmentation to reward their interaction, synchronized with their rhythm.

The system uses an algorithm for real-time detection of meter and tempo, which was implemented based on the strict requirements of the system. The algorithm takes a set of rhythmic notes (i.e. events that have onset time and force/velocity, but not pitch) and estimates the tempo by constructing a signal based on those events, and calculating the correlation between the signal and itself when shifted in time. The shift value that results in the highest correlation is chosen as estimated tempo. The meter is then computed by calculating the correlation of the signal with ad-hoc pre-built signals representing different

meters. Using these two estimates, the system can also predict the onset time of the next measure in the improvisation. The serious game uses the information computed with this algorithm to synchronize music generated by the computer with the interaction between the players. Depending on how well the players are interacting, the music also changes in intensity and instrumentation. The quality of this application was evaluated via quantitative metrics and with a questionnaire to test users. The testing show the potential of this application, both as a way to design rhythm-based human-computer interactions, and as an example application of musical co-creativity where computer-generated music is at the service of human users in a therapeutic context.

Chapter 4 describes a system for representing structural information of music based on three tree-based representations, and a system that leverages the same representations to generate music with satisfatory long-term structure.

The representation system accepts a corpus of musical pieces. Each is divided into segments, and each segment is reduced utilizing a Schenkerian-like approach. This reduction iteratively eliminates notes from the original segment, creating a tree where each node represents how notes were reduced. The trees resulting from these segments are compared with each other, creating a second representation called Difference Tree. In this representation, each node describes how the reduction operated in one segment differs from the reductions operated in the compared segment. Finally, by joining Difference Trees created from the various pieces in a corpus, an Abstraction Tree can be built. This represents the probability distributions of the various operations described in the Difference Trees. In practice, it represents the probability distributions for all the structural developments within a corpus. Using Information Theory concepts, namely Information Content, it is possibly to describe how unexpected (with respect to the given corpus) a structure of a musical piece that was not included in the corpus is. This was used as a fitness function within a genetic music generation system. This serves as a way to choose possible continuations for the piece that is being generated. Doing so, the system ensures that the chosen continuations are structurally plausible. To generate the musical material that is then evaluated

in this genetic approach, the system uses Markov chains to generate pitch and dictionaries to generate rhythm. While the results present unusual intervals, the algorithm is capable of generating pieces that show satisfactory structure despite the simplicity of the melody generation process.

The work presented in this thesis follow two main directions that were chosen as the result of a review of the state of the art in Music Generation. These two directions deal with social aspects of music and the utility of using computer-generated music for humans on one side, and with the description od+f structure on the other. The two aspects, while apparently rather diverse, both contribute to the advancement of music generation and can possibly lead to advancements in creativity of musical systems as well. The general position that guided this work is that, in order to obtain a form of creativity that is recognized as such by humans, it is necessary to start from human cognition and to develop algorithms and representations that reflect how humans view and perceive music. This position naturally guides towards co-creative implementations of music generation systems, as in the case of the presented serious game. Other researchers might suggest that such a human-centric view of creativity is limiting, and that we should explore other concepts of machine-based creativity (Colton et al., 2020). I would argue instead that to ensure that Computational Creativity is useful to humans, we must keep the focus on human intelligence, human cognition, and human creativity. Indeed, some definitions of Computational Creativity explicitly require humans to define if something is creative or not (Wiggins, 2006, 2021a).

One reason for which I suggest keeping the human at the centre is that generating computer music that is more similar to our music and that listeners can perceive as "human" makes it easier to allow for human-computer co-creation, ideally exchanging ideas with the computational composer (Sturm and Ben-Tal, 2021). Another reason is that, when we look at a broader picture beside musical applications, we can see that in most long-term goals of Computational Creativity, where creativity is embedded in any intelligent system, the requirement for human-like creativity is the only one that makes sense. For example, in the context of affective computing, if creativity is needed for us to relate more easily to

a computational system, the model of creativity must be based on the one we relate to, which is the human one. In more pragmatic or scientific applications, creativity can be used to guide human reasoning (for example in creativity applied to math), and thus the involved creativity must be akin to the human one so that the reasoning is understandable by us. Finally, it is worth considering that most definitions of creativity see it as a two sided feature, balancing novelty and value. While novelty leaves a lot of possibility for exploration beyond what would be normally considered human, value is a concept that is still strongly related to our benefit and idea of what is valuable. Building a creative system around human-like reasoning and cognition could in principle ensure that the results are valuable from a human viewpoint.

5.2 | Contributions

The contributions of the work described in the thesis are the following:

- A review of the state of the art of Computational Creativity and Music Generation systems, underlining open challenges in the field (published as open access article)
- The study of social musical interactions through a real-time game:
  - A system for the detection of meter and tempo in real time based solely on rhythmic input seen as onset time + velocity (delivered as open source code)
  - A system that generates a real time accompaniment for a rhythmic input (delivered as open source code)
  - A serious game that analyzes a rhythmic interaction between two players and generates musical accompaniment that fits their rhythm and evaluates the quality of their interaction (delivered as open source code)
- The study of structure in music through specialized representations:
  - A dataset of 24 allemandes for musicological research on structure (delivered as open data)

- An algorithm for the reduction of melodies inspired by Schenkerian Analysis (described in academic publications)
- A method for music representation based on the comparisons of tree representations of a musical piece (described in academic publications)
- A system for computing such representations, and for computing cognitively relevant statistical features of a musical piece in comparison with a given corpus using such representations (delivered as open source code)
- A system for generating music that imitates the style and structure of a given corpus, based on the same statistical features (delivered as open source code)

5.2.1 | Publications

The projects described in this thesis led to the publication of one open-access journal article and eight contributions in scientific conferences proceedings during the period of the Ph.D. course.

- Carnovalini, F., & Rodà, A. (2020). Computational Creativity and Music Generation Systems: An Introduction to the State of the Art. *Frontiers in Artificial Intelligence*, 3, 14. `https://doi.org/10.3389/frai.2020.00014`
- Simonetta, F., Carnovalini, F., Orio, N., & Rodà, A. (2018). Symbolic Music Similarity through a Graph-Based Representation. *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion - AM'18*, 1–7. `https://doi.org/10.1145/3243274.3243301`
- Carnovalini, F., & Rodà, A. (2019). A Multilayered Approach to Automatic Music Generation and Expressive Performance. *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*, 41–48. `https://doi.org/10.1109/MMRP.2019.00016`

- Carnovalini, F. (2019). Open Challenges in Musical Metacreation. *Proceedings of the 5th EAI International Conference on Smart Objects and Technologies for Social Good*, 124–125. https://doi.org/10.1145/3342428.3342678

- Carnovalini, F., Rodà, A., & Caneva, P. (2019). A Musical Serious Game for Social Interaction through Augmented Rhythmic Improvisation. *Proceedings of the 5th EAI International Conference on Smart Objects and Technologies for Social Good*, 130–135. https://doi.org/10.1145/3342428.3342683

- Carnovalini, F., & Rodà, A. (2019). A Real-Time Tempo and Meter Tracking System for Rhythmic Improvisation. *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound*, 24–31. https://doi.org/10.1145/3356590.3356596

- Carnovalini, F., Rodà, A., Harley, N., Homer, S. T., & Wiggins, G. A. (2021). A New Corpus for Computational Music Research and A Novel Method for Musical Structure Analysis. *Audio Mostly 2021 (AM '21)*, 4. https://doi.org/10.1145/3478384.3478402

- Carnovalini, F., Harley, N., Homer, S. T., Rodà, A., & Wiggins, G. A. (2021). Meta-Evaluating Quantitative Internal Evaluation: A Practical Approach for Developers. *Proceedings of the 12th International Conference on Computational Creativity*, 5. https://computationalcreativity.net/iccc21/wp-content/uploads/2021/09/ICCC_2021_paper_98.pdf

- Carnovalini, F., Harley, N., Homer, S., Roda, A., & Wiggins, G. A. (2021). Studying Structural Regularities through Abstraction Trees. *Proceedings of the 15th International Symposium on Computer Music Multidisciplinary Research*, 10. https://cmmr2021.github.io/proceedings/pdffiles/cmmr2021_19.pdf

Moreover, an additional journal paper was submitted and after being accepted with minor revision is awaiting evaluation from the editor.

- Carnovalini, F., Rodà, A., & Caneva, P. (2022?) A Rhythm-Aware Serious Game for Social Interaction. Submitted to *Multimedia Tools and Applications*.

5.2.2 | Deliverables

- The code for the Serious Game is available at `https://gitlab.dei.unipd.it/facoch/sympaddy/`
- The Leone dataset, used for the structural experiments, can be found here: `https://doi.org/10.5281/zenodo.5145348`
- The code for the Music Generation System, as well as the analytical tool it leverages, is available at `https://gitlab.dei.unipd.it/facoch/leone/`

5.3 | Closing Remarks

In this thesis I explored two main directions for the advancement of Music Generation and of Computational Creativity. The result is the development of new tools for the creation and co-creation of music with the help of software, rather than a theoretical advancements on what is necessary to achieve creativity in computational system. In particular, looking back at the questions in Section 1.3, most of those still have no clear answer. It would be especially interesting to verify what impact the tools presented in this thesis have to the perception of creativity in the musical output of a system, as verifying this during the PhD period was not possible due to the hardships imposed by Covid-19.

