IT UNIVERSITY OF COPENHAGEN

DOCTORAL THESIS

Affective Music Generation and its effect on player experience

Author:
Marco SCIREA

Supervisors:
Dr. Julian TOGELIUS
Prof. Peter EKLUND
Dr. Sebastian RISI

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Center for Computer Games Research
Digital Design Department

August 31, 2017

Declaration of Authorship

I, Marco SCIREA, declare that this thesis titled, "Affective Music Generation and its effect on player experience" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Date: 31/08/2017

"People usually complain that music is so ambiguous; that it is so doubtful what they ought to think when they hear it; whereas everyone understands words. With me it is entirely the converse.... The thoughts which are expressed to me by a piece of music which I love are not too indefinite to be put in words, but on the contrary too definite. "

Felix Mendelssohn

IT UNIVERSITY OF COPENHAGEN

Abstract

Center for Computer Games Research
Digital Design Department

Doctor of Philosophy

Affective Music Generation and its effect on player experience

by Marco SCIREA

Procedural content generation for games –the automated creation of some type of asset– has become increasingly popular in the last decade, both with academics and game developers. This interest has been mainly motivated by how complex games have become, requiring a huge amount of assets to be created to support them. While most focus has been put on generating levels, textures, and 3D models, there are also examples of games that generate music dynamically, which is the focus of this thesis (e.g. *Spore*). In games, unlike in traditional linear storytelling media such as novels or films, narrative events unfold in response to player input. Therefore, the music composer in an interactive environment needs to create music that is dynamic and non-repetitive. This thesis investigates how to express emotions and moods in music and how to apply this research to improve player experience in games. This focus on the emotional expression that procedurally generated music should express has also been identified by Collins [1] as one of the missing features that currently prevent procedurally generated music being more widely used in the game industry. The research therefore focuses on investigating the expression of moods, and the effect affective music can have on the listener during game play.

In this thesis three systems are described: the METACOMPOSE affective music generator, its prototype, and a system for co-evolution of improvisational modules (Primal-Improv). The characteristics of METACOMPOSE are: (i) expressing different affective states using a variety of AI-techniques, (ii) generating such music in real-time, and (iii) reacting in real-time to external stimuli. Its architecture is comprised of three main components: the composition generator, the real-time affective music composer and an archive of compositions. A novel feature of our approach is the separation of composition and affective interpretation: the system creates abstractions of music pieces (called *compositions*) and interprets these in real-time to achieve the desired affective expression, while simultaneously introducing stochastic variations. Notably, the abstraction generation component includes a graph traversal-based chord sequence generator, a search-based melody generator and a pattern-based accompaniment generator. The melody generation uses a novel constrained multi-objective evolutionary technique combining FI-2POP and Non-dominated Sorting Genetic Algorithm (NSGA-II). The thesis presents the results of several evaluation studies, evaluating both the design of the systems, their affective expression, and exploring the effects on player experience when integrating METACOMPOSE with the game of Checkers.

Proceduremæssig indholdgeneration for videospil – den automatiske skabelse af nogle typer assets har opnået en stigende popularitet indenfor det sidste årti, både akademisk og hos spiludviklere. Motivationen bag stammer fra hvor komplekse videospil er blevet, da der kræves en underbyggelse af store mængder assets. Selvom størstedelen har fokus på at generere levels, teksturer og 3D modeller, har der også været eksempler på spil der dynamisk har kunnet generere musik. Dette er hvad denne afhandling vil fokusere på. I modsætning til traditionelt lineær historiefortælling i medier såsom bøger og film, så udfolder narrative begivenheder i spil sig som respons til spillerens handlinger. Derfor er der behov for, at musik komponisten, i et interaktivt miljø, skaber musik som er dynamisk og ikke gentagende. Denne afhandling undersøger hvordan man kan udtrykke følelser og humør i musik, og hvordan man kan anvende denne forskning til at forbedre spilleroplevelsen i videospil. Fokus på det følelsesmæssige udtryk, som proceduelt genereret musik kan skabe, er blevet identificeret af Collins [1], som værende en af de manglende funktioner der på nuværende tidspunkt forhindrer proceduelt genereret musik i at blive brugt oftere i spilindustrien. Denne forskning fokuserer derfor på at undersøge udtrykkelsen af humør, og den effekt affektiv musik kan have på spilleren under spillet.

I denne afhandling er tre systemer beskrevet: the METACOMPOSE affektive musik generator, dens prototype, og et system for co-evolution af improviseret moduler (Primal-Improv). Karakteristikkerne af METACOMPOSE er: (i) at udtrykke forskellige affektive stadier ved brug af et assortiment af AI-teknikker, (ii) generer sådant musik i realtid, og (iii) reagerer i realtid til eksterne stimuli. Dens arkitektur består af tre hovedkomponenter: komposition-generatoren, den i realtid affektive musik komponist og et arkiv af kompositioner. En innovativ funktion af vores tilgang er adskillelsen af komposition og den affektive fortolkning: systemet skaber abstraktioner af musikstykker (kaldet *kompositioner*), og fortolker disse i realtid for at opnå den ønsket affektive udtrykkelse, mens den samtidig introducerer stokastiske variationer. Bemærkelsesværdigt er det, at komponenten for abstraktionsgeneration inkluderer en graf-baseret akkordsekvensgenerator, en søgningsbaseret melodigenerator og en mønstre-baseret akkompagnementgenerator. Denne melodigenerator benytter en nyskabende begrænset multi-objektive evolutionsteknik, der kombinerer FI-2POP og Nondominated Sorting Genetic Algorithm (NSGA-II). Denne afhandling præsenterer resultatet af adskillige evalueringsstudier, der både har undersøgt systemets design, dets affektive udtrykkelse og udforsket effekten det har haft på spiloplevelsen når METACOMPOSE er integreret i spillet Dam.

Acknowledgements

Firstly, I would like to thank my supervisors: Julian Togelius, Peter Eklund, and Sebastian Risi. Their trust in my vision, the great discussions, and feedback they provided me, helped immeasurably to make me the academic that I am today. Their input and suggestions have helped steer my research and shaping this thesis in what it is today. I am especially grateful to the *Center for Computer Games Research* and the *Robotics, Evolution and Art Lab* and all its members for providing discussion, helpful criticism, and even more helpful experimental subjects. More specifically, I would like to thank Espen Aarseth, Miguel Sicart, Hans-Joachim Bache, Martin Pichlmair, Rune Kristian Lundedal Nielsen, Laura Beloff, Kasper Støy, Niels Justensen, Frank Veenstra, Stig Anton Nielsen, and so many more to list here. I also thank the members of my examination committee: Philippe Pasquier, Palle Dahlstedt, and Miguel Sicart. Of course, this thesis wouldn't have been possible without the support of the IT University of Copenhagen, that supported me during these three fantastic years both financially and by providing a great working environment. I would like to extend a special thanks to ITU's Ph.D. school, for supporting my travel to many conferences and for providing quick responses to my many many questions. I would like to show my gratitude to Georgios Yannakakis and the Institute of Digital Games at the University of Malta for hosting me during the first part of my stay abroad period and for giving me excellent supervision, in particular I want to thank the lovely people of the institute for welcoming me and providing with useful insights: Antonios Liapis, Daniel Vella, Costantino Oliva, Phil Lopes, Daniele Gravina, and Daniel Karavolos. I also want to underline the contribution of Georgios Yannakakis and Phil Lopes in providing discussions that led to the design of the experiment described in Chapter 7. Of course I must also thank the Game Innovation Lab at New York University, which hosted me for the second part of my stay abroad, in particular Andy Nealen, Christoffer Holmgaard, Ahmed Khalifa, Fernando de Mesentier Silva, Aaron Isaksen, Philip J Bontrager, and many more. It would be a crime not to thank my "academic sister" Gabriella Barros, that shared an office with me before she moved to the Game Innovation Lab; I will forever cherish that I could always turn to her to discuss research topics and any kind of shared interest, and the great fun we had together, even from afar. Finally, I have to thank my girlfriend Anita Simonsen, my family, and my friends for their continued support over the years. They all played a big part in maintaining my sanity and their continued encouragement, especially through stressful times, has been inestimable. Without all these people (and many more that I have not mentioned) I'm sure this thesis would have been very different.

Contents

List of Figures

List of Tables

For/Dedicated to/To my…

Chapter 1

Introduction

The game “The Witcher 3" (CD Project RED, 2015), a fantasy role playing game based on the novels by Polish author Andrzej Sapkowski, has been hailed as one of the best games of the last few years. The game has a very long and complex story which puts the player in front of non-trivial choices with long lasting consequences. While the music produced for the game is also of very high quality, as I was 10 to 20 hours in the game, my immersion started getting broken by the same tracks being repeated all over. Is it possible that in every inn the same music is always played? Another issue that should be highlighted is the transitions between different pieces, which can be detrimental to immersion if not dealt with properly. In commercial games transitions are often achieved using simple crossfade, or a sound effect to mask the transition (e.g. an explosion, or alarm) [2]. More high-budget projects can also have a set of intro/outro to each piece and specific transitions written between pieces but, as you can imagine, this approach greatly increases the complexity and the amount of music that the composer has to produce, subsequently increasing production costs.

The many reasons to build computer systems that can competently generate music: adapting to a dynamic environment, performing concurrently with a human player, reflecting upon music composition practice, and many more [3]. Music can express and evoke moods and emotions (even music generated algorithmically [4]), yet expressing affective states is one of the least explored aspects of music producing systems. In some cases the main purpose of a music generation algorithm is to evoke a particular mood. This is especially true for music generators that form part of highly interactive systems, such as those supporting computer games. In such systems a common goal of music generation is to elicit a particular mood that dynamically suits the current state of the game-play.

Computer games have properties that make them particularly interesting and challenging for this kind of music generation: unlike traditional sequential media, such as novels or movies, events unfold in response to player input rather than a linear narrative. Therefore, a music composer for an interactive environment needs to create music that is dynamic, while also holding the listener’s interest and avoiding repetition. This applies to a wide range of games although not all;

for example rhythm games (e.g. *Guitar Hero*) make use of semi-static music around which the game-play is constructed [5], [6]. Most commercial games use a range of techniques that consist of either fading in new tracks (or layers) when some specific event happens (e.g. play the "combat music" when the player encounters an enemy). These techniques are not very flexible, as the pieces of music are static human composed ones and can lead to excessive repetition.

Moreover, creating dynamic music is not enough: music can be seen as a communication tool and we believe that it is important to use it to convey meaning. This research explores how to express narrative events through mood expressive music. It is important to note that in this domain there is not only a predefined narrative, but also an emergent one dictated by the player's actions.

We aim to fill what we believe is the gap that is holding procedural music generation back: emotion expression. Karen Collins, a leading researcher in game audio, has published multiple books on the topic. In her book *Game Sound* she identifies affective expression as one of the lacking features that prevent procedurally generated music from being more widely used in the game industry [1]. A number of works have been published in the area of affect, semiotics and mood-tagging [7]–[9] but our focus lies in the real-time generation of background music capable of expressing moods.

Music generation for computer games can also be seen as an instance of the experience-driven procedural content generation framework (*EDPCG*) [10], where the game adaptation mechanism generates music with a particular mood or affect expression in response to player actions. As such, this research is also a step forward towards complete game generation: one of the most complete of these systems is Angelina [11] but it deals with the music aspect with just a selection of human composed pieces.

1.1 Research Question

Ultimately, the question that this research aims to answer is:

> **How can music be automatically generated in games in order to influence player experience and express affect?**

Expecting a definite answer to this question might be asking too much from a single thesis, as it would require many more studies than what feasible in the allotted timespan. We have tried to build up to this question by breaking it up into smaller questions:

Q1 How can different moods be expressed in music in a computationally feasible and reliable way?

Q2 How can a music generation system create convincing dynamic music while also expressing different moods?

Q3 How do players react to exposure to such music while playing a game?

We developed a mood expression theory that maps some musical features (*tempo, volume, timbre, rhythmic strength, regularity and dissonances*) to Russell's bi-dimensional affective model [12]. We hypothesize that each of the features is connected to either the valence or the arousal dimension.

This thesis describes the structure of two music generators which use our mood expression theory:

- Moody Music Generator: a prototype that produces ambient semi-random music.
- METACOMPOSE: the final product of this Ph.D. study, it is able to create music in real-time that can express different mood-states.

METACOMPOSE presents a novel structure where, to achieve the flexibility needed to express different moods, an abstraction of music is generated. These abstractions can be transformed in the complete score in real-time through an improvisational component consisting of some algorithms that decide what to play at any given time based on the abstraction.

It is important to note that the purpose of METACOMPOSE is to produce music to be used for background in an interactive/dynamic experience, as such it does not have to be as complicated and structured as, for example, a classical piece of music. The music generator has rather been designed to create small, loopable compositions with the focus on being able to dynamically change the affective expression. The moods are directed by an interactive application (e.g. a game) and the system has to be able to change between expressed moods fast enough to enable real-time interaction between the user and the music. The system builds on experience from several earlier prototypes [13], [14] which experimented with affective expression using simpler musical structures.

1.2 Main Contributions

This thesis has produced several contributions to the research fields of algorithmic music and evolutionary computation, summarized below:

- The formalization of a mood expression theory based on the manipulation of a relatively small amount of parameters.
- The introduction of Two-population Non-dominated Sorting Genetic Algorithm (2POP-NSGA). This algorithm combines constrained and multi-objective search allowing the search of solutions to problems that don't have an obvious objective while

giving a degree of control on the evolved individuals through constraints.

- The introduction of a novel architecture for music generation that allows for real-time rendering of a music piece with any kind of affective expression allowed by the theory.
- The exploration of emergent music from co-evolutionary techniques using very simple criteria for evaluation (Primal-Improv).
- The formalization of a novel method of crowdsourcing annotations to characterize control parameters that is based on free text answers to avoid bias.

1.3 Publications

Eight papers were published while trying to answer the questions delineated in Section 1.1, which are reported here. Two additional papers relating to preliminary work done during my Master studies have been omitted from the list, as they were not part of my Ph.D. studies [13], [15]. While each of these papers is self-contained and addresses a specific problem, their content is the basis of most of the content of this thesis.

1. Scirea Marco, Julian Togelius, Peter Eklund, and Sebastian Risi. "Affective Evolutionary Music Composition with MetaCompose" in the Genetic Programming and Evolvable Machines journal (2017). Included in Chapter 5 and Chapter 6.
2. Scirea Marco, Peter Eklund, Julian Togelius and Sebastian Risi. "Can You Feel It? Evaluation of Affective Expression in Music Generated by MetaCompose" in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) (2017). Included in Chapter 6.
3. Scirea Marco, Peter Eklund, Julian Togelius and Sebastian Risi. "Primal-Improv: Towards Co-Evolutionary Musical Improvisation" to appear in Proceedings of the Computer Science and Electronic Engineering Conference (CEEC) (2017). Included in Chapter 8.
4. Scirea Marco, Julian Togelius, Peter Eklund and Sebastian Risi. "MetaCompose: A Compositional Evolutionary Music Composer" in Proceedings of Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMusArt) (2016) **[Best Paper Award]** . Included in Chapter 5 and Chapter 6.

5. Scirea Marco and Joseph Alexander Brown. "Evolving Four Part Harmony Using a Multiple Worlds Model." in Proceedings of International Conference on Evolutionary Computation Theory and Applications (2015). Not included in the thesis.

6. Scirea Marco, Gabriella A. B. Barros, Noor Shaker and Julian Togelius. "SMUG: Scientific Music Generator." in Proceedings of International Conference on Computational Creativity (2015). Not included in the thesis.

7. Scirea Marco, Mark J. Nelson and Julian Togelius. "Moody Music Generator: Characterising Control Parameters Using Crowdsourcing." in Proceedings of Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMusArt) (2015) **[Nominated for Best Paper Award]**. Included in Chapter 4.

8. Scirea Marco, Byung Chull Bae, Yun-Gyung Cheong and Mark Nelson "Evaluating musical foreshadowing of videogame narrative experiences." in Proceedings of Audio Mostly (2014). Included in Chapter 4.

1.4 Thesis Structure

First we provide an exhaustive literature review of related works in Chapter 2, presenting theories and methods that are used to support the presented research. In Chapter 3 we describe specific algorithms that have been applied to, or developed for, the work presented in this thesis. In Chapter 4 we describe the prototype music generator, the *Moody Music Generator*, and the evaluation studies conducted on it. After discussing the limitations of the prototype that became apparent through the conducted experiments, we describe in Chapter 5 the design of the system that was created to address these: METACOMPOSE. In this chapter, we also include the description of the mood expression theory used by the system. The various experiments conducted on METACOMPOSE, to evaluate the architecture of the system and its expressive qualities, can be found in Chapter 6. In Chapter 8 we describe the co-evolutionary improvisation system Primal-Improv and an initial evaluation of the produced musical artefacts. In Chapter 7 an experiment on the effect of METACOMPOSE's dynamic affective music on the perception of the game of Checkers is presented and discussed. Finally in Chapter 9 we conclude by putting together all the results obtained to discuss the hypothesis and describe the final impact of our research on the field. This chapter also includes discussion on this thesis' findings, looking at limitations, extensibility, and future avenues of research.

Chapter 2

Related Work

This chapter serves to contextualize the research presented in this thesis. As stated in the introduction, the main objective is to explore how to create generative music that can express affective states; therefore it is relevant to introduce a brief review of relevant research such as: music generation, procedural content generation in games, computational creativity, and affect theory. Not all the work discussed here has been used in the thesis, but serves to give better context to the research. In particular, the work on player modelling is not applied to this thesis, but is relevant since future work would include it (see Chapter 9.2).

While this chapter focuses on literature review and more theoretical approaches, Chapter 3 will present related work relevant to the algorithms specifically used in the systems described in this thesis.

2.1 Evolutionary Computation

The main algorithm developed during this thesis, and used for melody generation, is based on evolutionary computation. Evolutionary Computation is a field that encompasses the use of techniques that use artificial evolution, which are inspired by real-world evolution described by Darwin [16]. The driving force of evolution is what Darwin describes as "the survival of the fittest": the idea that the most proficient individuals in a species (or the ones that present some advantageous trait) will be able more likely to reproduce, and thus transmit these traits to their offspring. With the continuous repetition of this process the species will eventually change to adapt to the environment, becoming more and more adept to thrive and survive. Since the 1960s different research has been using these principles: biologists wanting to test evolutionary models in simulations, computer scientists trying to create better artefacts or solutions to problems, and artificial life researchers, just to cite a few [17].

As might be imagined, so many applications and approaches have led to many different variants of evolutionary algorithms. Yet the core process remains more or less the same (see also Algorithm 1): at first a population is initialized randomly, then it is evaluated with a fitness measure. Some of the best individuals are used as seeds for

the next generation; this is done through the application of recombination and/or mutation operators, leading to a new set of candidates (offspring) that will then compete for position in the next generation. This process is repeated until a termination condition is met. This condition is often either a target fitness value, or that a specified number of generations that have passed.

Algorithm 1 General scheme of an Evolutionary Algorithm as pseudo-code

```
1: BEGIN
2: INITIALISE population with random candidate solutions;
3: EVALUATE each candidate;
4: while TERMINATION CONDITION is satisfied do
5:     SELECT parents;
6:     RECOMBINE pairs of parents;
7:     MUTATE the resulting offspring;
8:     EVALUATE new candidates;
9:     SELECT individuals for the next generation;
10: end while
11: END
```

The main forces that influence the evolution in this system, apart from the fitness function, are the variation operators (recombination and mutation) and the selection method. Recombination is an operator applied to two or more individuals (the "parents") that results in one or more individuals (the offspring). Mutation is only applied to one individual and results in a new individual that generally is very similar to the original one.

The main components of evolutionary algorithms are:

Representation: the first step in defining an EA is to define a representation – how can we relate the real-world problem to the solution space. In the original problem context, possible solutions are defined as **phenotypes**, while their encodings are called **genotypes**. A biological example would be to consider each of us humans as phenotypes, each of which is encoded by our DNA (genotype). The choice of representation can greatly affect the evolutionary process, depending on how well the genotype reflects the characteristics that the phenotypes need to solve the problem.

Evaluation function: also referred to as *fitness function*, has the role of representing the requirements to which the population of solutions should adapt. This assigns a quality measure to the genotypes, based on an evaluation of their phenotypes. In case of multi-objective problems – where there are more than one evaluation functions – the terminology **objective function** is used. This is often the most critical part of the EA, as it has

to provide (ideally) a smooth gradient that evolution can use to progress towards better solutions. As an example, in the domain of evolving agents to play games, an intuitive fitness function is the score achieved by the agent. Yet, this score might hide many dimensions that will not be properly observed by the fitness function. A possible solution to this is multi-objective algorithms, which we will describe in Section 2.1.2.

Selection: the role of selection is to choose between the current individuals (based on their fitness), namely which ones will be used to create more offspring. Typically this process is probabilistic, giving a higher chance to high fitness individuals to be chosen as parents, while still allowing the possibility of lower fitness parents being selected.

Mutation: this is a unary operator which, given a genotype, returns a slightly modified one. This is a stochastic operator that acts as local search: exploring small variations of an individual, and theoretically not moving very much in the search-space.

Recombination: this is an operator between two or more individuals (often called **crossover**) that is used to create new individuals from parents. The most common implementation is the single-point crossover, where the two parents' genomes are cut at a specific index and two new individuals are created by the combination of the first part of the first parent and second part of the second parent, and vice-versa. Compared with the mutation operator, recombination is a much more global search operator as – while although the offspring present some characteristics of the parents – they can also be considerably different from them.