Despite being a practical work for most part, some lessons emerged. The importance of the human factor in designing intelligent and creative system is the first one. Many researches suggest that self-reflection is crucial in any creative framework. I suggest that the study of human cognition is the foundation for building such reflection systems. Moreover, the focus on human cognition and creativity can lead to applications that are useful to us. One direction I intend to explore going forward with my research career is the application of creative approaches to other AI applications. One example would be applying creativity

to recommendation systems, to enable such systems to go beyond the metrics that are generally used that only rely on the fact that people will like popular things. A creative recommendation could be able to suggest things that are not very popular (novelty), but are still of interest to the user (value). More generally, if it is true that creativity is a part of human intelligence, it is time to see whether our studies in creativity can lead to better performances in our intelligent systems, or at least make such artificial intelligences feel more humane.

References

Samer Abdallah, Nicolas Gold, and Alan Marsden. Analysing symbolic music with probabilistic grammars. In David Meredith, editor, *Computational Music Analysis*, pages 157–189. Springer International Publishing, Cham, 2016. ISBN 978-3-319-25931-4. doi: 10.1007/978-3-319-25931-4_7. URL https://doi.org/10.1007/978-3-319-25931-4_7.

Kat Agres and Dorien Herremans. Music and motion-detection: A game prototype for rehabilitation and strengthening in the elderly. In *2017 International Conference on Orange Technologies (ICOT)*, pages 95–98, Singapore, December 2017. IEEE. ISBN 978-1-5386-3276-5. doi: 10.1109/ICOT.2017.8336097. URL https://ieeexplore.ieee.org/document/8336097/.

Kat Agres, Jamie Forth, and Geraint A. Wiggins. Evaluation of musical creativity and musical metacreation systems. *Computers in Entertainment (CIE)*, 14(3):33, 2016. doi: 10.1145/2967506.

Kat R. Agres, Rebecca S. Schaefer, Anja Volk, Susan van Hooren, Andre Holzapfel, Simone Dalla Bella, Meinard Müller, Martina de Witte, Dorien Herremans, Rafael Ramirez Melendez, Mark Neerincx, Sebastian Ruiz, David Meredith, Theo Dimitriadis, and Wendy L. Magee. Music, Computing, and Health: A Roadmap for the Current and Future Roles of Music Technology for Health Care and Well-Being. *Music & Science*, 4:205920432199770, January 2021. ISSN 2059-2043, 2059-2043. doi: 10.1177/2059204321997709.

Rita Aiello. Music and language: Parallels and contrasts. In Rita Aiello and John A. Sloboda, editors, *Musical Perceptions*, pages 40–63. Oxford University Press, Oxford, UK, 1994.

Kenneth Aigen. In Defense of Beauty: A Role for the Aesthetic in Music Therapy Theory. *Nordic Journal of Music Therapy*, 16(2):112–128, January 2007. ISSN 0809-8131. doi: 10.1080/08098130709478181.

Rory Allen and Pamela Heaton. Autism, Music, and the Therapeutic Potential of Music in Alexithymia. *Music Perception*, 27(4):251–261, April 2010. ISSN 0730-7829, 1533-8312. doi: 10.1525/mp.2010.27.4.251.

Miguel Alonso, Bertrand David, and Gaël Richard. Tempo and beat estimation of musical signals. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR), Barcelona, Spain*, 2004.

Teresa M. Amabile. A Consensual Technique for Creativity Assessment. In Teresa M. Amabile, editor, *The Social Psychology of Creativity*, Springer Series in Social Psychology, pages 37–63. Springer New York, New York, NY, 1983a. ISBN 978-1-4612-5533-8. doi: 10.1007/978-1-4612-5533-8_3. URL https://doi.org/10.1007/978-1-4612-5533-8_3.

Teresa M. Amabile. The social psychology of creativity: A componential conceptualization. *Journal of Personality and Social Psychology*, 45(2):357–376, 1983b. ISSN 1939-1315(Electronic),0022-3514(Print). doi: 10.1037/0022-3514.45.2.357.

Teresa M. Amabile, Regina Conti, Heather Coon, Jeffrey Lazenby, and Michael Herron. Assessing the work environment for creativity. *Academy of management journal*, 39(5):1154–1184, 1996.

Torsten Anders and Eduardo R. Miranda. Constraint programming systems for modeling music theories and composition. *ACM Computing Surveys*, 43(4):1–38, October 2011. ISSN 03600300. doi: 10.1145/1978802.1978809.

Christopher Anderson, Arne Eigenfeldt, and Philippe Pasquier. The Generative Electronic Dance Music Algorithmic System (GEDMAS). In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment (AIIDE'13) Conference*, page 4, Boston, MA, 2013. AAAI Press.

Shunya Ariga, Satoru Fukayama, and Masataka Goto. Song2Guitar: A Difficulty-Aware Arrangement System for Generating Guitar Solo Covers from Polyphonic Audio of Popular Music. In *ISMIR*, pages 568–574, 2017.

Christopher Ariza. The interrogator as critic: The turing test and the evaluation of generative music systems. *Computer Music Journal*, 33(2):48–70, 2009.

John Baer and Sharon S. McKool. Assessing creativity using the consensual assessment technique. In *Handbook of research on assessment technologies, methods, and applications in higher education*, pages 65–77. IGI Global, 2009.

Eloi Batlle and Pedro Cano. Automatic segmentation for music classification using competitive hidden markov models. In *ISMIR*, 2000.

Chip Bell. Algorithmic Music Composition Using Dynamic Markov Chains and Genetic Algorithms. *J. Comput. Sci. Coll.*, 27(2):99–107, December 2011. ISSN 1937-4771.

Manjinder Singh Benning, Ajay Kapur, Bernie C. Till, and George Tzanetakis. Multimodal sensor analysis of sitar performance: Where is the beat? In *2007 IEEE 9th Workshop on Multimedia Signal Processing*, pages 74–77, 2007. doi: 10.1109/MMSP.2007.4412821.

Samuel Benveniste, Pierre Jouvelot, Edith Lecourt, and Renaud Michel. Designing wiimprovisation for mediation in group music therapy with children suffering from behavioral disorders. In *Proceedings of the 8th International Conference on Interaction Design and Children*, IDC '09, page 18–26, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605583952. doi: 10.1145/1551788.1551793. URL https://doi.org/10.1145/1551788.1551793.

Rick Bidlack. Chaotic systems as simple (but complex) compositional algorithms. *Computer Music Journal*, 16(3):33–47, 1992.

John Biles, Peter Anderson, and Laura Loggi. Neural network fitness functions for a musical IGA. In *Proceedings of the Soft Computing Conference*, page 11, 1996.

John A. Biles. GenJam: A genetic algorithm for generating jazz solos. In *ICMC*, volume 94, pages 131–137, 1994.

John A. Biles. Autonomous GenJam: eliminating the fitness bottleneck by eliminating fitness. In *Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop Program, San Francisco*, page 7, 2001.

John A Biles. Performing with Technology: Lessons Learned from the GenJam Project. In *Musical Metacreation: Papers from the 2013 AIIDE Workshop*, page 6, 2013a.

John A Biles. Straight-Ahead Jazz with GenJam: A Quick Demonstration. In *Musical Metacreation: Papers from the 2013 AIIDE Workshop*, page 4. Association for the Advancement of Artificial Intelli- gence, 2013b.

Sebastian Böck and Matthew EP Davies. Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation. In *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR), Montreal, QC, Canada*, pages 12–16, 2020.

Margaret A. Boden. Chapter 9 - Creativity. In Margaret A. Boden, editor, *Artificial Intelligence*, Handbook of Perception and Cognition, pages 267–291. Academic Press, San Diego, January 1996. ISBN 978-0-12-161964-0. doi: 10.1016/B978-012161964-0/50011-X. URL http://www.sciencedirect.com/science/article/pii/B978012161964050011X.

Margaret A. Boden. Creativity and artificial intelligence. *Artificial Intelligence*, 103(1-2):347–356, 1998.

Margaret A. Boden. *The creative mind: Myths and mechanisms*. Routledge, London, United Kingdom, 2004.

Margaret A. Boden. Computer models of creativity. *AI Magazine*, 30(3):23, 2009.

Paul M Bodily and Dan Ventura. Musical Metacreation: Past, Present, and Future. In *Mume 2018*, page 5, Salamanca, Spain, 2018. University of Salamanca.

Selmer Bringsjord, Paul Bello, and David Ferrucci. Creativity, the Turing test, and the (better) Lovelace test. In *The Turing Test*, pages 215–239. Springer, 2003.

Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. *Deep Learning Techniques for Music Generation*. Computational Synthesis and Creative Systems. Springer International Publishing, New York, NY, 2020. ISBN 978-3-319-70162-2. doi: 10.1007/978-3-319-70163-9. URL `https://www.springer.com/gp/book/9783319701622`.

Daniel Brown. Mezzo: An Adaptive, Real-Time Composition Program for Game Soundtracks. In *Musical Metacreation: Papers from the 2012 AIIDE Workshop*, page 5. AAAI, 2012a.

Daniel Lankford Brown. *Expressing narrative function in adaptive, computer-composed music*. PhD Thesis, UC Santa Cruz, 2012b.

Gino Brunner, Andres Konrad, Yuyi Wang, and Roger Wattenhofer. MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, pages 747–754, 2018. URL `http://ismir2018.ircam.fr/doc/pdfs/204_Paper.pdf`.

Kenneth E. Bruscia. *Improvisational models of music therapy*. Thomas, Springfield, IL, 1987. ISBN 978-0-398-06040-4 978-0-398-05272-0. OCLC: 246139778.

Pierre Brémaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*, volume 31. Springer Science & Business Media, 2013.

Bruce G Buchanan and Edward A Feigenbaum. Dendral and meta-dendral: Their applications dimension. *Artificial intelligence*, 11(1-2):5–24, 1978.

Valentin Bégel, Antoine Seilles, and Simone Dalla Bella. Rhythm workers: A music-based serious game for training rhythm skills. *Music & Science*, 1:2059204318794369, 2018. doi: 10.1177/2059204318794369.

Emilios Cambouropoulos. *Towards a general computational theory of musical structure*. PhD thesis, Ph.D. thesis, University of Edinburgh, 1998.

Emilios Cambouropoulos. The Local Boundary Detection Model (LBDM) and its Application in the Study of Expressive Timing. In *ICMC*, page 8, 2001.

S. Canazza, A. Rodà, G.D. Poli, and A. Vidolin. *Expressiveness in music performance:Analysis, models, mapping, encoding*. IGI Global, 2012. doi: 10.4018/978-1-4666-2497-9.ch008.

S. Canazza, G. De Poli, and A. Rodà. Caro 2.0: An interactive system for expressive music rendering. *Advances in Human-Computer Interaction*, 2015, 2015. doi: 10.1155/2015/850474.

Filippo Carnovalini. Open Challenges in Musical Metacreation. In *Proceedings of the 5th EAI International Conference on Smart Objects and Technologies for Social Good*, pages 124–125, Valencia Spain, September 2019. ACM. ISBN 978-1-4503-6261-0. doi: 10.1145/3342428.3342678. URL http://dl.acm.org/doi/10.1145/3342428.3342678.

Filippo Carnovalini and Antonio Rodà. A Multilayered Approach to Automatic Music Generation and Expressive Performance. In *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*, pages 41–48, Milano, Italy, January 2019a. IEEE. ISBN 978-1-72811-649-5. doi: 10.1109/MMRP.2019.00016. URL https://ieeexplore.ieee.org/document/8665367/.

Filippo Carnovalini and Antonio Rodà. A Real-Time Tempo and Meter Tracking System for Rhythmic Improvisation. In *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound*, pages 24–31, Nottingham United Kingdom, September 2019b. ACM. ISBN 978-1-4503-7297-8. doi: 10.1145/3356590.3356596. URL https://dl.acm.org/doi/10.1145/3356590.3356596.

Filippo Carnovalini and Antonio Rodà. Computational Creativity and Music Generation Systems: An Introduction to the State of the Art. *Frontiers in Artificial Intelligence*, 3:14, April 2020. ISSN 2624-8212. doi: 10.3389/frai.2020.00014.

Filippo Carnovalini, Antonio Rodà, and Paolo Caneva. A Musical Serious Game for Social Interaction through Augmented Rhythmic Improvisation. In *Proceedings of the 5th EAI International Conference on Smart Objects and Technologies for Social Good*, pages 130–135, Valencia Spain, September 2019. ACM. ISBN 978-1-4503-6261-0. doi: 10.1145/3342428.3342683. URL http://dl.acm.org/doi/10.1145/3342428.3342683.

Filippo Carnovalini, Nicholas Harley, Steve Homer, Antonio Roda, and Geraint A Wiggins. Studying Structural Regularities through Abstraction Trees. In *Proceedings of the 15th International Symposium on Computer Music Multidisciplinary Research*, page 10, Tokyo/Virtual, 2021a.

Filippo Carnovalini, Antonio Rodà, Nicholas Harley, Steven T Homer, and Geraint A Wiggins. A New Corpus for Computational Music Research andA Novel Method for Musical Structure Analysis. In *Audio*

Mostly 2021 (AM '21), page 4, virtual/Trento Italy, 2021b. ACM. ISBN 978-1-4503-8569-5. doi: 10.1145/3478384.3478402. URL https://doi.org/10.1145/3478384.3478402.

Ali Taylan Cemgil, Bert Kappen, Peter Desain, and Henkjan Honing. On tempo tracking: Tempogram representation and kalman filtering. *Journal of New Music Research*, 29(4):259–273, 2000. doi: 10.1080/09298210008565462.

Mauro Ceroni and Giovanni Maria Prosperi. Free will, subjectivity and the physics of the nervous system. *Open Journal of Philosophy*, pages 317–341, 2018. ISSN 2163-9442. doi: 10.4236/ojpp.2018.83023.

David J. Chalmers. Facing up to the problem of consciousness. *Journal of consciousness studies*, 2(3): 200–219, 1995.

Lee Cheatley, Wendy Moncur, and Alison Pease. Opportunities for computational creativity in a therapeutic context. In Kazjon Grace, Michael Cook, Dan Ventura, and Mary Lou Maher, editors, *Proceedings of the 10th International Conference on Computational Creativity*, Proceedings of the 10th International Conference on Computational Creativity, ICCC 2019, pages 341–345. Association for Computational Creativity (ACC), USA, 2019. ISBN 9789895416011. URL http://computationalcreativity.net/iccc2019/assets/iccc%5fproceedings%5f2019.pdf.

Lee Cheatley, Margareta Ackerman, Alison Pease, and Wendy Moncur. Co-creative songwriting for bereavement support. In *Eleventh International Conference on Computational Creativity: ICCC'20*, pages 33–41. Association for Computational Creativity, 2020.

M. Chemillier. Toward a formal study of jazz chord sequences generated by Steedman's grammar. *Soft Computing*, 8(9):617–622, September 2004. ISSN 1433-7479. doi: 10.1007/s00500-004-0386-3.

Ke Chen, Weilin Zhang, Shlomo Dubnov, Gus Xia, and Wei Li. The effect of explicit structure encoding of deep neural networks for symbolic music generation. In *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*, pages 77–84. IEEE, 2019.

Elaine Chew. The Spiral Array. In Elaine Chew, editor, *Mathematical and Computational Modeling of Tonality: Theory and Applications*, International Series in Operations Research & Management Science, pages 41–60. Springer US, Boston, MA, 2014. ISBN 978-1-4614-9475-1. doi: 10.1007/978-1-4614-9475-1_3. URL https://doi.org/10.1007/978-1-4614-9475-1_3.

Noam Chomsky. *Syntactic Structures*. Janua Linguarum. Mouton & Co, The Hague, 1957.

Tom Collins and Christian Coulon. FreshJam: Suggesting Continuations of Melodic Fragments in a Specific Style. In *Proceedings of the 2012 AIIDE Workshop*, page 3, 2012.

Simon Colton. Creativity Versus the Perception of Creativity in Computational Systems. In *AAAI spring symposium: creative intelligent systems*, volume 8, page 7, 2008.

Simon Colton and Geraint A. Wiggins. Computational creativity: The final frontier? In *ECAI*, volume 2012, pages 21–16, Montpellier, France, 2012. University of Montpellier.

Simon Colton, Alison Pease, and Graeme Ritchie. The effect of input knowledge on creativity. In *Proceedings of the ICCBR'01 Workshop on Creative Systems*, page 7, 2001.

Simon Colton, John William Charnley, and Alison Pease. Computational Creativity Theory: The FACE and IDEA Descriptive Models. In *ICCC*, pages 90–95, 2011.

Simon Colton, Alison Pease, Joseph Corneli, Michael Cook, and Teresa Llano. Assessing Progress in Building Autonomously Creative Systems. In *ICCC*, pages 137–145, 2014.

Simon Colton, Alison Pease, Christian Guckelsberger, Jon McCormack, Maria Teresa Llano, et al. On the machine condition and its creative expression. In *ICCC*, pages 342–349, 2020.

Darrell Conklin and Christina Anagnostopoulou. Representation and discovery of multiple viewpoint patterns. In *ICMC*, pages 479–485, 2001.

John Conway. The game of life. *Scientific American*, 223(4):4, 1970.

D. Cope. Recombinant music: using the computer to explore musical style. *Computer*, 24(7):22–28, July 1991. ISSN 0018-9162. doi: 10.1109/2.84830.

David Cope. Computer Modeling of Musical Intelligence in EMI. *Computer Music Journal*, 16(2):69, 1992. ISSN 01489267. doi: 10.2307/3680717.

Nailson dos Santos Cunha, Anand Subramanian, and Dorien Herremans. Generating guitar solos by integer programming. *Journal of the Operational Research Society*, 69(6):971–985, 2018.

Shuqi Dai, Zheng Zhang, and Gus G Xia. Music style transfer: A position paper. *arXiv preprint arXiv:1803.06841*, 2018.

Roger B. Dannenberg. An on-line algorithm for real-time accompaniment. In *ICMC*, volume 84, pages 193–198, Ann Arbor, MI, 1984. Michigan Publishing.

Alfonso Ortega de la Puente, Rafael Sánchez Alfonso, and Manuel Alfonseca Moreno. Automatic composition of music by means of grammatical evolution. In *ACM SIGAPL APL Quote Quad*, volume 32, pages 148–155. ACM, 2002.

Diana Deutsch, William F Thompson, Henkjan Honing, and Stephen McAdams. *Psychology of Music*. Elsevier, 2013.

Sander Dieleman, Aaron van den Oord, and Karen Simonyan. The challenge of realistic music generation: modelling raw audio at scale. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7989–7999. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/8023-the-challenge-of-realistic-music-generation-modelling-raw-audio-at-scale.pdf`.

Simon Dixon. Automatic Extraction of Tempo and Beat From Expressive Performances. *Journal of New Music Research*, 30(1):39–58, March 2001. ISSN 0929-8215. doi: 10.1076/jnmr.30.1.39.7119.

Simon Durand, Juan Pablo Bello, Bertrand David, and Gaël Richard. Robust downbeat tracking using an ensemble of convolutional networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(1):76–89, 2017. doi: 10.1109/TASLP.2016.2623565.