Replacement: finally, the replacement mechanism has the role of distinguishing among individuals based on their quality, similarly to the selection mechanism. The difference lies in the stage of the evolutionary process in which it is used. The replacement mechanism is used once the mutation and recombination operators have been applied; usually at this stage there is a surplus of individuals compared to the maximum population size which has to be removed. Different from the selection mechanism, replacement is often deterministic, often just selecting the best individuals and removing the worst.

2.1.1 Constrained Optimization

Problems that include constraints on their solutions are very common; while evolutionary computation has been very successful in solving numerical optimization problems, it has never had a straightforward method to deal with constraints. Examples of constraints

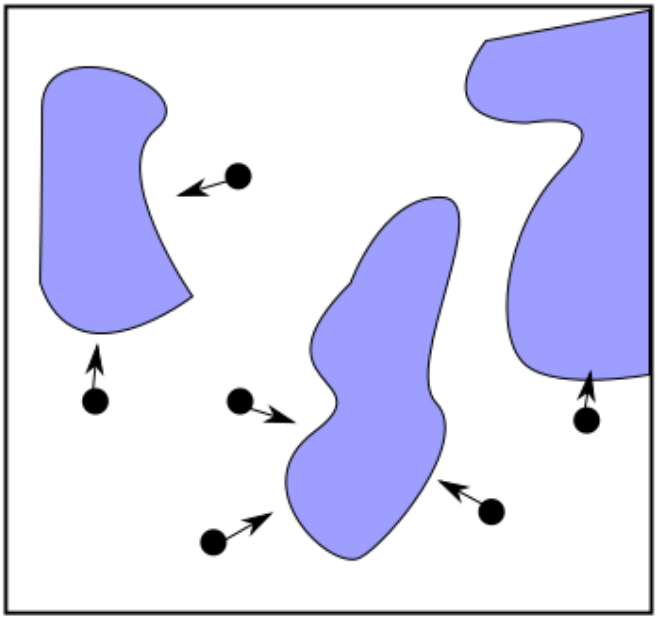

FIGURE 2.1: Example of the solution space for a constrained problem. The space is divided in two different areas: the feasible solution (in blue) and the infeasible solutions. In this type of problem there is the added complexity that feasible areas might be far away from each other, so the algorithm has to balance searching through the infeasible solution space for new feasible areas, and finding better solutions inside a known feasible area.

might be not only inherent in the problem, but can also derive from practical needs such as: performance, production costs, and size. Constrained problems present a fractured solution space, where possible solutions are not only differentiated by how well they solve the problem but also if they satisfy the constraints (feasible solutions) or not (infeasible solutions), see Figure 2.1. Depending on the constraints, the solution space might consist of large areas of feasible solutions or small islands of feasible solutions separated by infeasible ones. The main challenge in constrained optimizations is to handle the infeasible solutions as, while not being acceptable solutions, their inclusion might lead to the discovery of parts of the feasible solution space. Another challenge of these problems is that often the optimal solution might lie on the boundary between feasible and infeasible solution spaces [18], for example a solution that is as close to the performance minimum while satisfying a maximum cost constraint.

One of the earliest approaches proposed to deal with constraints was the "death penalty": during the evaluation of the solution, if a solution is found that does not satisfy all constraints it would get

assigned a fitness value of zero (or a value equivalent minimum fitness value). While this method might work for solution spaces that consist mostly of feasible solutions, it can lead to a loss of genetic material that can make it difficult for the GA to find isolated "islands" of feasible solutions [19]. Another issue with this approach arises when a sufficient number of feasible solutions cannot be generated as the population is initiated, as it would lead to a population with all equal fitness values, reducing the GA to a random search.

The most common method for constraint handling is to introduce a *penalty score* for individuals that do not satisfy constraints. This method allows infeasible individuals to survive in the generative process, while still giving priority to the actual objective. The issue with this approach is the definition of such *penalty scores*: too stringent penalties lead to the death penalty approach, too lenient ones lead to superfluous search in the infeasible search space. These penalties can be implemented in various ways [20], from a simple constant value [21], to a measure of feasibility [22], to a dynamically adapted measure [23]. In the event that the infeasible solutions cannot be evaluated by the fitness function (e.g. if they have to represent a valid 3D shape), a *repair* function can be put in place to transform them into feasible solutions. Often the number of operations needed to repair an individual is used as penalty for its fitness score. This approach can be complicated, as designing repair functions requires significant domain knowledge; moreover the repair function might change the solution in ways that might not be representative of its genome, leading the evolutionary search astray.

Another approach, which we will describe in detail in Section 3.2, consists in evolving feasible and infeasible populations separately using the Feasible-Infeasible two population method (FI-2pop) [24].

2.1.2 Multi-Objective Optimization

Multi-Objective Optimization (MOO) is defined as the process of simultaneously optimizing multiple objective functions. In most multi-objective optimization problems, there is no single solution that simultaneously optimizes every objective. In this case, the objective functions are said to be partially conflicting, and there exists, a number (possibly infinite) of Pareto optimal solutions. A solution is called "non-dominated", Pareto optimal, Pareto efficient or "non-inferior", if none of the objective functions can be improved in value without degrading some other objective values (see Figure 2.2). Therefore, a practical approach to multi-objective optimization is to investigate a set of solutions (the best-known Pareto set) that represent the Pareto optimal set as much as possible [25]. Many Multi-Objective Optimization approaches using Genetic Algorithms (GAs) have been developed. The literature on the topic is vast; Coello lists more than

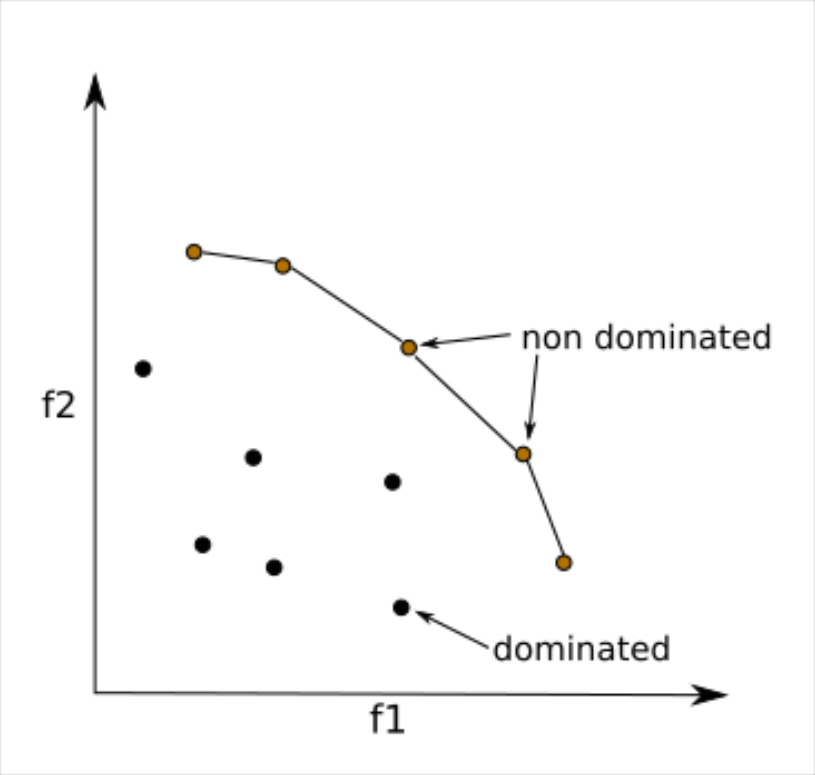

FIGURE 2.2: Example of a Pareto front for a 2-objective ($f1$ and $f2$) maximization problem. The elements that form the Pareto front are all the solutions that are not dominated by any other solutions found. A solution is non-dominated when no other solution presents better evaluation values for all objectives.

2000 references on this topic on his website[1].

Many creative tasks can be categorised as multi-objective problems, as it is unlikely that a specific measure of the quality (or creativity) of an artefact can be defined. This is particularly true in artistic fields, as often specifically desired features can be defined, but the relationships between them is much harder to characterize. In this type of problems, instead of weighting each feature, it is preferable to consider them as different objectives and allow the search-algorithm to explore the combinations of these.

Our approach builds on the successful and popular NSGA-II algorithm [26]. The objective of the NSGA-II algorithm is to improve the adaptive fit of a population of candidate solutions to a Pareto front, constrained by a set of objective functions. The population is sorted into a hierarchy of sub-populations based on the ordering of Pareto dominance. Similarity between members of each sub-group is evaluated on the Pareto front, and the resulting groups and similarity measures are used to promote a diverse front of non-dominated solutions. A more in-depth description of NSGA-II can be found in

[1] http://www.cs.cinvestav.mx/~constraint/papers/

Section 3.1

2.1.3 Co-evolution: cooperative/competitive

Co-evolution is a process of reciprocal genetic change in one species in response to another; this process can be observed in nature where each party exerts selective pressures on the other, as already observed by Darwin in the evolutionary interactions between flowering plants and insects [16]. Co-evolution has also been applied to evolutionary algorithms and in general is divided into two categories: as a competitive arms race [27] or as a cooperative effort. An example of the former is the co-evolution of test cases for a problem (predators) with solutions (prey) [28]. Conversely, in cooperative approaches the populations can be seen as components of the final solution [29], [30].

An interesting application of co-evolution applied to music can be found in *Living Music* [31], a system where agents are populating a virtual world and have to produce sounds to mate. The aggregation of the sounds leads to the music produced by the system. This system shares the philosophy that for a system to be creative it should create structures that one cannot immediately envision, and that from simple rules complex behaviours can often arise. A key difference between the objective of *Living Music* and the Primal-Improv system is that our focus is on the interaction between human composed music and the system, while *Living Music* creates completely autonomous music.

2.2 Procedural Content Generation

Procedural content generation (PCG) describes the use of algorithms to automatically generate some kind of content; this content can be any aspect of creative works including computer games, for example in games: levels, items, textures, models, game mechanics, etc., can be procedurally generated. PCG has been used in the computer games industry since the 1980s, the most famous early examples being the dungeon (level) generation of *Rogue* (Michael Toy and Glenn Wichman, 1980), and the creation of entire galaxies to explore in *Elite* (David Braben and Ian Bell, 1984). At that time, the main reason for using algorithms to generate content was to escape the limitations of the hardware: for example, the limitations on memory did not allow game designers to include many visual assets in their games. Since then, the processing power and available storage of computers has increased dramatically, in some sense defeating the initial argument for PCG. Today, PCG is used with a higher focus on providing the player with new content every time they play the game to enhance re-play value.

Togelius *et al.* [32] propose a taxonomy of algorithms used for generating content. They describe three main families: constructive algorithms, generate-and-test algorithms, and search-based algorithms.

The main characteristic of constructive algorithms is that the artefacts generated by the system are immediately presented to the user without going through a validation phase. This does not mean that the created content is "random" or of low quality, but rather that the algorithm is specifically designed to incorporate some rules to create content that is both playable and that makes sense in the context of the game. For example *Borderlands* series (Gearbox Software, 2009-2015) presents a weapon generation system that creates new weapons from a combination of different modules, yet never creates game-breaking ones and scales the power of the weapons with the player level. This kind of generation is generally based on the use of a random number generator to pick between possible designer-created options. One of the earlier games that made this approach famous is *Diablo*, where multiple random rolls are used to create the weapons and equipment that the player will find. Perlin noise [33] has had many applications as part of the generation process of terrains, textures, shaders, etc. An important commercial example is *Minecraft*, where the algorithm plays a role in terrain generation. A more complex constructive PCG can be found in games that create worlds for the player to explore when starting the game (*Minecraft*, *Dwarf Fortress*, *Don't starve*, etc.) where the final content is governed by many rules and constraints, for example what kind of topology should a specific area (biome) possess and what kind of enemies/objects will be found in it.

Generate-and-test algorithms, as opposed to constructive algorithms, do a validation of the content quality after generating it. The quality is generally evaluated as the satisfaction of some constraints, be it gameplay-related (Does the artefact break the game? Is it balanced?) or technology-related (Is the artefact well formed? Can it be rendered properly?). Usually if the content is found not to satisfy all the constraints it is discarded and the generation process rerun. The main challenge in designing these kinds of algorithms is the definition of such constraints: too stringent constraints might never be satisfied, too loose constraints might allow for content with high quality variability. Of course this is also dependent on the qualities of the generation process: one with a wide range of expression might be more difficult to evaluate, as the number of variables that impact quality can be large. Note that these first two categories are not necessarily mutually exclusive; in fact, Dwarf Fortress, in the first step of its generation process uses a generate-and-test approach where it discards generated maps where they do not satisfy some constraint.

Finally search-based algorithms use – as with the generate-and-test family – an evaluation module, but rather instead of testing the

final product they evaluate potential candidates and use this evaluation to guide the generation process. From the description given here, it can be immediately seen how this approach is well-suited for evolutionary computation, a family of algorithms that abstract the process of biological evolution. A longer description of evolutionary computation can be found in Section 2.1. The main challenge in this approach is the definition of the evaluation function. Contrary to the generate-and-test approach, a binary response (constraint satisfied/unsatisfied) is not opportune any more, as the generation process needs to be able to distinguish between gradations of fitness to be able to properly guide the search for better artefacts. Such evaluation is generally described by a mathematical formulation inspired by expert knowledge, game theory, simulation (e.g. observing an agent interact with the artefact), or by comparing the content generated with a sample "ideal" archetype. Another option is to have the player herself evaluate the generated content; this is usually in conjunction with *interactive evolution*. This interaction can be explicit [34], [35] or implicit, as is the case of *Galactic Arms Race* [36] where the fitness of generated weapons is defined by how popular such a weapon is among the players. Another approach is to estimate user-preference through predictive models which might be trained on a large set of player interactions (more general) or on the interactions with a specific user (player-tailored content). The latter approach falls into the domain of experience-driven procedural content generation [10], where the content is specifically tailored to the user.

2.3 Computational Creativity

Computational creativity is an interdisciplinary field that explores problems such as: how can "creativity" be defined and quantified, whether it is possible for a machine to replace (or improve) human creativity, and if it is possible for an algorithm to create content that displays a human-like level of creativity. During the last few decades AI has become more and more successful at solving both abstract and concrete problems. One of the most recent famous examples is Google's AlphaGo, a program to play the game of Go, which has consistently beaten the best Go players in the world. What makes this an impressive feat is the sheer complexity of the game. As AI systems continue to become more and more competent at addressing tasks at a human-competitive level, it is still unclear if and how these systems can be creative.

As stated by Pasquier *et al.* [37], the field of computational creativity can be divided as the investigation of:

- *Creativity as It Is*. It is striving to understand and simulate human creativity. What is creativity? If we can understand this complex notion, can we simulate it? As such, when such a cognitive modelling approach is chosen, it is also part of cognitive science.
- *Creativity as It Could Be*. The field is also devoted to exploring processes of which we know humans alone to be incapable. The outcomes of those artificial processes might nonetheless be considered novel and valuable, that is creative.

To be able to explore the "creativity" of AI systems, it is important to define what "creativity" is. While no definition is universally accepted [38], philosopher Margaret Boden [39], from the idea that "being creative means to create something new", defines creativity as the ability to create artefacts with two key properties: originality and value. Boden proposed to distinguish between psychological and historical creativity (respectively P- and H-creativity):

- P-creativity refers to the creation of artefacts that are novel and valuable to the individual.
- H-creativity refers to the creation of artefacts that are novel and valuable for humanity.

Boden further describes three types of creativity that take in account the context in which creativity is applied: *exploratory*, *combinatorial* and *transformational*. These serve to distinguish between finding novelty within a pre-existing creative space (exploratory), combining creative spaces in new ways (combinatorial), and transforming the creative space (transformational). These concepts can be applied to any kind of creative domain, for example in music a creative space could be considered a specific music genre (e.g. Irish folk music) while in games it could be defined by a combination of core game mechanics (e.g. "platformers"). Bown [40] proposed a distinction between *generative* and *adaptive* creativity. The core of this distinction lies in distinguishing between processes that display creativity while lacking a specific actor that benefits from the artefact (e.g. evolution), and creativity as an intentional cognitive act. This definition is relevant to us because many human processes do not display any obvious value connected to the created artefact.

If we consider the creative process, Ritchie [41] observes that "the creativity of humans is normally judged by what they produce", while the creative processes themselves are not observable, and hence not reliable to assess creativity. Another argument he makes is that, by including assessment of the creative process itself, there is a risk of a circular argument: "the artefact is creative because it is the result

of a creative process because the result is creative". Ritchie proposes that the creative process itself should be considered a separate artefact: this way a method could be considered creative if it possesses novelty and value, even if (some or all of) the artefacts created are not.

Colton [42] expands on Ritchie's idea by stating how the artefact and the creative process should be equally considered. He makes the argument with his famous example of an art lover's encounter with two very similar paintings:

> *Imagine an art-lover at an exhibition entitled 'Dots 2008'. He speaks to two artists, each displaying a painting. In both cases, the art-lover cannot see past the seemingly random arrangement of dots of paint. He mentions this to the first artist, who says: "Oh, no, they're not randomly placed. Each dot represents a friend of mine. The colour of the dot represents how I feel about them, and the position indicates how close I am to them." The art-lover moves on to the next artist, and mentions again that the dots look like they have been randomly placed. The artist replies: "Yes, that's right – I just mixed a random colour and dabbed it onto the canvas" . Returning a week later with a friend, our art-lover wants to purchase the first painting, explaining to the friend that it represents feelings. Neither artist is present, and the art-lover cannot remember which painting is which. The friend points out that perhaps they could work out which one represents feelings, but they fail to see anything but randomness in both works. Finally, when the friend points out that both paintings look alike, so they should just choose one, our art-lover is inconsolable, and buys neither.*

Colton proposes a model to assess the behaviour of software, and whether it should be considered creative or not: the Creative Tripod. The three legs of the tripod represent three behaviours that Colton defines as important for creativity: skill, appreciation, and imagination. Moreover each leg is composed by three segments that represent the parties that could be contributing creatively to the artefact: the programmer, the algorithm, and the audience. He states how if we perceive the software as skilful, appreciative and imaginative (i.e. at least one segment is extended in each leg), then it should be considered "creative" regardless of the behaviour of other parties. It is important to note that Colton's framework for evaluating the creativeness of algorithmic systems is not the only one, in fact this is a topic of debate in this community, and many other measures of creativeness and frameworks have been proposed [41], [43]–[45].

2.4 Algorithmic Music

Procedural generation of music is a field that has received much attention in the last decade [46]. The approaches are diverse and range from creating simple sound effects, to avoiding repetition when playing human-authored music, to creating more complex harmonic and melodic structures [1], [47]–[51]. Wooller [52] divides approaches to procedural music generation into two categories, namely *transformational* and *generative* algorithms. Our music generator, METACOMPOSE, falls in the latter category as it creates music without having any predefined audio clips to modify or recombine.

Transformational algorithms act upon an already prepared structure, for example by having music recorded in layers that can be added or removed at a specific time to change the feel of the music. Historical examples of this approach can be found in 18C musical dice games, Henry Cowell's *String Quartet no. 3* (1935), or Stockhauesen's *Klavierstuck XI* (1956). Note that this is only an example and there are a great number of transformational approaches [53], [54], but a complete study of these is beyond the scope of this thesis.

Generative algorithms instead create the musical structure themselves, which leads to a higher degree of complexity in having the music remain consistent, especially when wanting to connect the music to game events. Such an approach requires more computing power, as the musical content has to be created dynamically and on the fly. An example of this can be found in the game *Spore*: the music generators were created by Brian Eno with the *Pure Data* programming language [55], in the form of many small samples that created the soundtrack in real-time.

In the projects described in this thesis, we adopt the latter approach, in particular focusing on generative procedural music generation in games for emotional expression. While the topics of affect [7], semiotics [8] and mood-tagging [9] are also interesting and significant, the focus of the system is *real-time generation of background music able to express moods*.

Many projects focus on expressing one (or more) affective states: an example is described by Robertson [56], where a music generator is developed to focus on expressing fear. There are parallels between this work and the approach presented here, for example musical data is represented via an abstraction (in their case via the CHARM representation [57], [58]), yet we claim METACOMPOSE has a higher affective expressiveness since it aims to express multiple moods in music. A more extensive example of a generative music system targeted at expressing particular emotions is described by Monteith [59] using Markov models, n-grams and statistical distributions from a training corpus of music. Chan and Ventura's work [60], much like this research, focuses on expressing moods; yet their approach relies on

changing the harmonization of a predefined melody, while mine generates the complete piece.

There are many examples of evolutionary algorithmic approaches to generating music, two notable examples are the methods to evolve piano pieces by Loughran *et al.* [61] and Dahlstedt [62], although many more can be found in the *Evolutionary Computer Music* book [63]. Other examples of real-time music generation can be found in patents: two examples are a system that allows the user to play a solo over some generative music [64] and another that creates complete concerts in real-time [65]. An interesting parallel between the second system [65] and mine is the incorporation of a measure of "distance" between music clips in order to reduce repetition. Still, neither of the patented systems present explicit affective expression techniques.

As already suggested, a wide array of techniques can be used in algorithmic music. Here we describe the categories which most approaches can be divided into, providing examples of methods and references to relevant research.