Kemal Ebcio\v{g}lu. An Expert System for Harmonizing Four-Part Chorales. *Computer Music Journal*, 12 (3):43–51, 1988. ISSN 0148-9267. doi: 10.2307/3680335.

Kemal Ebcio\v{g}lu. An expert system for harmonizing chorales in the style of J.S. Bach. *The Journal of Logic Programming*, 8(1):145–185, January 1990. ISSN 0743-1066. doi: 10.1016/0743-1066(90)90055-A.

Douglas Eck and Juergen Schmidhuber. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In *Proceedings of the 12th IEEE workshop on neural networks for signal processing*, pages 747–756. IEEE, 2002a.

Douglas Eck and Jürgen Schmidhuber. Learning the long-term structure of the blues. In *International Conference on Artificial Neural Networks*, pages 284–289. Springer, 2002b.

A. Eigenfeldt, A. Burnett, and P. Pasquier. Evaluating musical metacreation in a live performance context. In *Proceedings of the 3rd International Conference on Computational Creativity, ICCC 2012*, pages 140–144, 2012.

Arne Eigenfeldt and Philippe Pasquier. A realtime generative music system using autonomous melody, harmony, and rhythm agents. In *XIII Internationale Conference on Generative Arts, Milan, Italy*, pages 67–76, 2009.

Arne Eigenfeldt and Philippe Pasquier. Realtime generation of harmonic progressions using controlled markov selection. In *Proceedings of ICCC-X-Computational Creativity Conference*, pages 16–25, 2010.

Henrik Ekeus, Samer A Abdallah, Mark D Plumbley, and Peter W McOwan. The Melody Triangle: Exploring Pattern and Predictability in Music. In *Musical Metacreation: Papers from the 2012 AIIDE Workshop*, page 8, 2012.

Antti J. Eronen and Anssi P. Klapuri. Music tempo estimation with <formula formulatype="inline"> <tex notation="tex">k</tex></formula>-nn regression. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(1):50–57, 2010. doi: 10.1109/TASL.2009.2023165.

Gilles Fauconnier and Mark Turner. *The way we think: Conceptual blending and the mind's hidden complexities*. Basic Books, 2008.

Jose D. Fernández and Francisco Vico. AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48:513–582, 2013.

Jamie Forth and Geraint A Wiggins. An approach for identifying salient repetition in multidimensional representations of polyphonic music. In *London Algorithmics 2008: Theory and Practice*. College Publications, 2009.

Eric Foxley. Nottingham Database, 2011. URL `https://ifdo.ca/~seymour/nottingham/nottingham.html`.

Samuel P. Fraiberger, Roberta Sinatra, Magnus Resch, Christoph Riedl, and Albert-László Barabási. Quantifying reputation and success in art. *Science*, page eaau7224, November 2018. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aau7224.

Bradley W. Frankland and Annabel J Cohen. Parsing of melody: Quantification and testing of the local grouping rules of lerdahl and jackendoff's a generative theory of tonal music. *Music Perception*, 21: 499–543, 2004.

Klaus Frieler. Beat and meter extraction using gaussified onsets. In *ISMIR*, page 6, Barcelona, Spain, 2004. Universitat Pompeu Fabra.

Takako Fujioka, Deirdre R. Dawson, Rebecca Wright, Kie Honjo, Joyce L. Chen, J. Jean Chen, Sandra E. Black, Donald T. Stuss, and Bernhard Ross. The effects of music-supported therapy on motor, cognitive, and psychosocial functions in chronic stroke. *Annals of the New York Academy of Sciences*, 1423(1): 264–274, 2018. ISSN 1749-6632. doi: 10.1111/nyas.13706.

Philip Galanter. Computational Aesthetic Evaluation: Past and Future. In Jon McCormack and Mark d'Inverno, editors, *Computers and Creativity*, pages 255–293. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-31727-9. doi: 10.1007/978-3-642-31727-9_10. URL `https://doi.org/10.1007/978-3-642-31727-9_10`.

Jacob W. Getzels and Philip W. Jackson. *Creativity and intelligence: Explorations with gifted students*. Creativity and intelligence: Explorations with gifted students. Wiley, Oxford, England, 1962.

Édouard Gilbert and Darrell Conklin. A probabilistic context-free grammar for melodic reduction. In *Proceedings of the International Workshop on Artificial Intelligence and Music, 20th International Joint Conference on Artificial Intelligence*, pages 83–94, 2007.

Samuel Gillespie and Oliver Bown. Solving adaptive game music transitions from a composer centred perspective. In *Proceedings of the 5th International Workshop on Musical Metacreation*, page 8. Association for Computational Creativity, 2017.

Jon Gillick, Adam Roberts, Jesse Engel, Douglas Eck, and David Bamman. Learning to groove with inverse sequence transformations. In *International Conference on Machine Learning (ICML)*, page 11, 2019.

Stuart E. Golann. Psychological study of creativity. *Psychological Bulletin*, 60(6):548–565, 1963. ISSN 1939-1455(Electronic),0033-2909(Print). doi: 10.1037/h0041573.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`.

Fabien Gouyon and Perfecto Herrera. Determination of the meter of musical audio signals: Seeking recurrences in beat segment descriptors. In *Audio Engineering Society Convention 114*, page 8, Amsterdam, Netherlands, 2003. Audio Engineering Society.

Ryan Groves. Towards the Generation of Melodic Structure. In *The Fourth International Workshop on Musical Metacreation, MUME 2016.*, page 8, 2016.

Yixing Guan, Jinyu Zhao, Yiqin Qiu, Zheng Zhang, and Gus Xia. Melodic phrase segmentation by deep neural networks. In *ISMIR*, 2018.

Christian Guckelsberger, Christoph Salge, and Simon Colton. Addressing the "Why?" in Computational Creativity: A Non-Anthropocentric, Minimal Model of Intentional Creative Agency. In *Proceedings of the 8th International Conference on Computational Creativity*, page 8. Goldsmiths, University of London, 2017. URL `http://ccg.doc.gold.ac.uk/wp-content/uploads/2017/05/iccc2017_guckelsberger.pdf`.

J. P. Guilford. Creative abilities in the arts. *Psychological Review*, 64(2):110–118, 1957. ISSN 1939-1471(Electronic),0033-295X(Print). doi: 10.1037/h0048280.

J.P. Guilford. *The nature of human intelligence*. The nature of human intelligence. McGraw-Hill, New York, NY, US, 1967.

Gaëtan Hadjeres, François Pachet, and Frank Nielsen. Deepbach: a steerable model for bach chorales generation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1362–1371. JMLR. org, 2017.

Susan Hallam. The power of music: Its impact on the intellectual, social and personal development of children and young people. *International Journal of Music Education*, 28(3):269–289, August 2010. ISSN 0255-7614. doi: 10.1177/0255761410370658.

Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Implementing "A generative theory of tonal music". *Journal of New Music Research*, 35(4):249–277, 2006.

Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Fatta: Full Automatic Time-Span Tree Analyzer. In *ICMC*, pages 153–156. Citeseer, 2007.

Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Melody Morphing Method Based on GTTM. In *ICMC*, pages 155–158. Citeseer, 2008.

Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Implementing methods for analysing music based on lerdahl and jackendoff's generative theory of tonal music. In *Computational Music Analysis*, pages 221–249. Springer, New York, NY, 2016.

Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. deepgttm-iii: Multi-task learning with grouping and metrical structures. In *International Symposium on Computer Music Multidisciplinary Research*, pages 238–251, Matosinhos, Porto, 2017. Springer.

Stuart Hameroff and Roger Penrose. Consciousness in the universe: A review of the 'Orch OR' theory. *Physics of Life Reviews*, 11(1):39–78, March 2014. ISSN 1571-0645. doi: 10.1016/j.plrev.2013.08.002.

Eliot Handelman, Andie Sigler, and David Donna. Automatic Orchestration for Automatic Composition. In *Musical Metacreation: Papers from the 2012 AIIDE Workshop*, page 6. AAAI, 2012.

Andrew Hawryshkewich, Philippe Pasquier, and Arne Eigenfeldt. Beatback: A Real-time Interactive Percussion System for Rhythmic Practise and Exploration. In *NIME*, pages 100–105, Sydney, Australia, 2010. University of Technology Sydney.

D. Herremans, S. Weisser, K. Sörensen, and D. Conklin. Generating structured music for bagana using quality metrics based on markov models. *Expert Systems with Applications*, 42(21):7424–7435, 2015. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2015.05.043.

Dorien Herremans and Elaine Chew. MorpheuS: automatic music generation with recurrent pattern constraints and tension profiles. In *Region 10 Conference (TENCON), 2016 IEEE*, pages 282–285. IEEE, 2016a.

Dorien Herremans and Elaine Chew. Tension ribbons: Quantifying and visualising tonal tension. In *Proceedings of the Second International Conference on Technologies for Music Notation and Representation (TENOR)*, page 10, Cambridge, UK, 2016b.

Dorien Herremans and Elaine Chew. MorpheuS: generating structured music with constrained patterns and tension. *IEEE Transactions on Affective Computing*, 10(4):16, 2017. ISSN 1949-3045. doi: 10.1109/TAFFC.2017.2737984.

Dorien Herremans, Ching-Hua Chuan, and Elaine Chew. A functional taxonomy of music generation systems. *ACM Computing Surveys (CSUR)*, 50(5):69, 2017.