2.4.1 Evolutionary methods

As previously discussed in Section 2.1, genetic algorithms have been shown to be effective search methods, especially when it is hard to define the search space, and when this search space is very large. Another reason why these methods often find use in creative applications is that they can provide multiple solutions, which in the music domain is an added bonus. These approaches have many different variations, but in the music domain they usually follow two main categories:

Objective fitness function (automatic): all GAs for which the chromosomes are evaluated through a formally stated fitness function, fall into this category. This function is usually based on some music theory or some computable characteristics of the generated artefact. These kind of algorithms have been used for a variety of purposes, such as thematic bridging between simple melodies [66] or harmonization of a melody [13], [67]. A famous example of these techniques can be found in the work by Papadopoulos and Wiggins [68], which used a symbolic GA to evolve jazz melodies based on a chord progression. They achieved this by introducing specialized genetic operators and a fitness function that evaluates eight characteristics of the melody. As can be seen from this example, the success of this kind of approach is generally dependent on the amount of knowledge that the system possesses. In this thesis we present two evolutionary algorithms for music generation: one which has

been given domain knowledge (the melody generation in METACOMPOSE), and one that explores what kind of emergent structures can arise with as little musical knowledge as possible (PrimalImprov). Wiggins remarks that the use of GAs is not ideal to simulate human musical thought, as "their operation in no way simulates human behaviour" [69]. Yet, when the purpose is not explicitly to recreate how humans create music, the use of GAs has seen much success.

User as fitness function (interactive): in this case, contrary to the previous case, there is no objective evaluation function. Instead a human actor is tasked with judging the artefacts created by the system and, through this, guide the generative process. This approach has been used to compose entire pieces of music [70], rhythmic patterns [71], and melodies [72]. One of the most famous examples is Biles' *GenJam* system, which evolves "a novice jazz musician learning to improvise". Another interesting approach is Hoover's MaestroGenesis system [73], which creates multi-part accompaniment to a given melody using the Functional Scaffolding for Musical Composition (FSMC) framework [74].

The drawbacks of this approach are:

- Subjectivity: the result is strongly influenced by the user, meaning artefacts can be of widely variable quality. Moreover, as the evaluation process happens through the user's mental processes, there is no way to analyse how it developed.
- Efficiency: maybe the biggest drawback of the interactive approach, the user has to listen to all potential solutions before being able to evaluate a population. This becomes very quickly infeasible, both as the amount of time the user has to listen to artefacts quickly grows and as fatigue renders the user less involved in the evaluation task.

2.4.2 Mathematical models

Mathematical models were among the first techniques used to generate algorithmic music, mainly Markov chains [75] and stochastic processes [76]. Interestingly many commercial programs have used –and still use– stochastic processes (e.g. Jam Factory and M [77]), although lately we can see the appearance of software based on deep-learning[2]. A famous example of the usage of these techniques is Cybernetic Composer [78], a system that composes pieces in different genres, such as jazz, rock, and ragtime. With a very different

[2] melodrive.com

approach we can also find systems based on chaotic non-linear systems [79], [80]. The issue with these systems is evaluating the quality of the produced music, as they do not use knowledge of "human music". In a way machine learning approaches can be included in this category as well, as they create mathematical models based on a corpus, but we will expand on them in Section 2.4.5. The main difficulties of this approach are that these models have to be extrapolated by analysing many pieces (e.g. defining the weights of a Markov chain), and that it is very complicated to manage deviations from the norm and how to properly incorporate them in the music.

2.4.3 Grammars

A grammar is the set of structural rules that govern the composition of words, phrases, and clauses. As music is a structured collection of pitches, the idea of a grammar of music has been very popular [81]. These grammars could be defined a-priori, extrapolated by some corpus, or be generative (e.g. L-systems [82]). In formal language theory, a grammar G is defined as a tuple $G = (V, \omega, P)$, where

- V is a set of symbols – an alphabet – containing both replaceable elements (variables) and elements that cannot be replaces (terminals).
- ω is a collection of symbols belonging to V, which defines the initial state of the system. This is generally referred to as *axiom* or *initiator*.
- P is a set of *production rules*. These define how to form strings from the alphabet V that are valid. These rules define how variables can be substituted by combinations of constants and other variables. As such, each rule that belongs to P has two components: a predecessor (the variable to substitute) and the successor (the new string to substitute to the variable).

Cope [83], in his seminal work Experiments in Musical Intelligence (EMI), explores the understanding of specific musical styles and the stylistic replication of various composers. EMI takes in input two or more works, from which it extracts what Cope calls "signatures" using pattern matching. These signatures then become the foundation of a grammar, and meaningful arrangements of these are performed using an augmented transition network.

A shortcoming of grammar techniques are that, in general, musical grammar implementations do not make claims about the semantics of the generated pieces. Another issue is that, while grammars are hierarchical structures, music is not necessarily so (take improvisation, for example). Roads [84] suggests that adding ambiguity might add to the representational power of grammars, yet that would introduce a new set of challenges.

2.4.4 Knowledge based systems

As we discussed in the previous sections, most of the approaches to generate music contain some kind of knowledge, so they could be considered knowledge based systems. In this section we will provide some examples of systems which are symbolic, or use some explicit rules. One of the main differences compared to the previously described approaches is that these systems have explicit reasoning, so it is possible to understand the choices they make.

Ebcioglu's CHORAL [85], a rule-based expert system for harmonization of chorales in the style of J.S. Bach, is one of the most famous examples of this approach. Other examples of systems for harmonization have been developed by Tsang and Aitken [86] through the use of constraint logic programming, and by Pachet and Roy [87] through the use of constraint satisfaction techniques. For a more complete survey of harmonization thorough constraints, see Pachet's survey [88].

Pachet [89] described the notion of potential actions: in order to emulate creativity, instead of using randomness, he proposed a set of potential actions as an initial state (reference) for the musical problem. Ramalho and Ganascia [90] implement this notion to create playable improvisations. Robertson *et al.* [56] developed an interesting system for generating atmospheric music for an educational virtual environment by adapting music techniques used in films. The system took in a tension curve that would be reflected in the music.

Knowledge based systems also have some shortcomings, the most important one being that it is difficult (and possibly very time-consuming) to define the knowledge that the system needs to use. Moreover this knowledge depends on the "expert" being able to properly describe the concepts that the system is going to use. Finally, in a domain such as music, there are many "exceptions to the rules" and coding them is probably infeasible, and even if it was possible it would greatly increase the complexity of the system.

2.4.5 Learning systems

The systems in this category do not have a-priori knowledge, but learn such knowledge (in general from a specific corpus). Two main approaches emerge according to the way these system store this information: subsymbolic representation (e.g. Artificial Neural Networks, see Section 3.4 for more details) and symbolic. There is a great number of applications of ANNs to the music domain, such as melody generation [91], [92], harmonization [93], and jazz improvisation [94]. Symbolic approaches are less common, Cope's EMI [83] can also fit this category, as the grammar used in the generative process is learned from a corpus. Another example is Schwanauer's MUSE [95] system, which used five different learning techniques to learn different harmonization tasks, both simple and complex.

A relatively new trend is to use Deep Learning [96] (DL), a type of machine learning method based on learning data representations using ANNs that contain more than one hidden layer. This technique, originally used for classification purposes, can also be used in a generative context. The underlying assumption of DL is that the ANNs layers correspond to different levels of abstraction of the specific domain (e.g. one layer might learn relationships between adjacent notes, while higher levels would learn relationships between phrases). Google's Magenta[3] project focuses on generative art and music using exactly such techniques. Another notable example of Deep Learning applied to music is Hadjeres and Pachet's *Deep Bach* [23], which trained an ANN to create harmonisations in J.S.Bach's style. While these kind of algorithms can create some very convincing music, an open issue is how to give better semantics to the generated pieces. Another issue is that these ANNs are extremely opaque, making it very hard to observe why choices were taken.

2.4.6 Hybrid systems

Hybrid systems, as the name suggests, use a combination of different AI techniques, in an attempt to mitigate the weaknesses of specific algorithms through the use of complementary ones. The METACOMPOSE system described in this thesis would fall in this category, as it presents both evolutionary and stochastic methods.

A common combination is the use of ANNs (that have been trained on some corpus) as evaluation functions for an evolutionary algorithm, examples can be found in the rhythm generation system by Burton and Vladimora [97], and in the "one measure response to one measure call" system by Spector and Alpern [98]. A different hybrid system is Hild and Feulner's HARMONET [99], [100], which harmonizes melodies using a combination of ANNs and constraint satisfaction techniques. Hybrid systems can usually become very complicated, leading to a higher complexity in implementation, verification and validation.

2.4.7 Live algorithms

There are many applications of computers and algorithms to music that use the *computer-as-composer* paradigm. In this section we briefly discuss some of the other paradigms to better situate our approach. *Live coding* is a practice that has become more and more common among computer musicians where algorithms produce music on the fly, while the musician can manipulate these in real-time. We can define such use as *computer-as-instrument*. In this approach the computer is just a tool that relies on human agency.

[3] https://magenta.tensorflow.org/

Blackwell *et al.* [101] proposed the concept of *Live algorithms for music* (LAMs) as a way of analysing live music performance systems that, to some extent, exhibit autonomy or agency. NN Music is a system that follows this concept: it is a performer-machine system for Max/MSP[4] that trains a feed-forward neural network mapped to stochastic processes for musical outputs [102]. Another example, which also uses Genetic Algorithms (GAs), is the system described by Bown and Lexter for generative and interactive musical performance [103].

2.5 Musical Metacreation

Musical Metacreation (MuMe) is a subfield of computational creativity. Problems addressed by MuMe can be very different, while not including all of them Pasquier *et al.* [37] defined some "canonical" problems that encompass most systems:

- *Composition*. The system must produce novel music compositions, either through algorithmic means, or using knowledge derived from a corpus of existing compositions. The output can be any symbolic representation, such as a score in MIDI format or other notation, of a pattern, part, piece, or group of musical pieces;
- *Interpretation*. Given a composition (in any format), the interpretation consists of producing an audio rendering of the composition. A typical example is a musician or an orchestra performing from a score;
- *Improvisation*. The creation and real-time performance of new musical material spontaneously, to varying degrees, and often enacted in groups. While it can be prepared and rehearsed, it is assumed that improvisation occurs in real-time.
- *Accompaniment*. One or several musical parts (e.g., melody, harmonic progression, rhythmic accompaniment) are provided, and other parts need to be either composed or interpreted (or both).

Given a problem there are many dimensions that are commonly used to categorize MuMe systems:

Autonomy: this dimension can be considered a spectrum that ranges between systems that are completely autonomous and ones with

[4] http://www.cycling74.com

no autonomy. The latter encompass software that is dependent on user input, these could sometimes be considered creativity enhancers that help the user in some task. Completely autonomous systems do not require human input: the user starts the system and the system executes the musical task it is designed to complete (a famous example is Cope's EMI [83]). The main system described in this thesis – METACOMPOSE – lies at this end of the spectrum but, while being autonomous, it is also interactive. Bradshaw *et al.* [104] define the autonomy of a system by two dimensions: self-sufficiency and self-directedness.

Origin of the System's "Knowledge": this dimension distinguishes between systems that require exposure to musical information as input, and corpus-based systems that have been exposed to music (e.g. training a neural network on a corpus). METACOMPOSE does not require musical input and, while it does not use a corpus of music, it can be considered on the side of the spectrum that has been exposed to musical information. In fact it has much musical information stored in the code more or less explicitly (see chapter 5).

Inputs: this dimension distinguishes between what kind of input is fed to the system. These are generally *symbolic* (musical notation, MIDI files, etc.), *audio signals*, or *hybrid* (a combination of the former two).

Online/Offline: this dimension distinguishes between systems that perform in real-time (online) – for example reacting to live input – and systems where the generation occurs either faster or slower than real-time. In this latter case the artefact of the system is generally non-interactive and can be accessible once the generation process is over. METACOMPOSE lies more on the online side of the spectrum, as it is designed to generate music in real-time, although it could also be used offline.

Generality: this dimension considers how specific/general a system is. Often MuMe systems are created with a specific aesthetic in mind, which leads more to variations of the same composition rather than wildly different musical pieces. An example of a very generic system is the one described by Conklin [105] that, given a corpus in input, attempts to abstract the musical knowledge present in the corpus. This gives the system the potential to create music related to any possible corpus. The METACOMPOSE system, falls closer to the specific end of the spectrum, although it has the potential to be made more generic by changing the implementation of the improvisational modules.

The Methods used by the system: as stated before there is a multitude of applications of MuMe, consequently there is a veritable

cornucopia of methods that can be applied to MuMe (see [106], [107]).

2.6 Emotions and moods

Emotions have been extensively studied within psychology, although their nature (and what constitutes the basic set of emotions) varies widely. Numerous models of emotion have been developed since the seminal studies of the early 20th century [108], [109], arguably one of the most influential is the theory of basic or discrete emotions devised by Ekman [110]–[112]. The theory of basic emotions posits that all affective experiences derive from a core set of basic emotions which are distinct and independent. An alternate approach to the study of emotions has been the development of dimensional models of affect, which posit that all emotions derive from a combination of two or more underlying psychological "dimensions" [113]–[115]. Lazarus argues that "emotion is often associated and considered reciprocally influential with mood, temperament, personality, disposition, and motivation" [116]. Therefore the approach presented in this work aims to produce scores with an identifiable mood, and in so doing, induce an emotional response from the listener.

Affect is generally considered to be the experience of feeling or emotion. Brewin states that affect is post-cognitive [117]; namely emotion arises only after an amount of cognitive processing has been accomplished. With this assumption in mind, every affective reaction (e.g., pleasure, displeasure, liking, disliking) results from "a prior cognitive process that makes a variety of content discriminations and identifies features, examines them to find value, and weighs them according to their contributions" [117]. Another view is that affect can be both pre- and post-cognitive, notably [118]; in this case, thoughts are created by an initial emotional response that then leads to an induced affect.

Mood is an affective state. However, while an emotion generally has a specific object of focus, mood tends to be more unfocused and diffuse [119]. Batson posits that mood "involves tone and intensity and a structured set of beliefs about general expectations of a future experience of pleasure or pain, or of positive or negative affect in the future"[120]. Another important difference between emotions and moods is that moods, being diffuse and unfocused, may last longer [121].

In this paper, we focus on mood instead of emotion, for we expect that in games – where the player listens to the background music for a longer time – moods are more likely to be remembered by the players after their game-play. In addition, they are easier for game designers to integrate, since they represent longer-duration sentiments, more suited to segments of game-play.

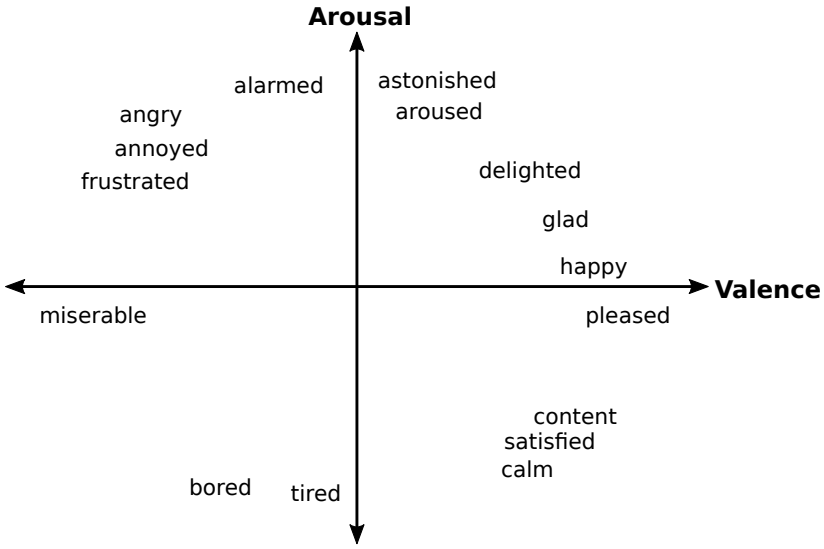

FIGURE 2.3: The Valence-Arousal space, labelled by Russell's [12] direct circular projection of adjectives.

2.7 Moods in music

There has been much discussion about whether music can actually express anything, two quotes by Stravinsky and Mendelssohn are particularly representative:

> "Music is, by its very nature, essentially powerless to express anything at all, whether a feeling, an attitude of mind, a psychological mood, a phenomenon of nature.... "(Stravinsky, 1936)

> "People usually complain that music is so ambiguous; that it is so doubtful what they ought to think when they hear it; whereas everyone understands words. With me it is entirely the converse.... The thoughts which are expressed to me by a piece of music which I love are not too indefinite to be put in words, but on the contrary too definite. "(Mendelssohn, 1842)

Psychologists and philosophers – such as Hevner [122], Meyer [123], and Clynes [124], [125] – have been arguing about the topic as well, but a consensus is still missing. The interested reader can find an excellent (if not so recent) review of these philosophical theories by Budd [126].

The set of adjectives that describe music mood and an emotional response to it is immense and there is no accepted standard vocabulary. For example, in the work of Katayose [127], the emotional adjective set includes *Gloomy, Serious, Pathetic* and *Urbane*.

Russell [12] proposed a model of affect based on two bipolar dimensions: *pleasant-unpleasant* and *arousal-sleepy*, theorizing that each affect word can be mapped into a bi-dimensional space (Figure 2.3).

Thayer [128] applied Russell's model to music using the dimensions of *valence* and *stress*; although the names of the dimensions are different from Russell's, their meaning is identical. Also, we find different terms among different authors [114], [129] for the same moods. We use the terms *valence* and *arousal*, as they are the most commonly used in affective computing research. In this way, affect in music can be divided into quadrants based on the dimensions of valence and arousal: *Anxious/Frantic* (Low Valence, High Arousal), *Depression* (Low Valence, Low Arousal), *Contentment* (High Valence, Low Arousal) and *Exuberance* (High Valence, High Arousal). These quadrants have the advantage of being explicit and discriminate; also they are the basic music-induced emotions described in [130], [131].

Researchers have put much effort in trying to understand the relationships between various music features and their affect. These studies cover features such as: tempo [132], [133], rhythm [134], mode [122], harmony [134], [135], texture [132], and volume [136]. One of the characteristics of these studies is that, most of the time, they only explore the participants' reactions to one single variable. Studies that explore the interplay of more variables are more rare; some notable examples are Scherer and Oshinsky's one [137] (where tempo, mode, and pitch were varied), and Alpert and Alpert's one [138] (where tempo, rhythm, harmony, and dynamics were varied). The METACOMPOSE system could be used to vary many variables in multiple combinations, making it a good candidate to perform such multi-variable experiments. Moreover, the automated procedure for creating music would significantly decrease the time necessary to set up such experiments.

It is unsurprising that music, assuming it does have the power to express (and possibly elicit) moods, has come under the attention of of marketing researchers [139]. Interestingly even in earlier work on this subfield an interest in algorithmic composition emerged, as the objective had more to do with creating affective music than artistically interesting music. Already in 1982, Clynes and Nettheim imagined that soon non-musicians would be able to compose "free from the considerations of musical notation and yet no without guiding order" [125].

2.8 Affective Player Modelling

Yannakakis *et al.* define player modelling as "the study of computational models of players in games"[140]. The main objective of player modelling is being able to make predictions, descriptions, or interpretations on the behaviour of the player given the available observations. These observations could be given by the player's interaction with the game (e.g. score, heatmaps, etc.), intrinsic game information (e.g. level difficulty), physiological responses (e.g. heart-rate sensor, skin conductance sensor), demographic information about the player

(e.g. age, gender). The purpose of these models can be very different, and as such the possible outputs of the system depend largely on the objective. Examples might be predicting how the player will behave in the future, classification of the player, and – the most interesting case for us – predicting the player's affective state.

There are two main methods of creating such a model: the top-down and the bottom-up methodologies. The top-down methodology builds the model based on a specific theoretical framework; these models are usually used to test these theories and compare them with the observations collected to evaluate their correctness. In this thesis we present a model of mood expression in music that follows this structure and we present multiple evaluations to show where it performs properly and where it does not.

The bottom-up methodology instead builds a model directly from a set of observations (model-less) without a-priori assumptions. These models are generally built using machine learning techniques, data mining (e.g. regression, in case both input and outputs are known), preference learning (when the user itself specifies a preference between two or more options), and classification (which can be used even in case there is no known output).

Modelling affect is still an open challenge [141]. Nonetheless, several studies have explored the interplay between measures of affect and physiology, providing some insight in the affective state of the user [142]. The most common measures are:

- Heart rate: the number of heart beats per minute. This measure is arousal dependent, where an increase in stress leads to an increase of heart rate. It's important to note that when related to music, increase in heart rate is limited regardless of arousal expression.

- Skin conductance: this measure is arousal dependent, where high arousal leads to a decrease in electrical resistance and an increase in skin conductance [143].

- Facial electromyography: patterns in facial expression appear when the user imagines happy, sad, and angry situations [144]. These subtle patterns, which are invisible to the human eye, can be observed through facial electromyography. This is a useful method to measure the valence of the perceived mood by the user.

- Facial action coding system (FACS): Ekman [145] defined this method of tracking zygomatic and corrugator facial muscles. He has shown in many studies how some of these muscles seem to have unconscious reaction to the internal state of the user [110], [111]. This measure uses Ekman's universal basic emotions theory.

- Head nod/shakes: measuring the amount of head nod/shakes can give an indication of how positive/negative the user's affective state is. Interestingly it seems that overt head movement (i.e. the physical act of performing the movement) can also influence the formation of preference of an artefact [146].

It is important to note that a major problem of most of these measures is that they are perceived as invasive by the user, possibly affecting the game/musical experience. Kivikangas *et al.* also provide a review of these physiological methods in the context of game research [147].

2.9 Music Generation and Games

The main distinction between creating music for games, compared for example to film music, is the non-linear nature of games. In games, the narrative is driven by the player creating two main problems: what music should be played when the player is in a particular state/level, and how should we transition from one piece to another. While we do not deal with the latter problem in this thesis, the next paragraph will discuss what challenges it presents.

Paul Hoffert [148] discusses what musical elements should be taken into consideration while composing a transition: volume, tempo, rhythm, key, harmony, texture, and style. While this already seems a huge set of (complex) variables to account for, in games music may transition at any time, potentially requiring an infinite number of cues. In the 8-bit and 16-bit era, the most common solution was to use direct hard cutting between different pieces. While this is a very simple approach, the result can be very jarring and immersion-breaking for the player. As already mentioned in the Introduction chapter, the most common approaches nowadays are to either use a fast fade-out and then begin the new cue, or to cross-fade between two pieces. *The Legend of Zelda: Ocarina of Time* (Nintendo, 1998) uses this technique quite effectively. While this approach is indubitably better than the hard-cut one, it can still lead to a feeling of abruptness, depending on the speed of the fades and the pieces themselves. Another common expedient to mask transitions is the use of a *stinger* (a quick sforzando chord, or an abrupt sound effect). This technique is particularly effective in transitions that include some violent element (e.g. entering combat).

There are many ways to introduce variability in game music, for example by changing: pitch, tempo, meter, volume dynamics, timbres, mixing, etc.. Algorithmic music is surprisingly rarely used in the industry, with the most famous examples being LucasArts' *Ballblazer* (1984), Maxis' *Spore* (2008), and Millennium Interactive's *Creatures* (1996). The former employed what composer Peter Langstron [149] defined as "riffology", an algorithm in which choices are made using

some dynamically adjusted weights. Examples of these choices are: which melody to play next (from an archive of 32 melodies), what tempo to use, what volume, etc..

The games that present more interesting approaches in their music use some sort of "open form": the structure of the pieces of music present openness on some level. The idea of "open form" can be traced back to at least the eighteenth century Europe, where we have records of musical dice games, and other forms of aleatory music games [150]. Again, *The Legend of Zelda: Ocarina of Time* presents a variable sequencing structure in the *Hyrule fields* section of the game. This is a very large area that acts as a "hub", connecting all other areas, and as the player is going to spend a large amount of time exploring it, special attention was given to its background music. The music was divided in many sequences, which were played randomly to maintain interest. This approach, while allowing for high quality music and variation, presents several difficulties for the composer. The main one being that the majority of Western music (so what the average Western listener is used to) has a goal-oriented and linear structure. As Kramer [151] underlines "Linear music exists in a time world that admits, and even depends on, continuity of motion." On the contrary open form music can have problems creating such motion, and connecting such motion to the player's responses makes the composition task even more complicated. If no dramatic curve is achieved it is very likely that the music will become "ambient", losing its communicative functions.

Still related to the open form approach, there is the transition matrix one: a transition matrix contains a set of small musical pieces that allows the computer to decide how to transition from one piece to another. This means that the composer has to anticipate all possible transitions, where in the piece such transitions can happen, and compose the transition pieces. As you can imagine this can become very quickly an herculean task, for example using this approach Bernstein [152] had to compose 220 segments for the game *Multiplayer Battletech* (Kesmai, 1996). We should also note that changes in music can not only be based on *interactive* events (e.g. changing level, or interacting with an object), but also on *adaptiveness* (e.g. player health, ammunition and so on). This means that, in addition to the many fragments to achieve an *interactive* soundtrack, many more might be needed to make it *adaptive* as well. An interesting approach can be found in *Anarchy Online* (Funcom, 2001), where a software tool was designed to combine many small sequences in a continuous (potentially very long) single track. 750 sequences were composed, each organized according to its relation to the other tracks and to the game [153].

From the research side, it is therefore important to mention the work by Livingstone [9], which defines a dynamic music environment where music tracks adjust in real-time to the emotions of the

game character (or game state). While this work is interesting, it is limited by the usage of predefined music tracks for affective expression. Finally, another notable project in affective expressive music in games is *Mezzo* [154]: a system that composes neo-Romantic game soundtracks in real time and creates music that adapts to emotional states of the character, mainly through the manipulation of *leitmotifs*.

Chapter 3

Algorithms

While the previous chapter focused on theoretical concepts and contextualization of the research presented in this thesis, this chapter presents a review of the techniques used throughout this research. In this chapter we also provide references to relevant research that uses such techniques in a similar context, and a brief discussion of how this approach differs from what is described in this thesis.

3.1 Non-dominated Sorting Genetic Algorithm

This section describes the Non-dominated Sorting Genetic Algorithm II (NSGA-II) described by Kalyanmov Deb [26]. This multiobjective evolutionary algorithm is one of the most efficient and although there has been a newer iteration (NSGA-III [155]), that is mostly relevant for many-objective optimization. The main characteristics of this algorithm are:

- A sorting non-dominated procedure in which all individuals are sorted according to non-domination. We remind the reader that an individual is dominated when all of its objective scores are worse than the second individual. The procedure consists of finding all non-dominated individuals, these will become part of the current first-order Pareto front. Then these individuals are removed from the population and the process is repeated to obtain a second-order Pareto front. The process is continuously repeated until no individual is left in the population, and a sorted list of non-dominated genomes is obtained.
- It features an elitism mechanism that stores all non-dominated solutions, assuring their survival to the next generation. This improves convergence speed.
- It includes a crowding distance measure, which helps guarantee diversity and spread of solutions on the Pareto front.

Additionally it can also include constraint handling through a modified concept of dominance: if a solution violates a constraint it is dominated by any other solution that satisfies all constraints.

Algorithm 2 NSGA-II

1: **BEGIN**
2: *INITIALISE population* with random candidate solutions;
3: **while** *TERMINATION CONDITION* is satisfied **do**
4: *EVALUATE AGAINST OBJECTIVES* the offspring;
5: *FAST NON-DOMINATED SORT* the population, creating non-domination ranks;
6: *SELECT* parents according to non-domination ranks;
7: *RECOMBINE* pairs of parents;
8: *MUTATE* the resulting offspring;
9: *EVALUATE AGAINST OBJECTIVES* offspring;
10: *SELECT* individuals for the next generation;
11: *FAST NON-DOMINATED SORT* the population, creating non-domination ranks;
12: *CROWDING DISTANCE ASSIGNMENT* for each of the fronts (composed by individuals in the same rank);
13: *SELECT* best individuals based on rank and crowding to become the next generation;
14: **end while**
15: **END**

3.2 Feasible/Infeasible 2-Population Genetic Algorithm

Many search/optimization problems have not only one or several numerical objectives, but also a number of constraints – binary conditions that need to be satisfied for a solution to be valid. The approach we adopt for melody generation contains such constraints, these are described in detail in Section 5.1.2. A number of constraint handling techniques have been developed to deal with such cases within evolutionary algorithms. In this category of problems, the solutions space is divided into feasible areas (solutions that satisfy all constraints) and infeasible areas (solutions that do not satisfy one or more constraints). As such, algorithms that try to approach these problems should be able to ensure that feasible content will be found, and that too much computational time shouldn't be used to explore infeasible solutions.

The Feasible/Infeasible 2-Population method (FI-2POP) [24] is a constrained evolutionary algorithm that keeps two populations evolving in parallel, where feasible solutions are selected and bred to improve their objective function values, while infeasible solutions are selected and bred to reduce their constraint violations (see Figure 3.1). In each generation, individuals are tested for constraint violations; if they present at least one, they are moved to the 'Infeasible' population, otherwise they are moved to the 'Feasible' population. This means that as evolution starts, most (if not all) possible solutions

will be infeasible, and during the subsequent generations more and more feasible solutions will be found. An interesting feature of this algorithm is that the infeasible population influences, and sometimes dominates, the genetic material of the optimal solution. Since the infeasible population is not evaluated by the objective function, it does not become fixed in a sub-optimal solution, but rather is free to explore boundary regions, where an optimum solution is most likely to be found. Sorenson and Pasquier have applied this method for the generation of platformer levels and adventure games, while Liapis used it in his Sentient Sketchbook [156] tool to support a designer in the creation of game levels, and to generate spaceship sprites [157], [158].

Algorithm 3 FI-2pop genetic algorithm

BEGIN
INITIALISE population with random candidate solutions;
CHECK FOR CONSTRAINT VIOLATION each candidate;
DIVIDE the population into Feasible population F and Infeasible population I;
while *TERMINATION CONDITION* is satisfied **do**
    for each Population P in $[F, I]$ **do**
        EVALUATE each candidate in P; ▷ this will be checking for constraint satisfaction in the Infeasible population, and fitness evaluation for the Feasible one
        SELECT parents;
        RECOMBINE pairs of parents;
        MUTATE the resulting offspring;
        EVALUATE new candidates;
        SELECT individuals for the next generation;
    end for
    CHECK FOR CONSTRAINT VIOLATION each candidate in I;
    CHECK FOR CONSTRAINT VIOLATION each candidate in P;
    MOVE each infeasible candidate found in P to I;
    MOVE each feasible candidate found in I to P;
    for each Population P in $[F, I]$ **do**
        EVALUATE new candidates;
        SELECT individuals for the next generation;
    end for
end while
END

3.3 Non-dominated Sorting Feasible-Infeasible 2 Populations genetic algorithm

When dealing with constrained optimization problems, the approach is usually to introduce penalty functions to act as constraints. Such an approach favours feasible solutions over the infeasible ones, potentially removing infeasible individuals that might lead to an optimal solution, and finding solutions that can be considered local optimum. There have been many examples of constrained multi-objective optimization algorithms [159]–[162].

The internals of METACOMPOSE use a combination of FI-2POP and NSGA-II dubbed Non-dominated Sorting Feasible-Infeasible 2 Populations (NSFI-2POP), uniting the benefits of maintaining an infeasible population, which is free to explore the solution space without being dominated by the objective fitness function(s), and finding the Pareto optimal solution for multiple objectives. The algorithm takes the structure of FI-2POP, but the objective function of the feasible function is substituted with the NSGA-II algorithm. In Section 5.1.2 an application of this approach to the evolution of melodies is described.

Some of our students have conducted an analysis of this algorithm[1] compared to: NSGA-II with constraint handling [26], Jimenez and Verdegay's nondominated sorting evolutionary algorithm [163] (NSEA), and Ray's algorithm [164]. They tested the algorithms on some of the constrained multi-objective problems defined by Deb *et al.* [159] and found out that NSFI-2POP outperforms all other algorithms apart from NSGA-II. Although the performance of NSGA-II is comparable with NSFI-2POP's one, they show that NSFI-2POP achieves a better coverage of the Pareto front, which confirms the hypothesis that using two separate populations can lead to a better exploration of the solution space.

3.4 Artificial Neural Networks

Artificial Neural Networks (ANNs) are mathematical models inspired by biological neural networks. They have many uses, but the primary ones are approximation of functions and pattern recognition. They were first described by McCulloch and Pitts [165], and other seminal research on the topic performed by Kohonen [166], Werbos [167], and Hopfield [168].

ANNs are composed neurons (or nodes), these are divided in three categories:

[1] Kasper Hjort Berthelsen, Jon Voigt Tøttrup, Sebastian Baunsgaard, Sebastian Benjamin Wrede, *Evaluation of multi-objective genetic algorithms*, Bachelor thesis, IT University of Copenhagen.

Algorithm 4 NSFI-2POP genetic algorithm

1: **BEGIN**
2: *INITIALISE population* with random candidate solutions;
3: *CHECK FOR CONSTRAINT VIOLATION* each candidate;
4: *DIVIDE* the population into Feasible population F and Infeasible population I;
5: **while** *TERMINATION CONDITION* is satisfied **do**
6: *EVALUATE* each candidate in I according to constraint satisfaction;
7: *EVALUATE* each candidate in F using NSGA-II;
8: *SELECT* parents in I;
9: *RECOMBINE* pairs of parents in I;
10: *MUTATE* the resulting offspring in I;
11: *SELECT* parents in F;
12: *RECOMBINE* pairs of parents in F;
13: *MUTATE* the resulting offspring in F;
14: *CHECK FOR CONSTRAINT VIOLATION* each candidate in I;
15: *CHECK FOR CONSTRAINT VIOLATION* each candidate in P;
16: *MOVE* each infeasible candidate found in P to I;
17: *MOVE* each feasible candidate found in I to P;
18: *EVALUATE* new candidates in I;
19: *EVALUATE* new candidates in F;
20: *SELECT* individuals for the next generation in I;
21: *SELECT* individuals for the next generation in F;
22: **end while**
23: **END**

- Input nodes: these represent the different inputs to the network, they do not do any computation and just pass the values they receive to the other neurons connected to them.
- Output nodes: these represent the outputs of the network, they perform a weighted sum of the values passed by the connected neurons of the previous layer and return the result.
- Hidden nodes: these are all the nodes that contribute to calculating values between the input and output nodes. Each of them performs a weighted sum of the values in input (to the node), then the result is fed into an activation function, which determines the output.

These neurons are usually divided in layers, where each layer between the input and output one is referred to as a *hidden layer*.

One of the main capabilities of ANNs is the ability to approximate and learn functions. Supervised learning can "train" a network by feeding inputs to it, calculating the error from the expected output, and back propagate the error through each connection. Through this process it is possible to adjust the weights and alter the threshold parameters of the activation functions. This process does not alter the topology of the network, but in section 3.5 we will describe the NEAT algorithm which is an example of neuroevolution of topologies, as well as connection weights and activation functions.

3.4.1 Compositional pattern-producing networks

Compositional pattern-producing networks (CPPN) [169] differ from typical artificial neural networks in their set of activation functions and how they are applied. While ANNs often contain only one type of function (i.e. sigmoid and sometimes Gaussian), CPPNs can potentially include any type of activation function. The choice of functions for the canonical set can be biased toward specific types of patterns and regularities. For example, periodic functions – such as sine – produce segmented patterns with repetitions, while symmetric functions, such as Gaussian, produce symmetric patterns. Linear functions can be employed to produce linear or fractal-like patterns. Thus, the architect of a CPPN-based genetic art system can bias the types of patterns it generates by deciding the set of canonical functions to include. These kind of ANNs, combined with NEAT, have been used in various research in creating art, including music [170], [171], images [35], [172], 3D objects [173], and more [174], [175].

3.5 NeuroEvolution of Augmenting Topologies

Usually in ANN applications the network topology is defined a-priori and then the connection weights are derived either through evolutionary computation or machine learning techniques. This means that often the programmer has to run the learning experiment multiple times to find the ANN architecture that provides best results; this process requires significant knowledge and experience with ANNs, and still requires a good dose of guess-work.

The NeuroEvolution of Augmenting Topologies (NEAT) genetic algorithm developed by Ken Stanley [176] presents the novelty of evolving both the weighting parameters and structures of networks. In this way, it attempts to find a balance between the fitness of evolved solutions and their diversity. Three main techniques are crucial to this algorithm:

- tracking genes with history markers to allow crossover among topologies
- applying speciation to preserve innovations
- developing topologies incrementally from simple initial structures

NEAT begins the evolution with a uniform population of the simplest possible ANNs (without any hidden layer) and random connection weights. During the evolutionary process, hidden nodes and new connections can appear. As stated before each new feature of the network possesses an historical marker; this marker is used to allow that even very different genomes can still be compatible when applying operators such as crossover. The genetic diversity is preserved through speciation; in this way each individual competes mainly with other similar individuals. This allows for species to optimize their topology and connectivity without being overwhelmed by other species that might have a more complex structure, but not be as optimal.

NEAT is usually used as an offline algorithm; an example of the implementation of an online and decentralized version of NEAT is odNEAT, a system for the evolution of multi-robot systems [177]. *odNEAT* is designed to be executed onboard robots themselves during task execution to continuously optimize the parameters and the topology of the artificial neural network-based controllers. This is a very similar architecture to the one designed for our system, which makes this variation of NEAT a promising candidate. The modules executing odNEAT can adapt to changing conditions (i.e. human interaction or change in modules) and learn new behaviours as they carry out their tasks. The online evolutionary process is implemented according to a physically distributed island model. In

the original implementation of odNEAT, each robot optimizes an internal population of candidate solutions (intra-island variation), and two or more robots exchange candidate solutions when they meet (inter-island migration). This might not be enough in our case as, while each robot is potentially self-sufficient, they are not really collaborating as our system requires.

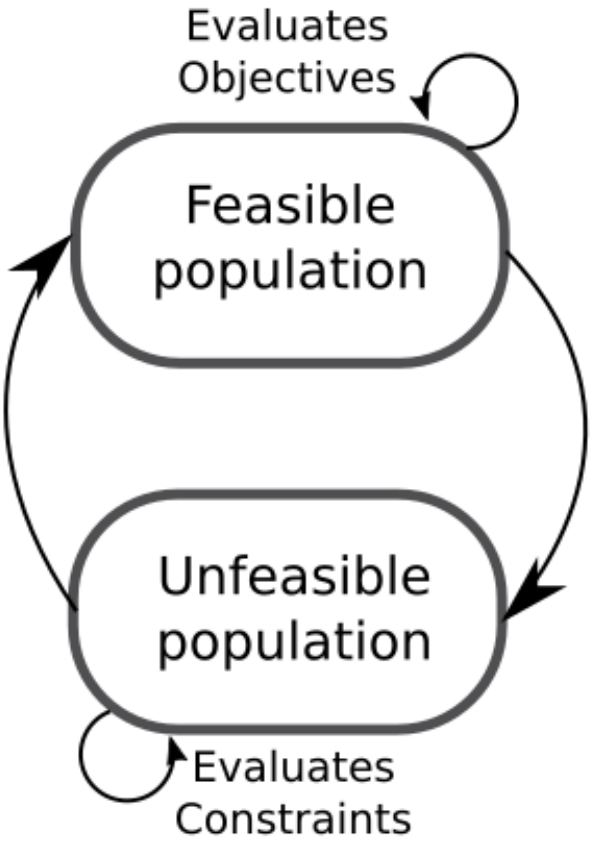

FIGURE 3.1: Diagram of the Feasible Infeasible two population method (FI-2POP): two populations are evolved in parallel, one towards constraint satisfaction and one towards the objective of the search. Exchange of genetic material between the two populations arise when an element of the infeasible population satisfies all constraints, and when an element of the feasible population breaks a constraint.

Chapter 4

Affective music prototype: Moody Music Generator

This chapter presents the initial prototype developed to test our research questions: the Moody Music Generator. This system was originally developed as part of my Master thesis and successively expanded and refined during the first year of my Ph.D. studies. We start by describing the system and then presenting the results of three evaluation studies conducted on it. Two of these studies are only very briefly summarized as they were not integral part of my Ph.D. studies (sections 4.2 and 4.3), but are relevant enough to the research to be given a small space here. References are included so that the interested reader might read the full papers related to these two studies.

4.1 Implementation

The project has been realized using PD (aka Pure Data) [55], a real-time graphical programming environment for audio, video, and graphical processing (see Figure 4.1. In PD, programs are written as graphical graphs called *patches*, in our project we used some patches taken from Brinkmann's website [178] (reverb and delay implementations).

The generated music is played by three instruments (synthesizers) and a drum machine. The system consists of five random number generators: four of these numbers will be converted into notes for the instruments and the drums, while the last one is used as a sound effect controller. Additionally each of the four note generators also generate another number that will determine the volume the note will be played.

These generators create semi-random numbers by adding (in decimal) from a start value a certain step each tick. Then each generator converts the number to another base and adds the digits of the result. This value will then become the note we'll play; at the same time through a slightly different sum we generate another number that will control the volume at which the note will be played.

The numbers, before being sent to the synthesizers that will generate the note, are filtered so that we can control which notes we want

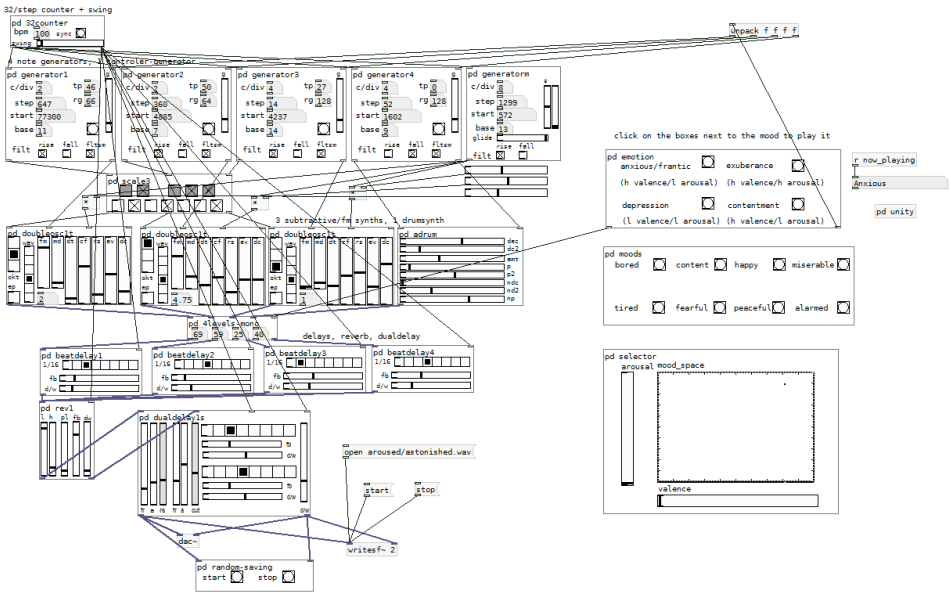

FIGURE 4.1: Moody music generator main patch. Each box is another patch that performs a specific function; while some connections between patches are visible, the connections with the mood control patches are hidden to maintain a semblance of readability of the system. On the left you see the proper generator, which in order from top to bottom consists of layers of: *metronome*, *random number generators* (in MIDI notation), *note filter*, *synthesizers*, *mixer*, *delay* and *reverb*. On the right side the control patches can be seen, the first two allow users to switch between predefined moods, while the bottom one allows them to explore the mood-space in a continuous manner.

and which we do not. This allows us to use dissonances or maintain a diatonic feel to emphasize moods. The numbers at this point represent notes in MIDI (Musical Instrument Digital Interface) notation (they can span from 0 to 127), so to filter them we just use a modulo operation to understand what note they represent. For example, as there are 12 notes in an octave, we can see that 60 modulo 12 equals 0, this means that 60 represents a C.