Lejaren A. Hiller Jr and Leonard M. Isaacson. Musical composition with a high-speed digital computer. *Journal of the Audio Engineering Society*, 6(3):154–160, 1958.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

James Hodson. The creative machine. In *ICCC*, pages 143–150, 2017.

Andrew Horner and David E. Goldberg. Genetic Algorithms and Computer-Assisted Music Composition. In *ICMC*, volume 91, pages 479–482, 1991.

Michael J Hove and Jane L Risen. It's all in the timing: Interpersonal synchrony increases affiliation. *Social cognition*, 27(6):949–960, 2009.

Jia-Lien Hsu, Arbee L. P. Chen, and C.-C. Liu. Efficient repeating pattern finding in music databases. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, CIKM '98, page 281–288, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 1581130619. doi: 10.1145/288627.288668. URL https://doi.org/10.1145/288627.288668.

Kenneth Jinghwa Hsü and Andrew Hsü. Self-similarity of the" 1/f noise" called music. *Proceedings of the National Academy of Sciences*, 88(8):3507–3509, 1991.

Patrick G. Hunter and E. Glenn Schellenberg. Music and emotion. In *Music perception*, pages 129–164. Springer, 2010.

David Huron. *Sweet Anticipation: Music and the Psychology of Expectation*. MIT Press, January 2008. ISBN 978-0-262-30330-9.

Tatsuhiko Itohara, Takuma Otsuka, Takeshi Mizumoto, Angelica Lim, Tetsuya Ogata, and Hiroshi G Okuno. A multimodal tempo and beat-tracking system based on audiovisual information from live guitar performances. *EURASIP Journal on Audio, Speech, and Music Processing*, 2012(1):1–17, 2012.

Bijue Jia, Jiancheng Lv, and Dayiheng Liu. Deep learning-based automatic downbeat tracking: a brief review. *Multimedia Systems*, 25(6):617–638, December 2019. ISSN 1432-1882. doi: 10.1007/s00530-019-00607-x.

A. Jordanous. Four PPPPerspectives on computational creativity in theory and in practice. *Connection Science*, 28(2):194–216, 2016. doi: 10.1080/09540091.2016.1151860.

Anna Jordanous. A standardised procedure for evaluating creative systems: Computational creativity evaluation based on what it is to be creative. *Cognitive Computation*, 4(3):246–279, 2012.

Anna Jordanous. Stepping back to progress forwards: Setting standards for meta-evaluation of computational creativity. In *Proceedings of the Fifth International Conference on Computational Creativity*, page 8, Ljubljana, Slovenia, 2014. Jožef Stefan Institute.

Anna Jordanous. Evaluating Evaluation: Assessing Progress and Practices in Computational Creativity Research. In Tony Veale and F. Amílcar Cardoso, editors, *Computational Creativity: The Philosophy and Engineering of Autonomously Creative Systems*, Computational Synthesis and Creative Systems, pages 211–236. Springer International Publishing, Cham, 2019. ISBN 978-3-319-43610-4. doi: 10.1007/978-3-319-43610-4_10. URL https://doi.org/10.1007/978-3-319-43610-4_10.

Anna Jordanous and Bill Keller. What makes musical improvisation creative? *Journal of Interdisciplinary Music Studies*, 6:151–175, October 2012. ISSN 1307-0401. doi: 10.4407/jims.2014.02.003.

Anna Jordanous and Bill Keller. Modelling Creativity: Identifying Key Components through a Corpus-Based Approach. *PLOS ONE*, 11(10):e0162959, 2016. ISSN 1932-6203. doi: 10.1371/journal.pone.0162959.

Anna Katerina Jordanous. *Evaluating computational creativity: a standardised procedure for evaluating creative systems and its application*. PhD Thesis, University of Sussex, 2013.

Patrik N. Juslin. *Handbook of Music and Emotion: Theory, Research, Applications*. Oxford University Press, January 2010. ISBN 978-0-19-169643-5. URL http://www.oxfordscholarship.com/view/10.1093/acprof:oso/9780199230143.001.0001/acprof-9780199230143.

Maximos Kaliakatsos-Papakostas, Aggelos Gkiokas, and Vassilis Katsouros. Interactive Control of Explicit Musical Features in Generative LSTM-based Systems. In *Proceedings of the Audio Mostly 2018 on Sound*

in Immersion and Emotion, AM'18, pages 29:1–29:7, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-6609-0. doi: 10.1145/3243274.3243296. URL http://doi.acm.org/10.1145/3243274.3243296. event-place: Wrexham, United Kingdom.

Michael Kassler. *Proving musical theorems I: The middleground of Heinrich Schenker's theory of tonality*. Basser Department of Computer Science, School of Physics, University of Sydney, 1975.

Haruhiro Katayose, Mitsuyo Hashida, Giovanni De Poli, and Keiji Hirata. On evaluating systems for generating expressive music performance: the rencon experience. *Journal of New Music Research*, 41(4): 299–310, 2012.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, page 14, 2013.

Alexis Kirke and Eduardo Miranda. A multi-agent emotional society whose melodies represent its emergent social hierarchy and are generated by agent communications. *Journal of Artificial Societies and Social Simulation*, 18(2):16, 2015.

Alexis Kirke and Eduardo R. Miranda. Emergent Construction of melodic pitch and hierarchy through agents communicating emotion without melodic intelligence. In *ICMC*, page 8, 2011.

Alexis Kirke and Eduardo Reck Miranda. A survey of computer systems for expressive music performance. *ACM Computing Surveys (CSUR)*, 42(1):3, 2009.

Phillip B Kirlin and Paul E Utgoff. A framework for automated schenkerian analysis. In *ISMIR*, pages 363–368, 2008.

Stefan Koelsch. Music-evoked emotions: principles, brain correlates, and implications for therapy. *Annals of the New York Academy of Sciences*, 1337:193–201, 2015. doi: 10.1111/nyas.12684.

Arthur Koestler. The act of creation: Hutchinson & Co. *Ltd., London, UK*, 1964.

Dimitra Kokotsaki and Susan Hallam. Higher education music students' perceptions of the benefits of participative music making. *Music Education Research*, 9(1):93–109, March 2007. ISSN 1461-3808. doi: 10.1080/14613800601127577.

Carolyn Lamb, Daniel G. Brown, and Charles L. A. Clarke. Evaluating Computational Creativity: An Interdisciplinary Tutorial. *ACM Computing Surveys*, 51(2):1–34, February 2018. ISSN 03600300. doi: 10.1145/3167476.

Peter Langston. Six techniques for algorithmic music composition. In *Proceedings of the International Computer Music Conference*, volume 60, page 59, 1989.

Olivier Lartillot. Automated motivic analysis: An exhaustive approach based on closed and cyclic pattern mining in multidimensional parametric spaces. In David Meredith, editor, *Computational Music Analysis*, pages 273–302. Springer International Publishing, Cham, 2016. ISBN 978-3-319-25931-4. doi: 10.1007/978-3-319-25931-4_11. URL https://doi.org/10.1007/978-3-319-25931-4_11.

Jeremy Leach and John Fitch. Nature, music, and algorithmic composition. *Computer Music Journal*, 19 (2):23–33, 1995.

Douglas Bruce Lenat. *AM: an artificial intelligence approach to discovery in mathematics as heuristic search.* Stanford University, 1976.

Fred Lerdahl and Ray S. Jackendoff. *A generative theory of tonal music*. MIT press, Cambridge, MA, 1985.

George E. Lewis. Too many notes: Computers, complexity and culture in voyager. *Leonardo Music Journal*, pages 33–39, 2000.

J. P. Lewis. *Music and Connectionism, chap. Creation by refinement and the problem of algorithmic music composition*. The MIT Press, Cambridge, 1991.

Man Yat Lo. *Evolving cellular automata for music composition with trainable fitness functions*. PhD Thesis, University of Essex, 2012.

ManYat Lo and Simon M. Lucas. Evolving musical sequences with n-gram based trainable fitness functions. In *2006 IEEE International Conference on Evolutionary Computation*, pages 601–608. IEEE, 2006.

O. Lopez-Rincon, O. Starostenko, and G. A. Martín. Algoritmic music composition based on artificial intelligence: A survey. In *2018 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pages 187–193, February 2018. doi: 10.1109/CONIELECOMP.2018.8327197.

Ada Lovelace. 'Notes on L. Menabrea's 'Sketch of the Analytical Engine Invented by Charles Babbage, Esq."'. *Taylor's Scientific Memoirs*, 3(1843):1843, 1843.

Bill Manaris, Dana Hughes, and Yiorgos Vassilandonakis. Monterey mirror: combining Markov models, genetic algorithms, and power laws. In *Proceedings of 1st Workshop in Evolutionary Music, 2011 IEEE Congress on Evolutionary Computation (CEC 2011)*, pages 33–40. Citeseer, 2011.

Alan Marsden. Automatic derivation of musical structure: A tool for research on schenkerian analysis. In *ISMIR*, pages 55–58, 2007.

Alan Marsden. Schenkerian Analysis by Computer: A Proof of Concept. *Journal of New Music Research*, 39(3):269–289, September 2010. ISSN 0929-8215, 1744-5027. doi: 10.1080/09298215.2010.503898.

Alan Marsden, Keiji Hirata, and Satoshi Tojo. Towards computable procedures for deriving tree structures in music: Context dependency in GTTM and Schenkerian theory. In *Proceedings of the Sound and Music Computing Conference 2013*, pages 360–367, Stockholm, Sweden, 2013. KTH Royal Institute of Technology.

Stephanie Mason and Michael Saffle. L-Systems, melodies and musical structure. *Leonardo Music Journal*, pages 31–38, 1994.