At this point the synthesizers generate the note, by converting the MIDI notation to a frequency and using PD's built-in audio wave generator. The synthesizers we use are constructed in a way that we can choose the waveform of the sound, modulation, and whether we want to transpose octaves.

After the notes are actually generated we have a patch that controls their volumes, so that we can decide if we want some instruments more prominent than the others: a mixer. First we have all the notes at the desired intensity, but before playing them we apply some effects. Fist we add a delay with feedback to each of the instruments,

afterwards we mix all the sounds and apply a reverb effect. Finally we use a dual delay to the mixed music; by dual delay we mean that we apply two different delays for high and low frequencies.

4.2 Constrained self-report of mood expression

We first conducted a pilot study to check whether our system could possibly represent higher definition moods and the users could recognize the differences in music. Ten students from IT University of Copenhagen, Denmark volunteered to participate in our pilot study. As seen in Figure 2.3, we used Russell's two-dimensional valence/arousal space to locate various types of mood. It is worth noting how these appear in a circular orbit around the origin, this means that the closer we get to the centre the more indistinct the mood would result.

We tried to express some of the adjectives through our music generator. After noticing that some are so close in the space that they were very difficult to differentiate (e.g., astonished and aroused), we finally decided to make eight clips. This decision was also brought forth from the desire to keep the length of the experiment below ten minutes so that the tester wouldn't get tired and so affect the quality of the data. The final emotions that we selected are bored, content, happy, miserable, tired, fearful, peaceful and alarmed. It should be noted that we have two moods that don't appear in Russell's study: fearful and peaceful. With these we wanted to express some feelings that are more commonly found in music, as Russel's study was only focused on emotions and not on music. We defined fearful to be a mood with medium-low valence and medium-high arousal, which would put it very close to the frustrated-annoyed-angry cluster as in Figure 2.3. Peaceful, on the other hand, has medium-high valence and medium-low arousal, so it would be part of the content-satisfied-calm cluster. With this and Content mood, we could explore if people could see a difference between such closely located moods. When we defined the moods, we asked the tester to place the mood they feel in the valence/arousal space. We believed this was important as people might have different definitions of the mood adjectives. It should also note that, for the majority of the participants, English was not their first language.

As the valence/arousal space is not something that most people use in ordinary life, we employed the SAM (Self Assessment Manikin) pictures with two sliders (representing a Likert value from one to five), with the texts describing the meaning of the dimensions [179]. In addition to the demographic data of the study participants, such as age and gender, data relating to their music preference (such as genre) and average time for listening to music were gathered in an open way, by having as answers a four point scale:

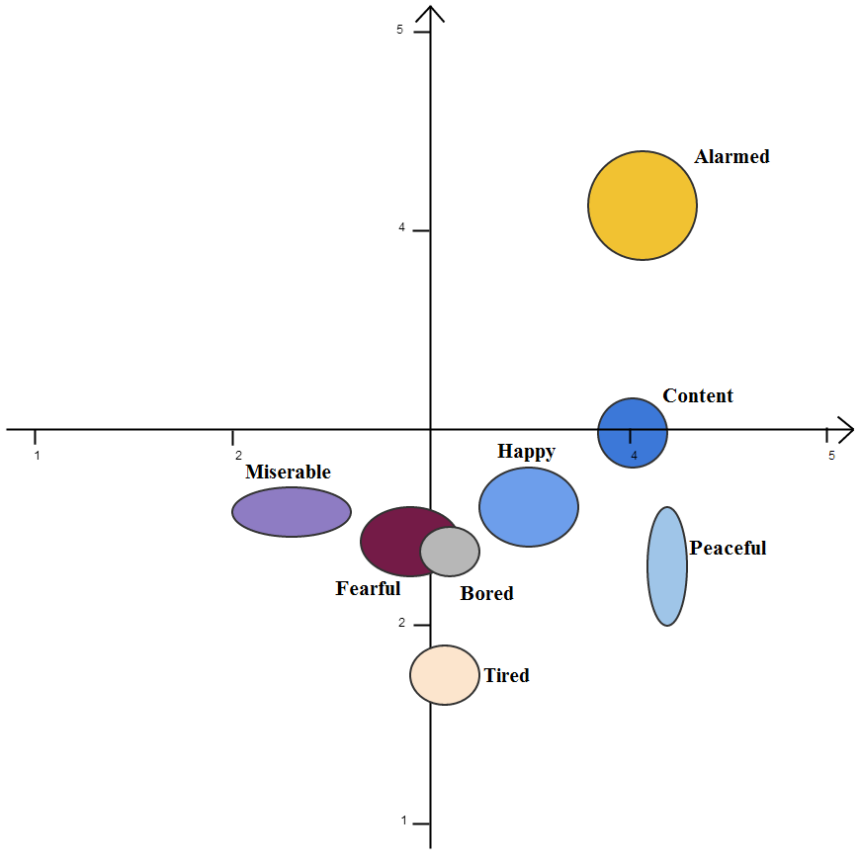

FIGURE 4.2: Distribution of the mood types recognized by the participants. (Vertical axis represents Arousal and horizontal axis represents Valence.)

"always" "often", "seldom", "never". Overall the experiment survey presented: a personal data section, a mood recognition questionnaire (where the participants listened to the clip and then specified which valence/arousal level they felt in the music) and finally a section in which they could select up to two emotion adjectives to describe the piece of music. As an effort to influence the participants as little as possible with the emotion words, we divided their decision of music recognition in two parts. So while listening to the music, the participants could already set valence and arousal values without seeing the emotion adjectives. Once the clip ended, the last part of the form appeared.

4.2.1 Results

Figure 4.2 and Table 4.1 shows the study results including percentage of correct answers by the study participants. The least correct answers were for happy music, as low as 30%. The mood types that had highest recognition were peaceful and alarmed (with an 80% of correct guesses) and bored (with 60%). The mean ratings of arousal and valence for each type of mood music are summarized in Table 2, with

TABLE 4.1: Average ratings of valence and arousal for the eight tested moods. Values can range from a minimum of *1* to a maximum of *5*.

	Valence average	**Valence stdev**	**Arousal average**	**Arousal stdev**
Peaceful	4.2	0.422	2.3	1.252
Fearful	2.9	0.994	2.4	0.699
Bored	3.1	0.568	2.4	0.516
Content	4	0.667	3	0.667
Alarmed	4.1	1,135	4.1	1.101
Happy	3.5	0.972	2.6	0.843
Miserable	2.3	1.252	2.6	0.516
Tired	3.1	0.738	1.8	0.632

a five point scale from 1 (most negative for valence and most calm for arousal) to 5 (most positive for valence and most active/stressful for arousal). For example, we can see how peaceful is perceived as a high valence (4.2) and medium/low arousal (2.3) mood, which is very close to what we expected. Some interesting results are yielded by fearful and alarmed:

- Fearful mood appears to be perceived as a slightly low arousal emotion (2.4), while our expectation was for it to be medium/high.
- Alarmed mood is expected to be an almost pure arousal mood with just a very small negative valence (or almost neutral), but it turned out high valence (4.1) with high arousal (4.1). It seems that the participants had no issue in recognizing the arousal component, but found the music to have a positive valence.

4.2.2 Conclusions

We had some interesting results regarding emotional adjectives. There seems to be a consensus that the semantic meaning of these words is lacking and, moreover, correlations between different emotion words seem to emerge (for example content, peaceful and happy). This made our early analysis seem to have pretty negative results for most moods (apart from peaceful, alarmed and bored). By closely examining the data, however, we could see how the results were much closer to what we expected. While early results of our pilot study with a small number of participants seem to indicate some positive results, it also shows us the problem of using emotional adjectives in this particular type of testing.

4.3 Evaluating musical foreshadowing in videogames

In this section we describe an experiment in using the Moody music generator as to express narrative foreshadowing in a game context. As foreshadowing is generally a subtle clue to the future narrative developments, it is particularly interesting to see if our system is able to reliably express it, and how it can influence the player experience. A game was designed for this experiment, with a focus on its narrative, and an experiment in which the music behaved in different ways to the narrative was conducted.

4.3.1 Foreshadowing

Foreshadowing is a narrative technique where an uncertain, significant event is anticipated, being cued by a less important event. Chatman defines foreshadowing as the seeding of an anticipatory satellite from which a kernel can be inferred [180]. Chatman defines kernel as a key event in the story that advances the storyline and cannot be eliminated without harming the story, while satellite events are minor plot nodes that do not contribute to the main story line. He further suggests that foreshadowing leads to invoking suspense in the reader and surprise in the story character by creating a disparity in knowledge between the character and the reader (that is, the reader knows more information about uncertain future events).

Genette characterizes foreshadowing as *advance mention*, being an indirect and implicit reference that may or may not be confirmed in its importance [181]. He also makes a distinction between two types of advance mention: genuine and false advance mentions. Genuine advance mentions give the reader correct cues about the development of the story, while false advance mentions give misleading knowledge (also called *snares* by Barthes [182]). By mixing these two types of foreshadowing we can engage the reader in trying to analyse the clues and find out which are the true and false ones.

Music can be used to express foreshadowing. This can range from subtly playing in the background music the theme associated with one character to prepare for an entrance (e.g., the *Indiana Jones* franchise makes great use of this expedient) to using the entire music at the time to foreshadow the next event (e.g. the shower scene in Hitchcock's 1960 movie *Psycho*). As evidenced in these films, a great amount of degrees of emotional intensity (and suspense) can be expressed through foreshadowing: for example the latter example creates a much greater feeling of suspense than the former.

FIGURE 4.3: A screen-shot from the game.

4.3.2 Experiment Design

Test groups

To test the relationship between music and gameplay, we first designed three different test groups. Each participant plays a game that has background music that foreshadows the correct future mood (true foreshadowing), a wrong mood (false foreshadowing), and a baseline mood (static - "content" mood).

- **Group 0:** true foreshadowing.
- **Group 1:** false foreshadowing.
- **Group 2:** control group.

Conditions

The test was conducted in a relatively quiet and calm environment. Before the experiment started, the participants were given information about the game controls (mouse and keyboard for interactions), and were told how the mood selector works. Then, the participants played the game with some good quality noise-cancelling headphones. In general, it took about ten minutes to finish the game. When the game ended, the participants were asked to complete the survey.

The Game

To express foreshadowing the game was designed to have a narrative component. Classically, narrative-heavy games have been *adventure games*, defined by Rollins *et al.* as *"a video game in which the player takes the role of the protagonist in an interactive narrative driven by puzzle-solving an exploration"* [183]. This genre had its peak of popularity in

the early to mid-1990s with classical games like *The Secret of Monkey Island* (Lucasfilm Games, 1990) and *Grim Fandango* (LucasArts, 1998). Lately it seems to have reacquired success with the critically acclaimed *The Walking Dead* (Telltale Games, 2012) and *Broken Age* (Double Fine Productions, 2015).

This genre suits our needs very well, as its main focus is on storytelling, and the set of features to implement it is relatively small compared to other genres, minimizing the need to expend development efforts on narratively extraneous game mechanics. Adventure games generally contain branching narratives, but our test game narrative progresses in a linear fashion, due to development time constraint.

The main gameplay consists of:

- Exploring the environment
- Picking up items (and generally interacting with them)
- Using items in the environment to solve puzzles

These gameplay actions are the most basic mechanics used in adventure games.

Narrative/Emotional Design

In this study, we investigated the following eight basic moods generated by the music generator [15]. The main moods that we identified in the story (in narrative chronological order) are:

- **Miserable**: in the beginning of the game, the girl who is the protagonist in the game is sad. This is the main mood.
- **Happy/Content/Peaceful cluster**: soon, it starts snowing and her mood becomes more positive. Foreshadowing can be applied in contrast with the previous mood.
- **Alarmed**: When the snow starts melting this is the main mood; we can use foreshadowing to give a musical cue to the ensuing drama.
- **Peaceful**: Resulting mood of the cathartic moment when the snowman speaks.

With this storyline associated with a particular mood, we identified three key story moments where we can use foreshadowing in a congruous and incongruous way (see figures 4.4 and 4.5).

The first significant event is when **the snow starts falling**, where the main character is uncertain whether this event is positive or negative. However, the resolution makes this positive, transiting the mood of the character to *peaceful*. Therefore, we foreshadow it with *content* for true foreshadowing and *alarmed* for false foreshadowing.

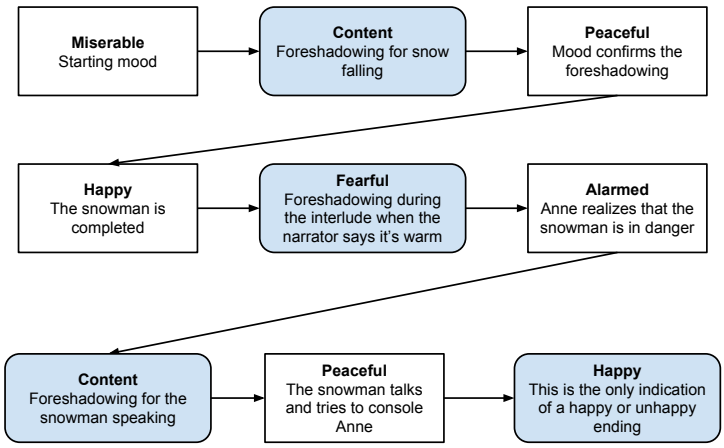

FIGURE 4.4: The emotional structure with **true foreshadowing** (for Group 0). In white we have the persistent moods and in blue the true foreshadowing ones.

The second key event is the realization that **the Snowman is going to melt**, we foreshadow it with *fearful* for true foreshadowing and *peaceful* for false foreshadowing. The event in the end is resolved in the *alarmed* mood.

Lastly, the girl realizes how her efforts to save the snowman are in vain and she despairs and the snowman consoles her about its imminent melting. Just before this event we foreshadow it by using a *content* mood for true foreshadowing or *miserable* mood for false foreshadowing.

Survey questions

We collected more information directly from the participants by having them complete a survey after having played the game. The survey consists of the following questions:

1. **"Select the three moods that you felt more while playing the game"**: in this question the participants could select up to three of the eight affect words describing the moods we can express with our music generator.

2. **"Did you find the music consistent with the narrative?"**: the participant selects one of the five point Likert scales [184] (1 = Strongly Disagree, 2 = Disagree, 3 = Neutral, 4 = Agree, 5 = Strongly Agree).

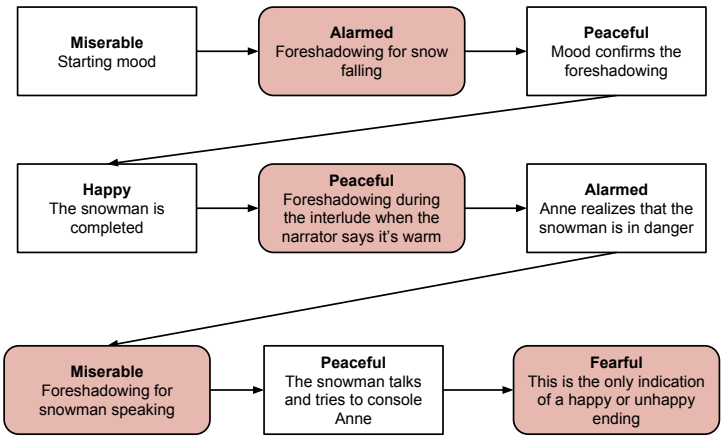

FIGURE 4.5: The emotional structure with **false foreshadowing** (for Group 1). In white we have the persistent moods and in red the false foreshadowing ones.

3. **"Write the first three events that come to mind about the story"**: this question was designed to test the impact of music foreshadowing on the recall of story events.

4. **"Rate these key events by emotional significance"**: the key events listed are:

   - First time seeing the snow
   - Completing the snowman
   - Noticing the snowman might be melting
   - The snowman speaks

   These events represent the resolution of the music foreshadowing as discussed in section 4.3.2. We employ the five points Likert scale (1=Very low significance, 2=Low significance, 3=Medium significance, 4=High significance, 5=Very high significance).

5. **"How much surprise did you feel at these key points?"**: the participant selects one of the five points Likert scale(1=Not surprising, 2=A little surprising, 3=Surprising, 4=Very surprising, 5=Extremely surprising).

6. **"How much did you enjoy the game?"**: the participant selects one of the five points Likert scale (1=Not at all, 2=Enjoyed a little, 3=Enjoyed it, 4=Very much enjoyed it, 5=Extremely enjoyed it).

7. **Personal questions** including demographic information (e.g., gender, age) and experience with written narratives and video games.

4.3.3 Results

A total of thirty undergraduate and graduate students at IT University of Copenhagen in Denmark participated in the study. The number of participants for three test groups are evenly distributed, ten participants for each group.

Music consistency with narrative

First we applied statistical analyses to the data to test how music impacts the gameplay experience. Our first hypothesis was that *players will perceive a higher consistency of the music with the narrative when (true or false) foreshadowing has been employed in contrast with the static music provided for the control group*. Table 4.2 shows the means of the participants' ratings about consistency. A difference of 22.5% was found between the two foreshadowing groups and the control group.

Fisher's test for independence (see Table 4.3) suggests that the difference is not statistically significant. However, it is noted that the difference is non-trivial (p=0.09) when the comparison is made between the two groups employing foreshadowing and the control group.

We can also try grouping all the even slightly positive answers (answers 2 to 5 in the Likert scale) and test them against the strongly negative one: in this case we obtain a p-value of almost 9%.

TABLE 4.2: Means of the **music consistency** ratings. Values can range from a minimum of *1* to a maximum of *5*.

	True F.	**False F.**	**Control**
Consistency	3.6	3.6	2.7

TABLE 4.3: The p values calculated with Fisher's test for the **consistency** ratings. (We have omitted the comparison between the *true foreshadowing groups* and *false foreshadowing group*, as the groups gave exactly the same answers.)

	All	**True F./Control**	**Grouping answers**
p	0.552	0.337	*0.089*

TABLE 4.4: Means of the **general enjoyment** ratings. Values can range from a minimum of *1* to a maximum of *5*.

	True F.	**False F.**	**Control**
Enjoyment	2.7	3.3	2.5

TABLE 4.5: The p values calculated with Fisher's test for the **enjoyment** ratings.

Comparisons	p
All groups	0.074
True F. / False F.	*0.045*
True F. / Control	0.546
False F. / Control	*0.046*

General enjoyment of the game

We analysed the data to test the hypothesis that the *participants in the groups that had foreshadowing music would enjoy the game more than the control group*. The data is the one collected through the survey question regarding enjoyment (see section 4.3.2).

We note that the means of enjoyment ratings (see Table 4.4) for *Group 0* and *Group 1* are higher than those ratings obtained from the participants in *Group 2*. In fact, we have a respective difference from the control group's score of 5% and 20% for the True and False foreshadowing groups.

The Fisher's test on the data shows a promising result of $p = 0.074$ (see Table 4.5), although not statistically significant. Interestingly, a significant difference was found in the player's enjoyment between the True foreshadowing and the False foreshadowing groups. This means that the participants experience higher enjoyment when false foreshadowing was used than when true foreshadowing was used.

4.3.4 Conclusions

While research is very active in the fields of procedurally generated music, studies on musical (and narrative) foreshadowing combined with audio in games are rare. Our study set out to explore if mood expressive music can foreshadow game narrative events and its impact on the gameplay experience. We designed and implemented a game prototype and collected the players' psychological perception about key story events, emotional states, and the recall of story events.

We will now summarize our empirical findings to answer our research questions stated earlier (All the empirical results are described in depth in section 4.3.3) and do some final consideration of what these might mean:

Q1 *In games, can procedurally generated mood expressive music serve as an indicator of narrative foreshadowing?* The high value of p through Fisher's test (between all the three groups) that we showed in section 4.3.3 is due to the fact that we were testing all the groups together, while our hypothesis implies a dependency between the *true foreshadowing group* and the *false foreshadowing group*.

The p-value calculated between any of the foreshadowing groups against the control group, while being a better result, is not statistically significant. This is possibly due to the small amount of participants and relatively large number of possible answers: each group has ten participants of which each can choose one out of five answers. The p-value calculated by grouping the positive answers together (see Table 4.3), however, shows a potential relationship between perceived music consistency and the use of foreshadowing.

Q2 *Can the use of musical foreshadowing improve the player's experience?* The enjoyment mean ratings of the participants (see Table 4.3.3) show that there is a possible difference for the False foreshadowing group. The Fisher's test (see Table 4.5) shows that there is a statistically significant difference between how much the true and false foreshadowing groups enjoyed the game.

Confronting the data from these groups with the control group we notice that we don't have a statistically significant difference between the true foreshadowing group and the control group. We can then infer that the use of true foreshadowing results neither positive nor negative towards the enjoyment of the story compared with a static soundtrack.

We can then conclude that probably **the usage of false foreshadowing creates a more enjoyable experience than true foreshadowing**. We hypothesize that this is due to false foreshadowing producing an additional element of surprise, which added more interest than the case in which foreshadowing caused the player to expect the outcome. It's worth noting that our study had only true or false foreshadowing, whereas it is possible for soundtracks to mix both true and false foreshadowing in one game. Therefore we don't suggest that composer should avoid true foreshadowing entirely, only that using false foreshadowing to subvert the players' expectations, at least occasionally, may increase the role the music plays in heightening narrative interest.

There are a few limitations that are important to recognize in our study. First, there were a limited number of test participants. Second, the study was conducted with only one instance of one genre of game, which may not generalize to the broad space of possible game designs. The number of test participants, while not being too small for us to obtain any meaningful result, gave us some inconclusive data, especially when the answers were free text. It also provided for a more difficult analysis of the five-point Likert scale answers, since

each group of ten participants could choose between five possible answers.

Lastly, we should consider the limitations dictated by our game: we chose to make an *adventure game* because of the important narrative component of these games, but this is of course only one particular genre and we might obtain different results in a first person shooter. Further studies would be necessary to prove or disprove the transferability of our results in other game genres.

Not only have we shown through our study that through procedurally generated music we can express foreshadowing in our game, but we have identified some characteristics that distinguish true and false musical foreshadowing and found some of the limits of the influence we can exert on players through these techniques.

4.4 Free text characterization of control parameters

4.4.1 Experiment design

As described in the previous section, we produced a generator intended to be parameterised by two control axes: arousal and valence. Although this construction is based on theoretical motivations and existing work on the relationship between musical parameters and perceived mood, it does not necessarily follow that these axes *actually* represent arousal and valence. To understand what kind of generative space our music generator actually produces, we designed a study to characterise how the two control axes of our generator influence listeners' perceptions.

Contrary to our previous pilot study [15], we employed a mix of closed-ended questions to validate the axes (e.g. a number of mood expressing words and a Likert scale for valence and arousal), we decided to provide completely open-ended questions to the participants, so as to eliminate as much bias as possible from their answers, and understand the effects of our generator's control parameters in an open-ended way.

We developed the online survey with HTML and PHP, using a MySQL database to hold the data collected; the participants were presented with a page consisting of a music clip and five blank boxes where they were asked to write emotional words that they thought the music expressed.

After each of the five responses we introduced a special page where the participants could review their answers, listen to the previous five clips again and see some of the most recent answers from other users for the same clips. We created this page to give feedback to the users and to make the survey, hopefully, more interesting for

them by giving them the opportunity to confront their answers with the ones other users provided.

The experiment has no pre-defined end: the user is able to continue answering until he/she wants, and can close the online survey at any time.

4.4.2 Music clip generation

We generated 100 clips of 30 seconds of music using our music generator, each of these expressing a randomly chosen point in the bi-dimensional mood space we described in section 2.7.

The music clips have been generated by linearly connecting the features and the respective axis, even though we are conscious that the relationships are probably more complex; in fact we hope the data collected through this study will help us better define these.

The maximum and minimum bounds we gave to the various musical features were:

- **Tempo**: 100-136
- **Intensity**:

	Synth 1	Synth 2	Synth 3 (Bass)	Drum machine
Minimum values	69	56	35	60
Maximum values	98	119	83	128

- **Rhythm strength**: -20% to +20% intensity to *Synth 3* and *Drum machine*.
- **Timbre**: three different settings for the synthesisers: the lowest is selected when the valence is less or equal of 33, the middle between 33 and 66 and the higher above 66 (valence goes from 0 to 100 in our system).
- **Steadiness**: three settings dependent on the valence axis as the Timbre: *steady*, *medium_steady* and *unsteady*. On the *steady* rhythm all the notes fall on the beat of the measure, on the *medium steady* rhythm all instruments play notes on the beat, while the drum machine plays off-beat. Finally for the *unsteady* rhythm only Synth 1 (which is the higher pitched instrument, and the one more resembling a lead voice) plays on the beat while all the other instruments play offbeat. Note that if all the instruments played on the offbeat the listener would have no way of telling the beat from the offbeat, effectively perceiving a steady rhythm.

- **Dissonances**: as discussed in section 5.2.2, we use a C major scale (C D E F G A B) for positive and an E♭ harmonic minor scale minus the third grade (E♭ F [G♭] Ab B♭ B D) for negative valence.

4.4.3 Results and analysis

We collected a total of 2020 free-text labels from 192 distinct users. We can consider patterns in these labels to constitute an open-ended, nonparametric characterisation of how users perceive the music's mood as we vary the control parameters intended to represent arousal and valence. The obvious question is then: are there any patterns, and do they provide any insight into the effects of these control parameters? With free-text labels, it is not entirely implausible that there could end up being no easily discernible patterns in the data. However there turn out to be some strongly localisable responses, particularly among the labels volunteered relatively frequently. Although listeners could in principle respond with any English word or phrase, some words recur often, e.g. "mysterious" was volunteered 34 times.

In order to characterise the control parameters using these labels from the users themselves, for each label we calculate the average (mean) arousal and valence of the clips for which that label was volunteered. The goal of doing so is to localise the label somewhere in the two-dimensional control space. We would like to say things such as: the label "rushed" appears on average in the high-arousal, high-valence part of the space, while the label "relaxed" is given on average to low-arousal, low-valence clips.

Of course, if a given label was only volunteered a few times, an average is not very reliable. Therefore we choose the 20 labels which are *best localised*, in the sense that we have enough data to more reliably determine their average location in the control space. To determine how well localised a label is, we rank labels by the standard error of their mean location on the arousal/valence axes (standard errors summed over both axes). The standard error of a mean, $se = \frac{stddev}{\sqrt{n}}$, will in general be lower for labels with lower sample standard deviations, and for labels which appear more times in the data set.[1]

Before ranking labels by standard error, we perform two preprocessing steps on the data. First, we stem the words using the Snowball stemmer,[2] in order to aggregate minor part-of-speech variations of labels—for example, *relaxed* and *relaxing* are both mapped to the

[1] Since we make no assumption about the distribution of data, we can't use the standard error as a basis for a confidence interval. Nonetheless, it is useful as a proxy for how well we can localise a label in the arousal/valence space, relative to other labels in our data set.

[2] http://snowball.tartarus.org/

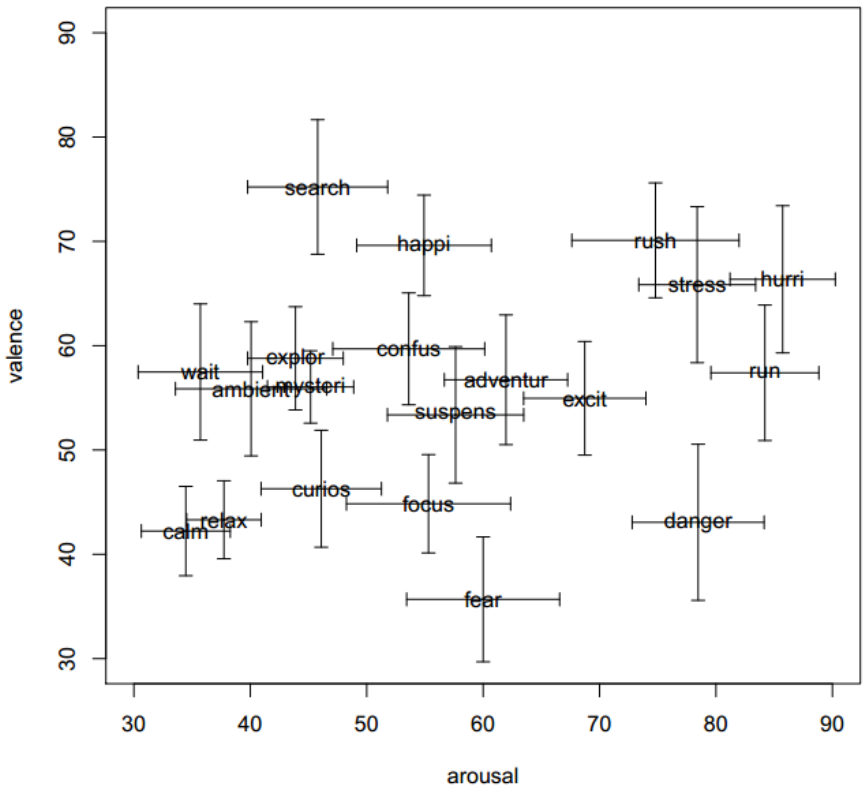

FIGURE 4.6: Free-text, crowdsourced characterization of moods across the generator's two-dimensional control space. Plotted labels are the 20 best-localised labels (post-stemming), plotted at the average location for which they were volunteered as labels. Error bars represent the standard error of the mean.

stem *relax*. In addition, we exclude labels that appear fewer than 5 times in the data set even after stemming, because the sample standard deviation is an unreliable measure for extremely small n.

Figure 4.6 plots the 20 best-localised label stems, at the average location of the (arousal, valence) parameter settings that elicited that label as a response. The standard errors of the mean are plotted as indicative error bars. This plot alone is surprisingly informative as a characterisation of the control parameters' effect on perception of musical moods. Especially considering that users volunteered free-text labels rather than selecting categories, the trends in the axes are rather striking.

We can make a few qualitative observations on the basis of these 20 well-localised labels. In a general sense, the "arousal" and "valence" theory that drove the development of our control axes does not seem to precisely align, in this setting, with the effect of the axes to which we've nominally given those labels, though arousal is closer than valence.

Arousal maps to something like a calm vs. stress axis (which is,

in fact, the definition of arousal). Low-arousal words include (unstemmed): curious, waiting, calm, relaxing, and ambient. High-arousal words include: rushed, stressed, hurried, run, and dangerous. Valence seems to be largely dominated by arousal, but modifies it in a way that has strong interaction effects.

Looking at high-valence clips, when coupled with a high arousal setting, they elicit labels that accentuate a feeling of being rushed: rushed, stressed, and hurried. So, raising valence, rather than being perceived as positive valence, instead emphasises a kind of speed in the context of high arousal, with even a somewhat negative sentiment. Low-valence, high-arousal clips are most often labelled as "danger" instead.

With a mid-level arousal setting, valence does seem to act as a relatively straightforward valence setting: high-valence clips are characterised by "happy", and low-valence clips by "fear". As arousal drops, however, the effect of the "valence" setting becomes particularly inconsistent with the intent that it be a knob used to vary perceived valence. The nominally low-valence clips, when coupled with low arousal, elicit quite *positive* labels: calm, relax. As "valence" increases while arousal stays low, the main effect is to get somewhat more active: from "calm" to "wait" and "explore" at mid-valence, to "search" at high valence settings.

4.5 Limitations

In classically composed music the key and mode have a tremendous effect on the piece, not to mention cadences key changes. In our generated music we have no real chord progression, the chords are created by the superposition of the random notes played by the three instruments; while in a classical piece much effort is spent in making chord progressions and melodies.

Our generated music lacks in both of these aspects (unless they appear randomly), this makes our music good for ambiance, but it lacks the depth of a normal piece of music. We should also note that if melodies appear randomly there is no system of repetition or alteration of melodies, as we don't memorize anything.

This is also the reason why we focus on moods instead of emotions: in a normal piece of music a lot of preparation is necessary to express emotions: the composer has to create a basic mood, then create a tension progression to then arrive at a climax where the tension gets released in a more or less satisfying resolution (depending on the effect wanted). Generally, apart from the harmonic aspect, there is also a melodic one that guides the listener through the piece, another feature that we lack.

An interesting fact we realized is that adding more instruments to our generator would probably make the final effect sound much worse. Right now, in a single instant, we can have at most a triad of

notes played at the same time. This means that the chords created can be:

- A normal chord (major or minor), composed by the *root*, *major or minor third* and *fifth* (e.g. C [C E G] or Dm [D F A])
- Suspended chords, where the third is replaced by the second or fourth (e.g. Csus2 [C D G] or Csus4 [C F G])
- Chords that are difficult to recognize because the root of the chord is ambiguous (e.g. a triad composed by [C E B] could be considered a C7 with the fifth omitted or also a Esus6)

If we imagine having four instruments we could potentially have chords containing four different notes (out of the seven of the diatonic scale we are in). In this case it would be very difficult to identify the root of our chord and, without some system intervening to prepare these chords, the result would be cacophonous.

Note that all these examples have been written by considering to be in the C major key, so we do not have any altered chords (like an augmented or diminished triads: Caug [C E G♯], Cdim [C E G♭]) as the scale we use in *Contentment* and *Exuberance* does not admit alterations. The only exception to this rule is the only diminished chord that belongs in the C major scale: Bdim [B D F].

Chapter 5

MetaCompose

As stated in the previous chapter, the Moody Music Generator was inadequate for the purposes of creating music that sounded interesting enough to use in the video-games domain. Therefore a new music generator was developed, which maintained the necessary components of being real-time, dynamic, and expressive, while being able to create more complex and interesting music. Another additional feature of METACOMPOSE, compared with the Moody Music Generator, is that it creates much more structured music. This chapter describes this system in detail, while chapter 6 presents the results of the evaluation studies we've conducted on it.

METACOMPOSE consists of three main components: (i) *composition generator* (Section 5.1), (ii) *real-time affective music composer* (Section 5.2) and (iii) an *archive* (Section 5.3) of previous compositions (Fig.5.1). The modular nature of METACOMPOSE allows components to be easily swapped for others or augmented with further components.

The *composition generator* (i) creates the basic abstraction of music that will be used by the *real-time affective music composer* in order to (ii) create the final score according to a specific mood or affective state. In other words, as a metaphor, the *composition generator* (i) serves as a composer that only writes the basic outline of a piece, while the *real-time affective music composer* (ii) acts as an ensemble, free to interpret the piece in different ways. The archive (iii) maintains a database of all the previous compositions connected to the respective levels/scenes of the game-state while also allowing a rank to be computed that measures the novelty of future compositions compared to those previously generated. METACOMPOSE is designed to be able to react to game events depending on the effect desired. Examples of responses to such events include: a simple change in the affective state, a variation of the current composition, or an entirely new composition.

5.1 Composition Generation

Composition in this paper refers to an abstraction of a music piece composed of is a *chord sequence*, a *melody* and an *accompaniment*. It

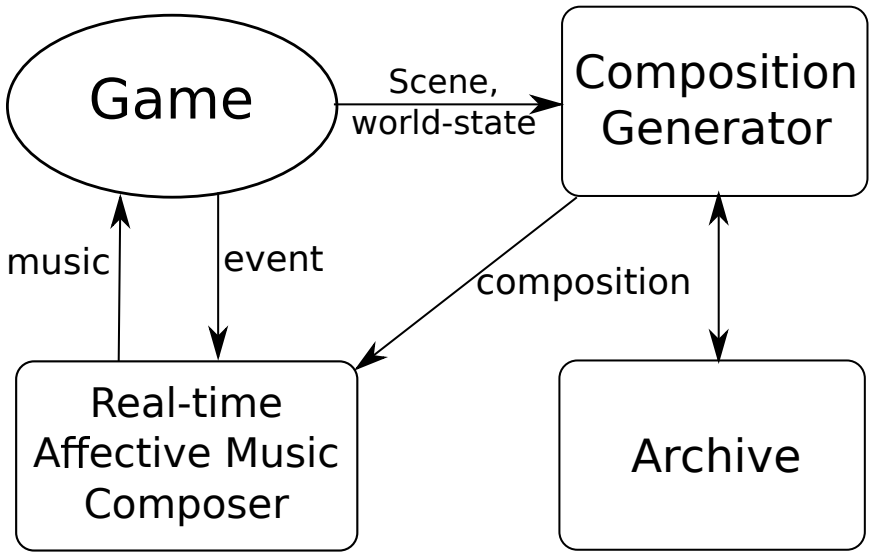

FIGURE 5.1: METACOMPOSE's architecture.

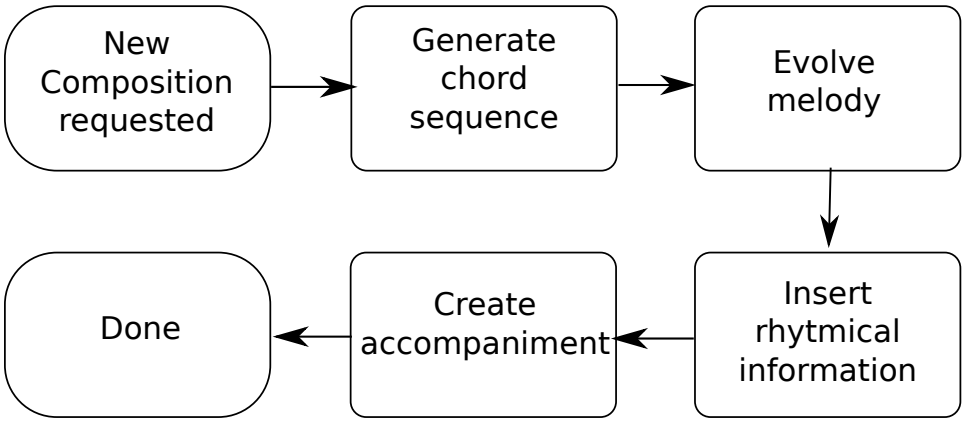

FIGURE 5.2: Steps for generating a *composition*.

is worth noting that the term *accompaniment* denotes an abstraction, not the complete score of a possible accompaniment, described in detail in Section 5.1.3 (below). The main reason for the deconstruction of compositions is to produce a general structure (an abstraction) that makes music recognizable and provides identity. Generating abstractions, which themselves lack some information that one would include in a classically composed piece of music, e.g. tempo, dynamics, etc., allows METACOMPOSE to modify the music played in real-time depending on the affective state the interactive media wishes to convey. The generation of compositions is a process with multiple steps: (i) creating a chord sequence, (ii) evolving a melody fitting this chord sequence, and (iii) producing an accompaniment for the melody/chord sequence combination (see Fig. 5.2).

5.1.1 Chord Sequence Generation

The method for generating a chord sequence works as follows: random walks are performed on a directed graph of common chord sequences (see Fig. 5.3) starting from a given chord. Referring to Figure 5.3, the graph does not use a specific key, but rather 'degrees': in music theory, a degree (or scale degree) is the name given to a particular note of a scale to specify its position relative to the 'tonic' (the main note of the scale). The tonic is considered to be the first degree of the scale, from which each octave is assumed to begin. The degrees in Fig. 5.3 are expressed in Roman numerals and, when talking about chords, the numeral in upper-case letters symbolizes a major chord, while lower-case letters (usually followed by an m) express a minor chord, which is sometimes omitted. Other possible variations on the chord are generally expressed with numbers and other symbols, these are not listed for the sake of brevity. So, if we consider the D major scale, the $Dmajor$ chord would correspond to a I degree, while a $iiim$ degree would be a $F\sharp minor$. Various parameters of the generated sequence can be specified, such as sequence length, first element, last element, chord to which the last element can resolve properly (e.g., if we specify that we want the last chord to be able to resolve in the V degree, the last element might be a IV or a iim degree).

An interesting aspect of this graph is that it also shows common resolutions to chords outside of the current key, which provide a simple way of dealing with key changes. Each chord can be interpreted as a different degree depending on which key is considered, so if we want a key change we can simply: (i) find out which degree the last chord in the sequence will be in the new key and (ii) follow the graph to return to the new key. This should produce harmonious key changes that do not sound abrupt.

5.1.2 Melody Generation

Melodies are generated with an evolutionary approach. We define a number of features to include (objectives) and to avoid (constraints) in melodies, these are based on classical music composition guidelines and musical practice. These features are divided into constraints and objective functions. Accordingly, we use a Feasible/Infeasible two-population method (*FI-2POP* [24]) with multi-objective optimization [186] for the Feasible population. Given a chord sequence, a variable number of notes is generated for each chord, which will evolve without duration information. Once the sequence of notes is created, we generate the duration of the notes pseudo-randomly.

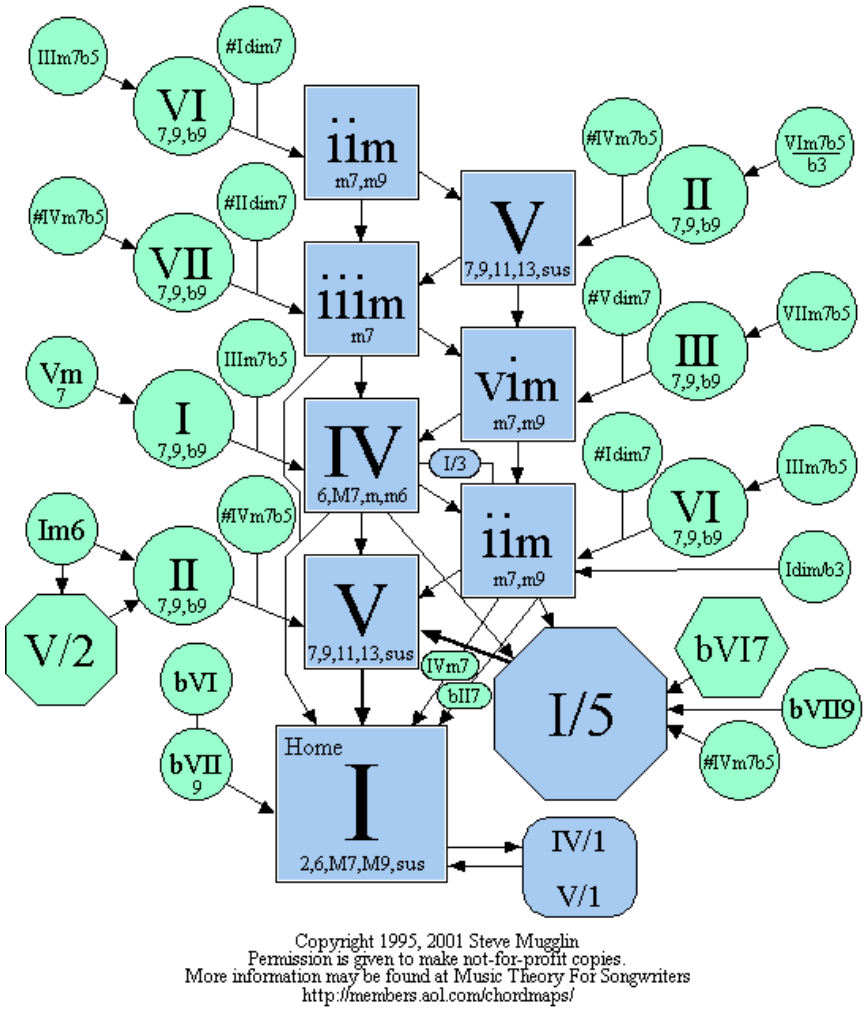

FIGURE 5.3: Common chord progression map for major scales, created by Steve Mugglin [185].