Panayotis Mavromatis and Matthew Brown. Parsing context-free grammars for music: A computational model of schenkerian analysis. In *Proceedings of the 8th International Conference on Music Perception & Cognition*, pages 414–415, 2004.

Kenneth McAlpine, Eduardo Miranda, and Stuart Hoggar. Making music with algorithms: A case-study system. *Computer Music Journal*, 23(2):19–30, 1999.

Matt McVicar, Satoru Fukayama, and Masataka Goto. AutoLeadGuitar: Automatic generation of guitar solo phrases in the tablature space. In *2014 12th International Conference on Signal Processing (ICSP)*, pages 599–604. IEEE, 2014.

Gabriele Medeot, Srikanth Cherla, Katerina Kosta, Matt McVicar, Samer Abdallah, and Marco Selvi. StructureNet: INDUCING STRUCTURE IN GENERATED MELODIES. In *19th International Society for Music Information Retrieval Conference*, page 7, Paris, France, 2018.

David Meredith. Cosiatec and siateccompress: Pattern discovery by geometric compression. In *Music Information Retrieval Evaluation eXchange (MIREX 2013)*. International Society for Music Information Retrieval, 2013. International Society for Music Information Retrieval Conference, ISMIR 2013 ; Conference date: 04-11-2013 Through 08-11-2013.

David Meredith, Kjell Lemström, and Geraint A. Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 31(4):321–345, 2002. doi: 10.1076/jnmr.31.4.321.14162.

L.B. Meyer. *Style and Music: Theory, History, and Ideology*. Studies in the criticism and theory of music. University of Chicago Press, 1989. ISBN 9780226521527.

Leonard B. Meyer. *Emotion and meaning in music*. University of chicago Press, 1956.

Marvin L. Minsky. Why People Think Computers Can't. *AI Magazine*, 3(4):3–3, December 1982. ISSN 2371-9621. doi: 10.1609/aimag.v3i4.376.

Eduardo R. Miranda. Cellular automata music: from sound synthesis to musical forms. In *Evolutionary computer music*, pages 170–193. Springer, 2007.

Eduardo Reck Miranda. Cellular automata music: An interdisciplinary project. *Journal of New Music Research*, 22(1):3–21, 1993.

Bart Moens, Chris Muller, Leon van Noorden, Marek Franěk, Bert Celie, Jan Boone, Jan Bourgois, and Marc Leman. Encouraging Spontaneous Synchronisation with D-Jogger, an Adaptive Music Player That Aligns Movement and Music. *PLoS ONE*, 9(12):e114234, December 2014. ISSN 1932-6203. doi: 10.1371/journal.pone.0114234.

David C. Moffat and Martin G. Kelly. An investigation into people's bias against computational creativity in music composition. In *Proceedings of the third Joint Workshop on Computational Creativity (as part of ECAI 2006)*, 2006.

Olof Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

Meinard Muller, Frank Kurth, and Tido Roder. Towards an Efficient Algorithm for Automatic Score-to-Audio Synchronization. In *ISMIR*, page 8, Barcelona, Spain, 2004. Universitat Pompeu Fabra.

Eita Nakamura and Kazuyoshi Yoshii. Statistical piano reduction controlling performance difficulty. *APSIPA Transactions on Signal and Information Processing*, 7, 2018.

Jean-Jacques Nattiez. *Fondements d'une sémiologie de la Musique*. Union Générale d'Editons, 1975.

Maria Navarro, Juan Corchado, and Yves Demazeau. A musical composition application based on a multiagent system to assist novel composers. In *5th International Conference on Computational Creativity*, page 4, Ljubljana, Slovenia, 2014.

Maria Navarro, Juan Manuel Corchado, and Yves Demazeau. MUSIC-MAS: Modeling a harmonic composition system with virtual organizations to assist novice composers. *Expert Systems with Applications*, 57: 345–355, 2016.

Gary Lee Nelson. Real time transformation of musical material with fractal algorithms. *Computers & Mathematics with Applications*, 32(1):109–116, 1996.

Gerhard Nierhaus. *Algorithmic Composition: Paradigms of Automated Music Generation*. Springer Science & Business Media, August 2009. ISBN 978-3-211-75540-2. Google-Books-ID: jaowAtnXsDQC.

Misato Ohkita, Yoshiaki Bando, Yukara Ikemiya, Katsutoshi Itoyama, and Kazuyoshi Yoshii. Audio-visual beat tracking based on a state-space model for a music robot dancing with humans. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5555–5560. IEEE, 2015.

António Pedro Oliveira and Amílcar Cardoso. Towards affective-psychophysiological foundations for music production. In *International Conference on Affective Computing and Intelligent Interaction*, pages 511–522. Springer, 2007.

Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. This time with feeling: learning expressive musical performance. *Neural Computing and Applications*, November 2018. ISSN 1433-3058. doi: 10.1007/s00521-018-3758-9.

Nicola Orio and Antonio Roda. A measure of melodic similarity based on a graph representation of the music structure. In *ISMIR*, pages 543–548, Kobe, Japan, 2009. ISMIR.

François Pachet. Computer Analysis of Jazz Chord Sequence: Is Solar a Blues? In Eduardo Reck Miranda, editor, *Readings in Music and Artificial Intelligence*, pages 85–114. Harwood Academic Publishers, 2000.

François Pachet. Interacting with a musical learning system: The continuator. In *Music and Artificial Intelligence*, pages 119–132. Springer, 2002.

François Pachet. Musical Virtuosity and Creativity. In Jon McCormack and Mark d'Inverno, editors, *Computers and Creativity*, pages 115–146. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-31727-9. doi: 10.1007/978-3-642-31727-9_5. URL `https://doi.org/10.1007/978-3-642-31727-9_5`.

George Papadopoulos and Geraint Wiggins. AI methods for algorithmic composition: A survey, a critical view and future prospects. In *AISB Symposium on Musical Creativity*, volume 124, pages 110–117. Edinburgh, UK, 1999.

Richard Parncutt. A Perceptual Model of Pulse Salience and Metrical Accent in Musical Rhythms. *Music Perception: An Interdisciplinary Journal*, 11(4):409–464, July 1994. ISSN 0730-7829, 1533-8312. doi: 10.2307/40285633.

M. Pearce and Geraint A. Wiggins. EXPECTATION IN MELODY: THE INFLUENCE OF CONTEXT AND LEARNING. *Music Perception*, 23:377–405, 2006.

Marcus Pearce and Geraint Wiggins. Towards a framework for the evaluation of machine compositions. In *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 22–32, 2001.

Marcus Pearce, David Meredith, and Geraint Wiggins. Motivations and methodologies for automation of the compositional process. *Musicae Scientiae*, 6(2):119–147, 2002.

Marcus T Pearce. Statistical learning and probabilistic prediction in music cognition: mechanisms of stylistic enculturation. *Annals of the New York Academy of Sciences*, 1423(1):378, 2018.

Marcus T Pearce and Geraint A Wiggins. A comparison of statistical and rule-based models of melodic segmentation. In *ISMIR*, pages 89–94, 2008.

Alison Pease and Simon Colton. Computational Creativity Theory: Inspirations behind the FACE and the IDEA models. In *ICCC*, pages 72–77. Citeseer, 2011a.

Alison Pease and Simon Colton. On impact and evaluation in computational creativity: A discussion of the Turing test and an alternative proposal. In *Proceedings of the AISB symposium on AI and Philosophy*, volume 39, page 8, 2011b.

Alison Pease and Joseph Corneli. Evaluation of Creativity. In Roberto Confalonieri, Alison Pease, Marco Schorlemmer, Tarek R. Besold, Oliver Kutz, Ewen Maclean, and Maximos Kaliakatsos-Papakostas, editors, *Concept Invention: Foundations, Implementation, Social Aspects and Applications*, Computational Synthesis and Creative Systems, pages 277–294. Springer International Publishing, Cham, 2018. ISBN 978-3-319-65602-1. doi: 10.1007/978-3-319-65602-1_10. URL https://doi.org/10.1007/978-3-319-65602-1_10.

Alison Pease, Daniel Winterstein, and Simon Colton. Evaluating machine creativity. In *Workshop on creative systems, 4th international conference on case based reasoning*, pages 129–137, 2001.

Camila Pérez-Arévalo, Cristina Manresa-Yee, and Victor M. Peñeñory Beltrán. Game to develop rhythm and coordination in children with hearing impairments. In *Proceedings of the XVIII International Conference on Human Computer Interaction*, Interacción '17, page 4, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450352291. doi: 10.1145/3123818.3123853. URL https://doi.org/10.1145/3123818.3123853.

Daniel Gilbert Perret. *Roots of musicality: music therapy and personal development*. J. Kingsley Publishers, London ; Philadelphia, 2005. ISBN 978-1-84310-336-3.

Matevž Pesek, Aleš Leonardis, and Matija Marolt. Symchm—an unsupervised approach for pattern discovery in symbolic music with a compositional hierarchical model. *Applied Sciences*, 7(11), 2017. ISSN 2076-3417. doi: 10.3390/app7111135.

Somnuk Phon-Amnuaisuk and Geraint A. Wiggins. The four-part harmonisation problem: a comparison between genetic algorithms and a rule-based system. In *Proceedings of the AISB'99 Symposium on Musical Creativity*, pages 28–34. AISB London, 1999.

Somnuk Phon-Amnuaisuk, Andrew Tuson, and Geraint Wiggins. Evolving Musical Harmonisation. In Andrej Dobnikar, Nigel C. Steele, David W. Pearson, and Rudolf F. Albrecht, editors, *Artificial Neural Nets and Genetic Algorithms*, pages 229–234. Springer Vienna, 1999. ISBN 978-3-7091-6384-9.