Genome Representation

The evolutionary genome consists of a number of values (the number of notes to be generated) that can express the notes belonging to two octaves of a generic key (i.e. 0–13). Here, we do not introduce notes that do not belong to the key, effectively making the context in which the melodies are generated strictly diatonic. Alterations will appear in later stages, in the real-time affective music composer module, when introducing variations of the composition to express affective states or chord variations. The length of the genome depends on how many chords were generated in the previous step and can range from 1 to 8 notes per chord.

Constraints

We have three constraints: a melody should: (i) *not have leaps between notes bigger than a fifth*, (ii) *contain at least a minimum amount of leaps of a second* (50% in the current implementation) and (iii) *each note pitch*

should be different than the preceding one.

$$\textit{Feasibility} = -\sum_{i=0}^{n-1}(\textit{Second}(i,i+1) + \textit{BigLeap}(i,i+1) + \textit{Repeat}(i,i+1))$$
where n is the genome length (5.1)

The three functions comprising eqn. 5.1 are all Boolean functions that return either 1 or 0 depending if the two notes at the specified indexes of the genome satisfy the constraint or not. As can be seen, this function returns a number that ranges from (potentially) $-3(n-1)$ to 0, where reaching the score 0 determines that the individual satisfies all the constraints and, consequently, can be moved from the unfeasible population to the feasible population.

On the constraints in eqn. 5.1, leaps larger than a fifth do appear in music but they are avoided here, as experience suggests they can be too hard on the ear of the listener [187]. Namely, if the listener is not properly prepared, leaps larger than a fifth can easily break the flow of the melody. We also specify a minimum number of intervals of a second (the smallest interval possible considering a diatonic context such as this one, see the *Genome Representation* section) because if the melody has too many larger leaps it feels more unstructured, and not something that we would normally hear or expect a voice to sing. Finally, the constraint on note repetition is justified by the fact that repetitions will be introduced by the *real-time affective music composer*.

Fitness Functions

Three objectives are used to compose the fitness functions: a melody should (i) *approach and follow big leaps (larger than a second) in a counter step-wise motion (explained below)* (eqn. 5.2), (ii) where the melody presents big leaps the *leap notes should belong to the underlying chord* (eqn. 5.3) and finally (iii) *the first note played on a chord should be part of the underlying chord* (eqn. 5.4).

First, we remind the reader that the definition of an interval in music theory is the **difference between two pitches**. In Western music, intervals are mostly differences between notes belonging to the diatonic scale, e.g. considering a $Cmajor$ key, the interval between C and D is a *second*, the interval between C and E is a *third* etc.

$$\text{CStep} = \sum_{i=0}^{n-1} \frac{\text{IsLeap}(i,i+1)(\text{PreCStep}(i,i+1) + \text{PostCStep}(i,i+1))}{leapsN}$$
where $CStep$ stands for $CounterStep$, $PreCStep$ for $PreCounterStep$, and $PostCStep$ for $PostCounterStep$ (5.2)

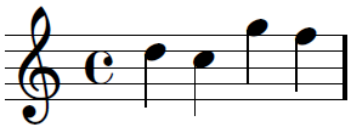

FIGURE 5.4: An example of counter step-wise approach and departure from a leap (C-G).

$CStep$ (eqn. 5.2) measures *counter step-wise approach and follow to big leaps*. To clarify what *counter step-wise motion* means: if we examine a leap of a fifth from C to G, as in Fig. 5.4, (assuming we are in a $Cmajor$ key), this is a upward movement from a lower note to a higher note, a counter step-wise approach would mean that the C would be preceded by a higher note (creating a downward movement) with an interval of a second, so a D. Likewise, following the leap in a counter step-wise motion would mean that we need to create a downward movement of a second after the G, so we need an F to follow.

The reason we introduce this objective is that this makes leaps much easier on the listener's ear, otherwise counter step-wise motions often sound too abrupt by suddenly changing the range of notes the melody is playing. The `PreCStep` and `PostCStep` functions are Boolean functions that respectively check if the note preceding and following the leap approaches or departs with a contrary interval of a second.

The reason for having the leap notes – the two notes that form a leap larger than a second – as part of the underlying chord, is that such leaps are intrinsically more interesting than a step-wise motion, this means that the listener unconsciously considers them more meaningful and pays more attention to them[187].

When these leaps contain notes that have nothing to do with the underlying chord, even if they do not present real dissonances, they will be perceived as dissonant because they create unexpected intervals with the chord notes. Including leaps as part of the chord gives a better sense of coherence that the listener will consider as pleasant.

$$\text{ChOnLeap} = \sum_{i=0}^{n-1} \frac{\text{IsLeap}(i, i+1)(\text{BeToChord}(i) + \text{BeToChord}(i+1))}{leapsN}$$

where $ChOnLeap$ stands for $ChordOnLeap$, and $BeToChord$ for $BelongsToChord$ (5.3)

The $ChOnLeap$ function (eqn. 5.3) calculates how many *leap notes belong to the underlying chord* by checking if there is a leap between each two notes of the melody ($IsLeap$) and, if that is the case giving a positive score for each of the two notes that is part of the underlying chord ($BeToChord$ = belongs to chord).

$$\text{FirstNtOnChord} = \sum_{i=0}^{n} \frac{\text{IsFirstNote}(i) \times \text{BeToChord}(i)}{chordsN}$$

where $FirstNtOnChord$ stands for $FirstNoteOnChord$, and $BeToChord$ for $BelongsToChord$ (5.4)

The last objective (eqn. 5.4) emphasizes the importance of the first note after a chord change, by playing a note that is part of the chord we reinforce the change and make the chord change sound less discordant.

Note that these objectives, by the nature of multi-objective optimization, will generally not all be satisfied. This is acceptable, as satisfying all objectives might make the generated music sound too mechanical and predictable, while such "soft" rules are only enforced to a certain point, namely the Pareto frontier (contrary to the constraints of the infeasible population, which always need to be satisfied).

5.1.3 Accompaniment Generation

Accompaniment is included in the composition because, not only do chords and melody give identity to music, but also rhythm. Accompaniment is divided into two parts: a **basic rhythm** (a collection of note duration) and a **basic note progression** (an *arpeggio*). We can progress from the accompaniment representation to a score of the accompaniment by creating notes with duration from the basic rhythm and pitches from the progressions (offset on the current underlying chord).

In previous work[188], we described an approach to generating rhythms consisting of a stochastic process involving combinations and modifications of some elements taken from a small archive of basic rhythms. METACOMPOSE was upgraded to make use of *Euclidean rhythms* [189], which have the property that their onset patterns are distributed as evenly as possible. Toussaint also shows how such rhythms include a large family of rhythms used as rhythmic *ostinatos* in world music. Euclidean rhythms can be generated very efficiently and fulfil the objective of having a basic "pulse" for the instruments to follow.

We will here adapt an example described by Bjorklund [190] to creating rhythms for the music domain: consider a sequence with with $n = 13$ *pulses*, we want to create a rhythm in this pulse series that contains $k = 5$ *onsets*. We represent these in a binary fashion, where a 1 represents a note offset while a 0 represents a rest. In this example we would have $13 - 5 = 8$ rests, so we start by considering a sequence of 5 ones followed by 8 zeroes.

[1][1][1][1][1][0][0][0][0][0][0][0][0]

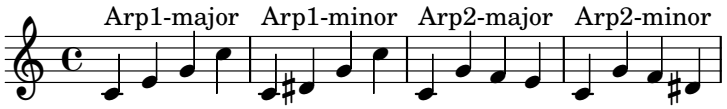

FIGURE 5.5: Basic arpeggios. These are represented as if they were played under a C major or C minor chord, but are transposed depending on what chord they appear underneath. Also the rhythmic notation of the arpeggio is dependent on the rhythmic structure.

If there are more than one identical arrays at the end of the sequence (in this case $[0]$), we proceed to concatenate these arrays one at the time (from right to left) with the other (different) arrays in the sequence. This way 5 of the $[0]$ will be concatenated with the 5 $[1]$, producing 5 $[10]$, and leaving 3 $[0]$ behind:

$$[10][10][10][10][10][0][0][0]$$

Again there is more than one identical array trailing the sequence ($[0]$), so we concatenate those with the first arrays in the sequence:

$$[100][100][100][10][10]$$

This time the repeated trailing arrays are $[10]$, and by concatenating them we obtain:

$$[10010][10010][100]$$

Finally the process ends, as the trailing array is unique ($[100]$), and the final rhythm is obtained by the concatenation of all the arrays:

$$[1001010010100]$$

Note that the problem becomes trivial when the number of onsets k can evenly divide the number of pulses n. For example, if $k = 4$ and $n = 12$, we would obtain the $[100100100100]$ sequence.

Arpeggios are generated through a stochastic process involving combinations and modifications (inversions, mutations, etc.) of some elements taken from a small archive of basic rhythms. Specifically we have two basic arpeggios (see Fig. 5.5).

The algorithm performs the following steps:

1. choose a basic arpeggio;
2. shuffle the elements of the arpeggio;
3. increase the arpeggio to match the size of the basic rhythm (done by introducing at a random index of the arpeggio a new random pitch that already belongs to the arpeggio).

The rhythm presented in the final music will be modified by the real-time affective music composer for variety or for affect expression, while still maintaining a rhythmic and harmonic identity that will be characteristic of the composition.

Implementation details

This section serves to describe the specific operators and parameters used to obtain the results discussed later in the paper. The feasible population (i.e. the one running NSGA-II) utilizes a **binary tournament** selection operator: two random individuals are chosen from the population and are compared. The individual that dominates the pair is selected as a parent for the crossover operator, obviously this operator has to be executed twice to obtain the two parents required by the crossover operator. In the event no one individual dominates the pair, a parent is chosen randomly from the two. The unfeasible population uses a **roulette-wheel** selection operator: the selection is a stochastic process where individuals have a probability of becoming parents for the next generation proportional to their fitness. This way individuals with higher fitness are more likely to be selected while individuals with lower fitness have a lesser chance, they may have genetic material that could prove useful to future generations. Both populations adopt a simple **single point crossover** operator.The **mutation** operator gives each gene a probability $1/l$, where l is the genome length, to mutate. This ensures that on average only one gene will mutate but allows for more than one or no mutation to occur. The mutation itself transforms the note the gene represents to either the note directly above or directly below the mutated note in the scale.

The parameters used are:

- Population size: 500
- Generation number: 5000
- Elitist factor: 25%
- Mutation rate: 10%

5.2 Real-Time Affective Music Composer

The purpose of the real-time affective music composer is to go from the abstract compositions we have created, to actual music. There are two main components of the process: a stochastic interpretation of the composition that creates music that reflects the abstraction while presenting variations, and the modification of some musical and sonorous features to express affective meaning.

5.2.1 METACOMPOSE Implementation

The system builds on a number of *instruments*, each with an algorithm to interpret some (or all) of the data that comprises a composition; this interpretation is responsible for turning the abstraction into the score the system actually plays. These algorithms can be simple or complex, but should always contain some stochastic element, in order to avoid repetition. We created this instrument system to give some creative freedom to the developers that might use this music generation system; for example by making the instruments play accordingly to a specific music style or to create music that fits better with their accompanying media.

The instruments send the notes they generate in real time to the standard MIDI device provided by the Java language. The system can be configured to be connected to an external MIDI device to provide for sound synthesis.

5.2.2 Mood expression theory

We now describe our model for mood expression in terms of music theory and how mood influences the production. We propose four musical features that influence perceived mood in music, these are: *intensity*, *timbre*, *rhythm*, and *dissonances*. These are mainly inspired by Liu *et al.* [191]. While Liu *et al.*'s research focused on mood classification via machine learning, we applied and expanded their model to generate music instead.

Intensity (or Volume)

is defined by how strong the volume of the music is. It is an arousal-linked feature: high arousal corresponds to high intensity; low arousal to low intensity. Intuitively, high volume music results in increased stress. In a similar way, lower volume music, being less intense, is less arousing and has lower intensity.

Timbre

is defined as the combination of qualities of a sound that distinguishes it from other sounds of the same pitch and volume. For example, timbre is what makes the C4 note sound different when played on a piano as opposed to a guitar. It is often associated with *"how pleasing a sound is to its listeners"* [192]. One of timbre's most recognizable feature is what we could call "brightness", that is, how much of the audio signal's frequencies are distributed. In previous literature audio features such as MFCC (Mel-Frequency Cepstral Coefficients [193] and spectral shape features [194] have been used to classify music on the basis of its timbral feature. We link timbre with valence: the more positive the valence, the brighter the timbre.

Rhythm

is divided into three features: strength, regularity and tempo [191].

- Rhythm strength: how prominent is the rhythmic section is (drums and bass). This feature is linked to arousal and our system acts by regulating the volumes of the instrument currently considered the “bass" to be proportionally higher or lower in the general mix.
- Regularity: how steady the rhythm is. This feature is linked to valence.
- Tempo: how fast the rhythm is. This feature is linked to arousal and influences the beats-per-minute (BPM) that the instruments follow.

In a high valence/high arousal piece of music, for instance, we observe that the rhythm is strong and steady. In a low valence/low arousal, on the other hand, the tempo is slow and the rhythm not as easily recognized.

Dissonance

is the juxtaposition of two notes very close to each other: for example C and C♯. The distance between these two is just a semitone, which gives the listener a generally unpleasant sensation. A dissonant interval does not always sound bad. In fact most music contains dissonances, they can be used as cues expressing something amiss. The listener's ear can also be trained to accept dissonances through repetition, which explains why some music genres rely heavily on intervals that are avoided by other ones.In general, the larger the interval between the two dissonant notes, the 'easier' it is on the listener's ear: a C and a C♯ are always dissonant, but the dissonance is more evident if the notes are played from the same octave. C.P.E. Bach, in his *Essay on the True Art of Playing Keyboard Instruments* [195], remarks on the affective power of dissonances, although in a more general way: “... dissonances are played loudly and consonances softly, since the former rouse our emotions and the latter quiet them".

Meyer [196] observes that the affect-arousing role of dissonances is evident in the practice of composers as well as in the writings of theorists and critics, remarking how the affective response is not only dependent on the presence of dissonances *per se*, but also upon conventional association. This means that depending on the conventions of the musical style, dissonances might be more or less acceptable to the listener, and so can arouse different affective reactions. A study of listening preferences for infants, conducted by Trainor and Heinmiller [197], shows that even these young listeners, with no knowledge of musical scale, have an affective preference for consonance.

It is important to notice that, while dissonances do occur in music constrained to a certain key, e.g. the Vth chord is the most dissonant and so most tension-building. We only consider the introduction of out-of-key notes to introduce a higher degree of dissonance in the music. We connect this feature to valence, hypothesizing that introducing more and more dissonances creates a more negative affect expression.

Out-of-key notes are introduced with the hypothesis that they create more negative valence. The more dissonances added, the more negative the valence. To maintain a feeling of purposeful composition, the possible notes that can appear correspond to the alteration of different scales, going further and further away from the Ionian mode (major mode) according to western music theory (this idea was inspired by Husain's study on mode's effects on mood [198]). It is important to note that METACOMPOSE does not perform proper mode changes; they are unprepared and the harmonic framework stays in a major key. Assuming a tonic of C, new notes that can appear in the piece are (from higher to lower valence):

- $B\flat$, from the Myxolydian mode
- $B\flat$ and $E\flat$, from the Dorian mode
- $B\flat$, $E\flat$ and $A\flat$, from the Aeolian mode
- $B\flat$, $E\flat$, $A\flat$ and $D\flat$, from the Phrygian mode

The reader might notice we are missing the Lydian and Locrian modes; we decided to exclude these scales for two different reasons. The Lydian mode is the same as the Ionian mode with its fourth degree raised, but it introduces an interval not as easily built upon to reach other modes (i.e. it requires the removal of the alteration to the fourth before adding new alterations). The Locrian mode is defined as a diminished mode as, although its third scale degree is minor, the fifth degree is diminished, instead of perfect. We have excluded it from our listening study mostly because we felt the music produced lost quality, as the diminished intervals make the piece sound less musically structured.

5.3 MetaCompose Archive

The archive's purpose is to store the previously generated compositions and associate these with game-related information, such as: levels, events or even entities. Another main function of the archive is the implementation of a distance measure that can give us information on how similar two compositions are. Such a measure helps manage diversity via similarity in previous and new tracks.

The distance measure we chose is the *Tonal Pitch Step Distance* [199], which measures distance of chord progressions on the basis of harmonic similarity. It uses a variant of Lerdahl's Tonal Pitch Space [200] to measure the distance of a chord to the tonic triad of its key. We have chosen this particular measure, even though it only considers the chord sequence part of our compositions, because we believe that much of the recognition of a song depends on the similarity of the chord sequence. Of course we realize that melodies are also very important, they are easy to recall and we generally connect them to a song, yet we believe that if you listen to the same melody, on two different chord sequences, it is not as immediately recognizable. On the other hand, the opposite is easier. In the future we might expand the distance measure to also include differences in the melody, but we decided it was not necessary for this first implementation of the system.

The main reason to have this mechanism is to allow the game-designer to direct METACOMPOSE to create music that the game-player would regard as consistent with the content presented to her previously. A case in which this might be useful is: if an important game-event occurs, which the game-designer wants to underline with a new composition and not just with a change in the affective meaning expression, she might want a new composition to share some similarity with the one previous heard and archived (e.g. the player might be in the same game-play level). As an example of the opposite, imagine the player moving from one area to another that is very different to anything previously seen. The game-designer may want to reflect this transition (at least partly) through music, so the game-designer directs METACOMPOSE to generate a track with a high distance score from any previously archived.

Chapter 6

Evaluation of MetaCompose

This chapter describes three studies we have conducted to evaluate different aspects of METACOMPOSE. Section 6.1 describes an evaluation of the system, trying to find out how the different parts of METACOMPOSE interact with each other and how they affect the music generated. As already shown in section 4.4, valence seemed to be harder to express reliably. As such we designed and conducted an experiment focusing on the way we introduced dissonance in the pieces to be sure that our assumptions led to correctly perceived valence (section 6.2). Finally, in section 6.3 we present the most complete study on the affective perception of the music generated by the complete system. Whenever possible we have used ranking in these studies, as Martinez and Yannakakis [201] suggest that ranking produces more consistent and reliable data when annotating affect information. Therefore we often chose to ask participants to compare two pieces of music, or two different experiment setups.

6.1 Evaluation of the generation technique

We subjected METACOMPOSE to a series of extensive quantitative studies in order to validate our music generation approach. The main objective of the study is to investigate the contribution of each component of the framework to the perceived quality of the music created. To do this, METACOMPOSE components were systematically switched off and replaced with random generation. From these random "broken" generators, and the complete METACOMPOSE system, we created various pair-wise samples to test against each other[1]. As the quality of music is a subjective matter, participants are asked to prefer one of two pieces of music presented to them, one generated by the complete algorithm and one from a "broken" generator with one component replaced with random generation. Quality is evaluated according to four criteria: *pleasantness*, *randomness*, *harmoniousness* and *interestingness*. These four criteria present a good overview of the preference expressed by the participant. Note that no definition of these terms is offered in the survey, and there is therefore no

[1]This method is inspired by the "ablation studies" performed by Stanley [176].

guarantee that participants interpret these criteria the same way (or for that matter differently).

Pleasantness intends to measure how pleasing to the ear the piece is, but this alone is not sufficient to describe the quality of the music produced. There are countless pieces of music that do not sound pleasant, but may nonetheless be considered by the listener as "good" music. In fact, in music, often uncommon (and even discordant) chord sequences or intervals are introduced to express different features in a score, such as affect, as well as other narrative information. Also note that some alterations or passages can be specific to a music style. Moreover, discordant intervals are more acceptable to the ear the more often they are repeated (see dodecaphonic music [202] for example).

Interestingness is a criterion introduced to overcome the just described limitations of the *pleasantness* criterion: in this way we intend to test if one of our "broken" scores might introduce something that results in something considered interesting to the listener, even when the composition is not as pleasant or harmonic. Note that this is a very subjective measure, as most people have a different opinion about how interesting they perceive a score to be.

On the other hand, *harmoniousness* might be confused with pleasantness, but we hope that it will be seen as a somewhat more objective measure: less of a personal preference and more of a measure of the listener's ability to recognize the presence of dissonances and harmonic passages.

Finally, *randomness* intends to gather a measure of how structured the music appears to the listener. It is not only a measure of dissonance (or voices being off-key), but also of how much the music seems to have a cohesive quality and coherent internal structure. Examples of coherent internal structure are: (i) voices working together well (ii) coherent rhythmic structure (iii) chord sequence presenting tension building and eventual resolution.