Florian Pinel, Lav R. Varshney, and Debarun Bhattacharjya. A Culinary Computational Creativity System. In Tarek R. Besold, Marco Schorlemmer, and Alan Smaill, editors, *Computational Creativity Research: Towards Creative Machines*, Atlantis Thinking Machines, pages 327–346. Atlantis Press, Paris, 2015. ISBN 978-94-6239-085-0. doi: 10.2991/978-94-6239-085-0_16. URL https://doi.org/10.2991/978-94-6239-085-0_16.

Richard C. Pinkerton. Information theory and melody. *Scientific American*, 194(2):77–87, 1956.

Dirk-Jan Povel and Peter Essens. Perception of Temporal Patterns. *Music Perception: An Interdisciplinary Journal*, 2(4):411–440, July 1985. ISSN 0730-7829, 1533-8312. doi: 10.2307/40285311.

Przemyslaw Prusinkiewicz. Score generation with L-systems. In *ICMC*, pages 455–457, 1986.

Donya Quick. Generating Music Using Concepts from Schenkerian Analysis and Chord Spaces. Technical report, Yale University, 2011.

Eve-Marie Quintin, Anjali Bhatara, Hélène Poissant, Eric Fombonne, and Daniel J. Levitin. Emotion Perception in Music in High-Functioning Adolescents With Autism Spectrum Disorders. *Journal of Autism and Developmental Disorders*, 41(9):1240–1255, September 2011. ISSN 0162-3257, 1573-3432. doi: 10.1007/s10803-010-1146-0.

Elio Quinton. *Towards the Automatic Analysis of Metric Modulations*. PhD thesis, Queen Mary University of London, 2017.

Elio Quinton, Mark Sandler, and Simon Dixon. Estimation of the reliability of multiple rhythm features extraction from a single descriptor. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 256–260, 2016. doi: 10.1109/ICASSP.2016.7471676.

B. S. C. Ranjan, L. Siddharth, and Amaresh Chakrabarti. A systematic approach to assessing novelty, requirement satisfaction, and creativity. *AI EDAM*, 32(4):390–414, November 2018. ISSN 0890-0604, 1469-1760. doi: 10.1017/S0890060418000148.

Christopher Raphael. A Bayesian Network for Real-Time Musical Accompaniment. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1433–1439. MIT Press, Cambridge, MA, 2002. URL http://papers.nips.cc/paper/2035-a-bayesian-network-for-real-time-musical-accompaniment.pdf.

Bruno H. Repp. Sensorimotor synchronization: A review of the tapping literature. *Psychonomic Bulletin & Review*, 12(6):969–992, December 2005. ISSN 1069-9384, 1531-5320. doi: 10.3758/BF03206433.

Mel Rhodes. An Analysis of Creativity. *The Phi Delta Kappan*, 42(7):305–310, 1961. ISSN 0031-7217.

Mark O. Riedl. The lovelace 2.0 test of artificial creativity and intelligence. *arXiv preprint arXiv:1410.6142*, 2014.

Graeme Ritchie. Assessing creativity. In *Proc. of AISB'01 Symposium*, page 10. University of Edinburgh, Department of Artificial Intelligence, 2001.

Graeme Ritchie. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines*, 17(1):67–99, 2007.

Graeme Ritchie. The Evaluation of Creative Systems. In Tony Veale and F. Amílcar Cardoso, editors, *Computational Creativity: The Philosophy and Engineering of Autonomously Creative Systems*, Computational Synthesis and Creative Systems, pages 159–194. Springer International Publishing, Cham, 2019. ISBN 978-3-319-43610-4. doi: 10.1007/978-3-319-43610-4_8. URL https://doi.org/10.1007/978-3-319-43610-4_8.

Ute Ritterfeld, Michael Cody, and Peter Vorderer. *Serious games: Mechanisms and effects*. Routledge, 2009. ISBN 9780203891650. doi: https://doi.org/10.4324/9780203891650.

Andrew Robertson and Mark Plumbley. B-Keeper: a beat-tracker for live performance. In *Proceedings of the 7th international conference on New interfaces for musical expression - NIME '07*, page 234, New York, New York, 2007. ACM Press. doi: 10.1145/1279740.1279787. URL http://portal.acm.org/citation.cfm?doid=1279740.1279787.

Pierre-Yves Rolland. Discovering patterns in musical sequences. *Journal of New Music Research*, 28(4):334–350, 1999. doi: 10.1076/0929-8215(199912)28:04;1-O;FT334.

Chiara Santolin, Sofia Russo, Giulia Calignano, Jenny R. Saffran, and Eloisa Valenza. The role of prosody in infants' preference for speech: A comparison between speech and birdsong. *Infancy*, 24(5):827–833, 2019. doi: 10.1111/infa.12295.

Prabir Sarkar and Amaresh Chakrabarti. Studying engineering design creativity-developing a common definition and associated measures. In *Proceedings of the NSF Workshop on Studying Design Creativity*, page 20, 2008.

Prabir Sarkar and Amaresh Chakrabarti. Assessing design creativity. *Design Studies*, 32(4):348–383, July 2011. ISSN 0142-694X. doi: 10.1016/j.destud.2011.01.002.

Heinrich Schenker. *Free Composition (Der freie Satz)*. Longman Music Series. Longman, New York, NY, USA, 1935.

Jürgen Schmidhuber. A Formal Theory of Creativity to Model the Creation of Art. In Jon McCormack and Mark d'Inverno, editors, *Computers and Creativity*, pages 323–337. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-31727-9. doi: 10.1007/978-3-642-31727-9_12. URL https://doi.org/10.1007/978-3-642-31727-9_12.

D. Schön, L. Akiva-Kabiri, and T. Vecchi. *Psicologia della Musica*. Bussole: Psicologia. Carocci, Rome, Italy, 2007. ISBN 9788843040605. URL https://books.google.it/books?id=LZEYAQAAIAAJ.

Hendrik Schreiber and Meinard Müller. A Single-Step Approach to Musical Tempo Estimation Using a Convolutional Neural Network. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, page 8, Paris, France, September 2018.

Thomas Schürmann and Peter Grassberger. Entropy estimation of symbol sequences. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 6(3):414–427, September 1996. ISSN 1054-1500, 1089-7682. doi: 10.1063/1.166191.

Marco Scirea, Peter Eklund, Julian Togelius, and Sebastian Risi. Evolving In-game Mood-expressive Music with MetaCompose. In *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion*, AM'18, pages 8:1–8:8, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-6609-0. doi: 10.1145/3243274.3243292. URL http://doi.acm.org/10.1145/3243274.3243292. event-place: Wrexham, United Kingdom.

John R. Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3):417–424, September 1980. ISSN 1469-1825, 0140-525X. doi: 10.1017/S0140525X00005756.

V. Shah, M. Cuen, T. McDaniel, and R. Tadayon. A rhythm-based serious game for fine motor rehabilitation using leap motion. In *2019 58th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pages 737–742, 2019.

Noor Shaker, Gillian Smith, and Georgios N. Yannakakis. Evaluating content generators. In Noor Shaker, Julian Togelius, and Mark J. Nelson, editors, *Procedural Content Generation in Games*, Computational Synthesis and Creative Systems, pages 215–224. Springer International Publishing, Cham, 2016. ISBN 978-3-319-42716-4. doi: 10.1007/978-3-319-42716-4_12. URL https://doi.org/10.1007/978-3-319-42716-4_12.

Federico Simonetta, Filippo Carnovalini, Nicola Orio, and Antonio Rodà. Symbolic Music Similarity through a Graph-Based Representation. In *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion - AM'18*, pages 1–7, Wrexham, United Kingdom, 2018. ACM Press. ISBN 978-1-4503-6609-0. doi: 10.1145/3243274.3243301. URL http://dl.acm.org/citation.cfm?doid=3243274.3243301.

Dean Keith Simonton. Creativity: Cognitive, personal, developmental, and social aspects. *American psychologist*, 55(1):151, 2000.

S. N. Sivanandam and S. N. Deepa. Genetic algorithms. In *Introduction to genetic algorithms*, pages 15–37. Springer, 2008.

Dave Soldier. Eine Kleine Naughtmusik: How Nefarious Nonartists Cleverly Imitate Music. *Leonardo Music Journal*, pages 53–58, 2002.

Mark J Steedman. The Perception of Musical Rhythm and Metre. *Perception*, 6(5):555–569, October 1977. ISSN 0301-0066, 1468-4233. doi: 10.1068/p060555.

Mark J. Steedman. A Generative Grammar for Jazz Chord Sequences. *Music Perception: An Interdisciplinary Journal*, 2(1):52–77, October 1984. ISSN 07307829, 15338312. doi: 10.2307/40285282.

Morris I. Stein. Creativity and Culture. *The Journal of Psychology*, 36(2):311–322, October 1953. ISSN 0022-3980. doi: 10.1080/00223980.1953.9712897.

Brynjulf Stige. Aesthetic Practices in Music Therapy. *Nordisk Tidsskrift for Musikkterapi*, 7(2):121–134, January 1998. ISSN 0803-9828. doi: 10.1080/08098139809477932.

Bob L Sturm. What do these 5,599,881 parameters mean? An analysis of a specific LSTM music transcription model, starting with the 70,281 parameters of its softmax layer. In *Proceedings of the 6th International Workshop on Musical Metacreation*, page 8, 2018.

Bob L. Sturm, Joao Felipe Santos, Oded Ben-Tal, and Iryna Korshunova. Music transcription modelling and composition using deep learning. *arXiv preprint arXiv:1604.08723*, 2016.

Bob L. Sturm, Oded Ben-Tal, {\'U}na Monaghan, Nick Collins, Dorien Herremans, Elaine Chew, Gaëtan Hadjeres, Emmanuel Deruty, and François Pachet. Machine learning research that matters for music

creation: A case study. *Journal of New Music Research*, 48(1):36–55, January 2019. ISSN 0929-8215. doi: 10.1080/09298215.2018.1515233.

Bob L. T. Sturm and Oded Ben-Tal. Folk the algorithms: (mis)applying artificial intelligence to folk music. In Eduardo Reck Miranda, editor, *Handbook of Artificial Intelligence for Music: Foundations, Advanced Approaches, and Developments for Creativity*, pages 423–454. Springer International Publishing, Cham, 2021. ISBN 978-3-030-72116-9. doi: 10.1007/978-3-030-72116-9_16. URL https://doi.org/10.1007/978-3-030-72116-9_16.

Martin Supper. A few remarks on algorithmic composition. *Computer Music Journal*, 25(1):48–53, 2001.

Tim Swingler. The invisible keyboard in the air: An overview of the educational, therapeutic and creative applications of the EMS Soundbeam. In *2nd European Conference for Disability, Virtual Reality & Associated Technology*, pages 253–259, Skövde, Sweden, 1998. University of Reading.

Véronique Sébastien, Henri Ralambondrainy, Olivier Sébastien, and Noël Conruyt. Score analyzer: Automatically determining scores difficulty level for instrumental e-learning. In *13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, pages 571–576, 2012.

Kıvanç Tatar and Philippe Pasquier. Musical agents: A typology and state of the art towards Musical Metacreation. *Journal of New Music Research*, 48(1):56–105, January 2019. ISSN 0929-8215. doi: 10.1080/09298215.2018.1511736.

David Temperley. *The cognition of basic musical structures*. MIT Press, Cambridge, Mass., 1. paperback ed edition, 2004. ISBN 978-0-262-70105-1 978-0-262-20134-6. OCLC: 255948904.

David Temperley. Composition, Perception, and Schenkerian Theory. *Music Theory Spectrum*, 33(2): 146–168, October 2011. ISSN 01956167, 15338339. doi: 10.1525/mts.2011.33.2.146.

Peter M. Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4): 27–43, 1989.

Petri Toiviainen. An Interactive MIDI Accompanist. *Computer Music Journal*, 22(4):63–75, 1998. ISSN 0148-9267. doi: 10.2307/3680894.

E. P. Torrance. The Torrance tests of creative thinking, 1974.

E. Paul Torrance. Scientific Views of Creativity and Factors Affecting Its Growth. *Daedalus*, 94(3):663–681, 1965. ISSN 0011-5266.

E. Paul Torrance. The nature of creativity as manifest in its testing. *The nature of creativity*, pages 43–75, 1988.

Yvonne Treadwell. Humor and Creativity. *Psychological Reports*, 26(1):55–58, February 1970. ISSN 0033-2941. doi: 10.2466/pr0.1970.26.1.55.

Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, October 1950.

Gissel Velarde, Tillman Weyde, and David Meredith. An approach to melodic segmentation and classification based on filtering with the haar-wavelet. *Journal of New Music Research*, 42(4):325–345, 2013. doi: 10.1080/09298215.2013.841713.

Francisco Ventura, A. Oliveira, and Amílcar Cardoso. An emotion-driven interactive system. In *Portuguese Conference on Artificial Intelligence*, page 12, 2009.

Hugues Vinet. The representation levels of music information. In *International Symposium on Computer Music Modeling and Retrieval*, pages 193–209. Springer, 2003.

Richard F. Voss and John Clarke. "1/f noise"in music: Music from 1/f noise. *The Journal of the Acoustical Society of America*, 63(1):258–263, 1978.

Gregory M. Werner and Peter M. Todd. Too many love songs: Sexual selection and the evolution of communication. In *Fourth European Conference on Artificial Life*, pages 434–443. MIT Press/Bradford Books. Cambridge, MA, 1997.

Nick Whiteley, A. Taylan Cemgil, and Simon Godsill. Bayesian Modelling of Temporal Structure in Musical Audio. In *ISMIR*, pages 29–34, Victoria, Canada, 2006. University of Victoria.

Gerhard Widmer and Werner Goebl. Computational models of expressive music performance: The state of the art. *Journal of New Music Research*, 33(3):203–216, 2004.

G. A. Wiggins. Computer Models of Musical Creativity: A Review of Computer Models of Musical Creativity by David Cope. *Literary and Linguistic Computing*, 23(1):109–116, 2007a. ISSN 0268-1145, 1477-4615. doi: 10.1093/llc/fqm025.

Geraint A. Wiggins. A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems*, 19(7):449–458, November 2006. ISSN 0950-7051. doi: 10.1016/j.knosys.2006.04.009.

Geraint A. Wiggins. Models of musical similarity. *Musicae Scientiae*, 11(1_suppl):315–338, 2007b. doi: 10.1177/102986490701100112.

Geraint A. Wiggins. A Framework for Description, Analysis and Comparison of Creative Systems. In Tony Veale and F. Amílcar Cardoso, editors, *Computational Creativity: The Philosophy and Engineering of*

Autonomously Creative Systems, Computational Synthesis and Creative Systems, pages 21–47. Springer International Publishing, Cham, 2019. ISBN 978-3-319-43610-4. doi: 10.1007/978-3-319-43610-4_2. URL https://doi.org/10.1007/978-3-319-43610-4_2.

Geraint A. Wiggins. Creativity, information, and consciousness: The information dynamics of thinking. *Physics of Life Reviews*, 34-35:1–39, December 2020. ISSN 1571-0645. doi: 10.1016/j.plrev.2018.05.001.

Geraint A Wiggins. Computational creativity and consciousness: Framing, fiction and fraud paper type: Study paper. In *Proceedings of the 12th International Conference on Computational Creativity (ICCC '21)*, pages 182–191, 2021a.

Geraint A. Wiggins. Structure, abstraction and reference in artificial musical intelligence. In *Handbook of Artificial Intelligence for Music*. Springer Nature Switzerland AG, Cham, 2021b. doi: https://doi.org/10.1007/978-3-030-72116-9_15.

Geraint A. Wiggins and Jamie Forth. IDyOT: A Computational Theory of Creativity as Everyday Reasoning from Learned Information. In Tarek R. Besold, Marco Schorlemmer, and Alan Smaill, editors, *Computational Creativity Research: Towards Creative Machines*, Atlantis Thinking Machines, pages 127–148. Atlantis Press, Paris, 2015. ISBN 978-94-6239-085-0. doi: 10.2991/978-94-6239-085-0_7. URL https://doi.org/10.2991/978-94-6239-085-0_7.

Geraint A. Wiggins and Abdelrahman Sanjekdar. Learning and Consolidation as Re-representation: Revising the Meaning of Memory. *Frontiers in Psychology*, 10, 2019. ISSN 1664-1078. doi: 10.3389/fpsyg.2019.00802.

Geraint A. Wiggins, George Papadopoulos, Somnuk Phon-Amnuaisuk, and Andrew Tuson. *Evolutionary methods for musical composition*, volume UNPUBLISHED. University of Edinburgh, Department of Artificial Intelligence, 1998.

Geraint A. Wiggins, Peter Tyack, Constance Scharff, and Martin Rohrmeier. The evolutionary roots of creativity: mechanisms and motivations. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 370(1664), 2015. ISSN 0962-8436. doi: 10.1098/rstb.2014.0099.

Duncan Williams, Alexis Kirke, Eduardo R. Miranda, Etienne Roesch, Ian Daly, and Slawomir Nasuto. Investigating affect in algorithmic composition systems. *Psychology of Music*, 43(6):831–854, 2015.

Anna Xambó, Johan Pauwels, Gerard Roma, Mathieu Barthet, and György Fazekas. Jam with Jamendo: Querying a Large Music Collection by Chords from a Learner's Perspective. In *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion*, AM'18, pages 30:1–30:7, New York, NY, USA, 2018.

ACM. ISBN 978-1-4503-6609-0. doi: 10.1145/3243274.3243291. URL http://doi.acm.org/10.1145/3243274.3243291. event-place: Wrexham, United Kingdom.

Gus G. Xia and Roger B. Dannenberg. Improvised Duet Interaction: Learning Improvisation Techniques for Automatic Accompaniment. In *NIME*, page 5, Copenhagen, Denmark, 2017. Aalborg University.

Li-Chia Yang and Alexander Lerch. On the evaluation of generative models in music. *Neural Computing and Applications*, November 2018. ISSN 0941-0643, 1433-3058. doi: 10.1007/s00521-018-3849-7.

Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847*, 2017.

J. Diana Zhang and Emery Schubert. A Single Item Measure for Identifying Musician and Nonmusician Categories Based on Measures of Musical Sophistication. *Music Perception*, 36(5):457–467, 06 2019. ISSN 0730-7829. doi: 10.1525/mp.2019.36.5.457.

Yixiao Zhang and Gus Xia. Symbolic melody phrase segmentation using neural network with conditional random field. In *National Conference on Sound and Music Technology*, pages 55–65. Springer, 2020.

Guo Zixun, Dimos Makris, and Dorien Herremans. Hierarchical recurrent neural networks for conditional melody generation with long-term structure. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021. doi: 10.1109/IJCNN52387.2021.9533493.