An online survey was developed with HTML and PHP, using a MySQL database to hold the data collected. Participants were presented with paired music clips and asked to evaluate them using the four criteria described. Each of the four criteria has a multiple choices question structured as:

> **Which piece do you find more pleasing?** *"Clip A"/"Clip B"/"Neither"/ "Both Equally"*,

where the last word (e.g. "pleasing") is dependent on the criterion. We also include the more neutral answers "Neither" and "Both Equally" to avoid randomness in the data from participants who cannot decide which clip satisfies their evaluation according to the criterion better or worse. Other benefits of doing this are: avoiding the participant getting frustrated, and giving us some possible information on

interesting individual pairs, where the pieces are considered equally good/bad.

Note that, for the first five questions in the survey, the paired clips always included one clip from the complete generator. After five trials, the clip pairs are picked at random between all the groups. In this way, we hoped to collect enough data to be able to make some observations between the four "broken" generators. The motivation behind this survey design choice is that our main question is evaluating the complete generator against all possible alternatives, so attention to the complete METACOMPOSE architectural has priority. This also has a practical justification in the fact that, with the number of groups we have (five), testing all possible combinations and gathering enough data would be practically impossible. The survey has no pre-defined end: the user is able to continue answering until he/she wants, and can close the online survey at any time or navigate away from it without data loss. However, in the preamble to the survey, we encouraged participants to perform at least five comparisons.

6.1.1 Music clip generation

The five groups were examined (as dictated by the architecture of METACOMPOSE):

A. Complete generator: the complete composition generator, METACOMPOSE, as described in Section 5.1;

B. Random chord sequence: the chord sequence module is removed and replaced with a random selection of chords;

C. Random unconstrained melody: the melody generation evolutionary algorithm is replaced with a random selection between all possible notes in the melody range (two octaves);

D. Random constrained melody: the melody generation evolutionary algorithm is replaced with a random selection between all possible notes belonging to the key of the piece in the melody range (two octaves). We decided this was necessary as (by design) our melody evolution approach is restricted to a diatonic context;

E. Random accompaniment: the accompaniment generation is replaced by a random accompaniment abstraction (we remind the reader that an accompaniment abstraction is defined by a basic rhythm and note sequence).

TABLE 6.1: Number of correct, incorrect and neutral answers to our criteria for the **complete generator** (A) against all the **"broken" generators** (B-E), combined. Note that in the case of the random criterion the listener is asked to select the clip that he/she feels the most random, so it is entirely expected that a low number of participants choose the **random clip** (E) against the **complete generator** (A).

Choice	Pleasing	Random	Harmonious	Interesting
METACOMPOSE (A)	654	197	671	482
Choose a "broken" generator (B-E)	240	633	199	327
A neutral answer	197	261	221	282
Total non-neutral answers	894	830	870	809
Binomial test p-value	7.44E-21	2.75E-77	2.05E-29	7.81E-02

For each of these 5 groups, 10 pieces of music are created. For the sake of this experiment the affect expression has been kept to a neutral state for all the groups and we used the same algorithms to improvise on the composition abstraction. There is therefore no exploration of the music generators' affect expression but rather an evaluation of the music quality from the complete architecture compared to the architectural alternatives. The clips for the various groups can be accessed at http://msci.itu.dk/evaluationClips/

6.1.2 Results and analysis

The data collected amounts to 1,291 answers for each of the four evaluation criteria from 298 participants. Of the survey trials generated, 1,248 contained a clip generated with METACOMPOSE (A). Table 6.1 shows how many responses were obtained for each criterion and how many neutral answers were collected.

For now we only consider definitive answers (where a participant chooses one of the music clips presented), we examine the impact of the neutral answers at the end of this section. Under this constraint, the data becomes Boolean: answers are either *"user chooses the clip from the complete generator (A)"* or *"user chose the clip from a broken generator (B-E)"*. To analyse this data we use a two-tailed binomial test, which is an exact test of the statistical significance of deviations from a theoretically expected random distribution of observations in two categories. The null hypothesis is that both categories are equally likely to occur and, as we have only two possible outcomes, that probability is 0.5.

TABLE 6.2: Answers and results of the binomial test for pairs comprised of the **full generator**, METACOMPOSE (A), and the one with **random chord sequences** (B).

METACOMPOSE (A) vs (B)	**Pleasing**	**Random**	**Harmonious**	**Interesting**
Successes	121	71	112	98
Failures	93	117	84	98
Totals	214	188	196	196
Binomial test p-value	3.23E-02	4.90E-04	2.68E-02	5.28E-01

6.1.3 Complete Generator against all other groups

Firstly, let us consider the combined results of all the "broken" groups (B-D) against METACOMPOSE (A): as can be seen from Tab. 6.1, we have statistically highly significant differences for the *pleasing*, *random* and *harmonious* categories, while we have a p-value of 0.078 for the *interesting* category. This means that we can refute the null hypothesis and infer a difference in distribution between choosing the music generated by the complete algorithm (A) and the "broken" ones (B-E).

We can affirm that METACOMPOSE (A) ranked better than all the others (B-E) for three of our four criteria, with the exception of *interestingness*, where there is no statistically significant difference. Interestingness is clearly a very subjective measure, and this may explain the result. Moreover, examining the ratio of neutral answers obtained for this criterion, it can be observed that it is almost 26%, a much higher neutral response than for the other criteria. This shows that in a higher number of cases participants could not say which composition they found more interesting. A possible explanation is that, as the affect expression (which also includes musical features such as tempo and intensity) is held in a neutral state, equal for all pieces, after hearing a number of clips listeners does not find much to surprise them. Also the duration of the generated pieces (ca. 30 seconds) might not allow sufficient time to determine interestingness.

6.1.4 Complete Generator against random chord sequence generation

If we only consider the pairs that included the METACOMPOSE (A) and the one with random chord sequences (B) (Tab. 6.2) we, again, obtain statistically significant differences in the distribution of the answers for the *pleasing*, *random* and *harmonious* criteria. In this case we have a very high p-value for *interestingness* (more than 0.5), in fact we have the same degree of preference for the METACOMPOSE (A) and the "broken" generator (B). We can explain this by considering

TABLE 6.3: Answers and results of the binomial test for pairs comprised of the **full generator**, METACOMPOSE (A) and the one with **unconstrained random melody** (C).

METACOMPOSE (A) vs (C)	**Pleasing**	**Random**	**Harmonious**	**Interesting**
Successes	221	21	236	144
Failures	26	221	19	72
Totals	247	242	255	216
Binomial test p-value	5.15E-40	1.44E-43	4.11E-49	5.46E-07

that the disruptive element introduced by this modification of METACOMPOSE is mitigated by the fact that the rest of the system tries to create as pleasing music as it can, based on the chord sequence produced. So, for most of the time, the music will not have notes that sound out of key or that do not fit well with the chord sequence. Still, we observe how the listener is capable of identifying that, while the piece does not sound discordant or dissonant, it lacks the structure of tension-building and tension-releasing. This explains how METACOMPOSE (A) is preferred for all other criteria. It is interesting to note how the act itself of presenting the listener with uncommon chord sequences does result in an increase of the interestingness of the music.

6.1.5 Complete Generator against unconstrained melody generation

When we consider the unconstrained melody group we have statistically significant differences for all criteria, with some extremely strong significance (Tab. 6.3). These results are as we expected, as the melody plays random notes that conflict with both the chord sequence and the accompaniment.

6.1.6 Complete Generator against constrained melody generation

The results given by the constrained random melody generation (D) are more interesting (Tab. 6.4). First, we notice no statistically significant values for the *pleasing* and *interesting* criteria. This is explained by the fact that the melody never goes off key, so it never presents off-key notes and never sounds abruptly "wrong" to the listener's ear. Yet, the *random* and *harmonious* criteria are statistically significant. Remembering how we described these criteria, we notice that the more objective criteria (*random* and *harmonious*) are those that demonstrate a difference in distribution. We believe this reinforces

TABLE 6.4: Answers and results of the binomial test for pairs comprised of the **full generator** METACOMPOSE (A), and the one with **constrained random melody** (D).

METACOMPOSE (A) vs (D)	**Pleasing**	**Random**	**Harmonious**	**Interesting**
Successes	125	81	120	108
Failures	100	109	85	94
Totals	225	190	205	202
Binomial test p-value	5.47E-02	2.49E-02	8.68E-03	1.80E-01

TABLE 6.5: Answers and results of the binomial test for pairs comprised of the **full generator**, METACOMPOSE (A), and the one with **random accompaniment** (E).

METACOMPOSE (A) vs (E)	**Pleasing**	**Random**	**Harmonious**	**Interesting**
Successes	188	25	203	132
Failures	21	186	12	63
Totals	209	211	215	195
Binomial test p-value	5.00E-35	6.58E-32	3.01E-46	4.35E-07

how, although compositions made in this group never achieve a bad result, the listener is still able to identify the lack of structure (randomness) and lack of consideration of the underlying chords of the melody (harmoniousness). An example of the first case would be a melody that jumps a lot between very different registers; this would make the melody sound more random than the melodies we evolve using (A) – METACOMPOSE – which follow more closely the guidelines of a singing voice. Harmoniousness can be influenced by the fact that, over a chord (expressed by the accompaniment), the melody can play notes that create intervals that 'confuse' the clarity of the chord to the listener's ear.

6.1.7 Complete Generator against random accompaniment generation

Finally, for the last group, the random accompaniment generation (E), gives us very clear statistically significant results on all criteria (Table 6.5). A lot of the harmony expression depends on the accompaniment generation, and when this is randomized it is no wonder that the piece sounds confusing and discordant. This is reflected in the trial data.

6.1.8 Conclusions

Returning to the main question of the experiment – *do all parts of the music generation system add to the music produced?* – we performed an extensive quantitative study to validate our music generation approach. The main objective was to investigate the contribution of each component of the framework to the quality of the music created. To do this, we systematically switched off components of our generator and replaced them with random generation. From these random "broken" compositions and the complete algorithm we created various pair-wise samples to test against each other (this method was inspired by the "ablation studies" performed by e.g. Stanley [176]). We have described an evaluation in which we created music with our generator substituting various components with randomized generators. In particular we observed four broken groups: *random chord sequences*, *random melody constrained* (to the key of the piece), *random melody unconstrained* and *random accompaniment*. An evaluation of the music clips generated by these "broken" versions was compared to music clips created by the complete algorithm according to four criteria: *pleasantness*, *randomness*, *harmoniousness* and *interestingness*.

Analysis of the data supports the assertion that participants prefer the complete system in three of the four criteria: (*pleasantness*, *randomness* and *harmoniousness*) to the alternatives offered. The results for the *interestingness* criteria are however not definitive, but suggest that some parts of our generator have a higher impact in this criteria. The results for the "interestingness" criteria are particularly thought-provoking: from our data, it appears that random chord sequences and random constrained melodies can offer a similar degree of this criteria compared with the complete method. Probable explanations for this phenomenon are the introduction of surprise (unexpected chord sequences and melody progressions), a key factor in "interestingness". We can also observe a higher ratio of neutral answers for the "interestingness" criterion: possible explanations of these are: participants might not really be sure about what they think is interesting, or that the pieces were not particularly different from each other to the listener's ear. This might also be caused by a potential high-threshold for interestingness in the participants.

6.2 Evaluation of valence expression through dissonance

An evaluation conducted on an earlier prototype of a mood expressive music generator indicates that our mood expression theory better expresses arousal than valence (positive/negative feelings) [203]. Therefore, we are conducting a series of user studies to learn, in more depth, what the effect of the features we believe influence valence

actually is. We present the results for an evaluation that focuses on the introduction of dissonant intervals by means of altered tones (i.e. notes that are not included in the prevailing tonality, from now on also referred as out-of-key notes for brevity) and in particular how it affects valence in algorithmically generated music. The research questions this study addresses are:

1. Can negative valence be expressed via the introduction of altered tones in the generated music?

2. Can the quality of generated music be maintained when such notes are added?

To this end, we present and discuss the results of a participant-based evaluation study.

Traynor [197] shows how infants prefer consonant intervals to dissonant ones, yet there is a significant difference between instinctive preference and perceived negative valence. Therefore the connection between dissonance and positive/negative valence in METACOMPOSE is recorded and measured via an experimental platform where users listen to music generated by variations of an algorithm. Statistical analysis of user preferences is performed to characterize the differences between the generative clips in which examples of dissonant and consonant music are presented.

6.2.1 Experiment design

The main objective of this study is the evaluation of the valence expression from the introduction of altered tones in METACOMPOSE. The secondary objective is the maintenance of perceived music quality under these circumstances. An experiment was designed where paired samples were tested against each other.

The survey asked participants to prefer one of two pieces of music presented and evaluate them according to three criteria: *most negative feeling expressed*, *most well-composed* and *most interesting*. The first criterion is the one to answer for our main research question: can negative valence by expressed by the introduction of altered tones? The other two ratings are more related to the secondary question: can we maintain the quality of the music while introducing altered tones? We expected lower preference in the "well-composed" criterion for the *Out-of-key* group, as it can introduce intervals that might be unpleasant (although this might be subjective and sensitive to the cultural background of the individual). The "interestingness" criterion we expected to be mostly balanced, to show how the music with altered tones can still be interesting to listen to.

Each criterion has a multiple choices question structured as:

> **Which piece seems to express more negative feelings?**
> *"Clip A"/"Clip B"/"Neither"/ "Both Equally"*,

where the wording is dependent on the three criteria. As in the study detailed in the previous section, we also include the more neutral answers "Neither" and "Both Equally" to avoid randomness in the data from participants who cannot decide which clip satisfies the evaluation criterion better or worse. The survey consists of ten questions, where the two clips presented are always one from the *consonant* and one from *dissonant* groups. The music pieces are chosen in a way that the participant will listen to all the clips produced for the experiment, but with random pairings between the two groups.

6.2.2 Music clip generation

Ten clips were created for each group (*consonant* and *dissonant*), for a total of 20 pieces. For the sake of this experiment the affect expression has been kept to a neutral state for all the mood-expressive features, apart from the *dissonances* feature. The same algorithms have been used to improvise on the composition abstraction. The clips for the various groups can be accessed at http://msci.itu.dk/dissonanceClips/.

TABLE 6.6: Participant's answers to our criteria. Also shown are the p-values, calculated using a two-tailed binomial test, and the Binomial Effect Size Display.

Choice	Most negative	Most well-composed	Most interesting
Altered group	329	118	191
Diatonic group	95	305	233
Neutral answer	112	113	112
Total not neutral answers	424	423	424
Binomial test p-value	1.38E-31	1.81E-20	2.32E-02
BESD	55.2%	-44.1%	-9.9%

6.2.3 Results and analysis

The data collected amounts to a total of 536 answers for each of the three evaluation criteria from 110 participants. Table 6.6 shows how many responses were obtained for each criterion and how many neutral answers were collected.

For now we only consider definitive answers (i.e. the participant chooses one of the music clips presented); we will look at the impact of the neutral answers at the end of this section. As for the previous study, under the definite choice constraint, the data becomes Boolean: the answers are either *"user chose the clip from the Out-of-key group"* or *"user chose the clip from the Diatonic group"*. To analysethis data we use a two-tailed binomial test, with as null hypothesis that both categories are equally likely to occur and, as we have only two

possible outcomes, that probability is 0.5. The Binomial Effect Size Display (BESD) [204] is another way of looking on the effects of treatments by considering the increase of success through interventions. This is an interesting measure, as it elucidates how much of an effect is created, in our case, by the introduction of altered tones.

As can be seen in Table 6.6, there is a strong statistical significance for the *most negative feeling*, *most well-composed* and *most interesting* categories. Thus the null hypothesis can be refuted and a difference in distribution can be inferred between choosing the music generated with (and without) the introduction of altered tones. This means that our system's introduction of out-of-key notes expresses more negative valence at the price of being perceived as less well-composed and less interesting. The BESD values reflect what can be inferred from the p-values, yet for the *most interesting* criterion the effect is much smaller (-9.9%), leading us to conclude that not as much interestingness is lost as the statistically significant p-value (2.32E-02) might suggest.

Outlier in dissonant group

An outlier was found by looking at the preference expressed for each of the *Out-of-key* music clips (see Figure 6.1). This specific clip has a very different distribution of answers than other *Out-of-key* clips with a much lower probability of being selected as more negative, a higher chance of being selected as more well-composed and a higher chance of being selected as more interesting. The piece is in $B\flat$ major, which would be composed of $B\flat$, C, D, $E\flat$, F, G and A. According to the system previously described, the piece should have its second, third, sixth and seventh degree altered by lowering them by a semitone, leading us to this scale: $B\flat$, $C\flat$, $D\flat$, $E\flat$, F, $G\flat$ and $A\flat$. Yet, we notice from the score that $C\flat$ and $G\flat$ never appear in the piece. This effectively removes all the strongest dissonances from the piece: those formed by notes distant by a semitone ($B\flat$-$C\flat$ and F-$G\flat$). This is a chance event, formed by a combination of both the way the *composition* was formed and the way the instruments have improvised over the abstraction.

By removing the data obtained by questions in which this clip appeared, we notice a slight increase in significance for all three criteria (in the order of 10^{-1}). This is expected, as it reinforces the distribution of the data we observed while considering all out-of-key samples.

6.2.4 Demographics

Our participant's population is composed of 89 males and 11 females. The average age is 29.5 (stdev 13.8). Participants were asked to rate their skill with a music instrument and their knowledge of music theory according to a five point Likert scale. The participants

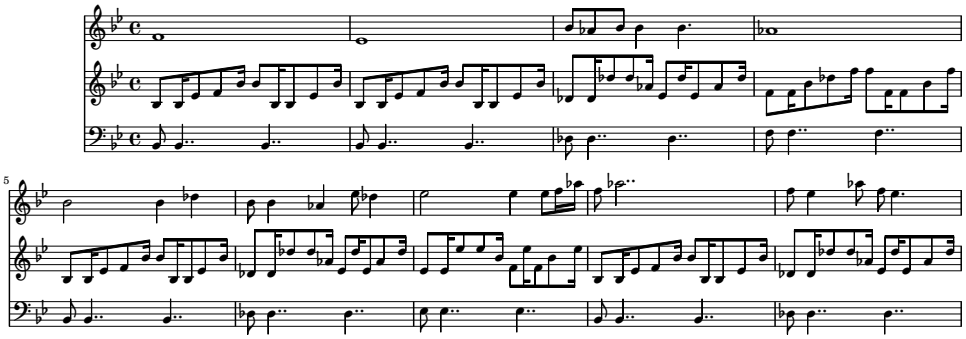

FIGURE 6.1: Score for the outlier piece in the *out-of-key* group.

have reported a similar level of musical training (avg: 1.1 stdev: 0.97) and instrument skill (avg: 1.4 stdev: 1.2). The homogeneity of the population may explain how, however we partition the population, we find no statistically significant difference in the answers given. We observed two participants that gave a high percentage of neutral answers (>75%). These participants self-reported to have close to no training and experience with music instruments. Yet the very low incidence of participants in this group makes it hard to make any assumptions, especially as many other participants reported a similar level of skill gave many definite choice answers.

If the population is divided by gender, we observe a higher preference in interestingness for dissonant pieces in males, yet the overall low number of female participants makes any conclusion here statistically unreliable.

6.3 Evaluation of the affective expression of MetaCompose

6.3.1 Experiment design

The main objective of this study is the evaluation of the affective expression in the music produced by METACOMPOSE. A secondary objective is evaluating in real-time changes in valence in order to better understand what music characteristics influence the listener's perception.

An experiment was designed where participants, while listening to a piece of generated music, would annotate changes in valence via manipulating an annotation wheel. By "annotation wheel" we mean a physical knob that the participants could turn clockwise to indicate

an increase in valence and counter clockwise for a decrease[2]. The annotation was conducted using software written by Phil Lopez[3] (and inspired by the work of Clerico *et al.* in annotating fun [205]) with the use of a Griffin Technology PowerMate programmable controller. Afterward participants were tasked with annotating the mood perceived at the start and end of the piece and provide an overall assessment of the music quality.

The questions asked were all in the form of 5-point Likert scales:

- How would you rate the quality of the music you just listened to? *Very low/Somewhat low/Moderate/Somewhat high/Very high*
- How positive/negative was the music at the beginning of the piece? *Very negative/Somewhat negative/Neither negative nor positive/Somewhat positive/Very positive*
- How tense/calm was the music at the beginning of the piece? *Very calm/Somewhat calm/Neither calm nor tense/ Somewhat tense/Very tense*

The last two questions are duplicated for the end of the piece.

A survey was developed with HTML and PHP, using a MySQL database to hold the data collected. The real-time annotation tool is a *C#* program which uses VideoLan's VLC to play the musical clips. The PHP code invokes the annotation tool through the *exec()* function, which effectively stops the execution of the PHP until the annotation terminates.

The experiment was designed to present the participants with 10 randomly chosen music clips (5 static and 5 with a transition, repetitions of the same piece were not allowed). As each clip has length of one minute the experiment was designed to last between 15 and 20 minutes for each participant.

6.3.2 Music clip generation

For the purpose of this experiment 19 music clips were generated using METACOMPOSE: 10 that exhibited a transition in affective expression, and 9 that did not. Of the 10 pieces with transitions: 2 present large changes in only one dimension, 4 present smaller changes in only one dimension, and the remaining 4 present a combination of changes in both valence and arousal (see Figure 6.2). The music clips are one minute long and the transitions occur half-way through the clip. The pieces themselves are synthesized using Java's MIDI synthesis, the current default method for METACOMPOSE.

[2]We would like to offer our special thanks to Professor Georgios Yannakakis and Phil Lopez for the discussions that led to the design of this experiment, and for putting to our disposal the real time annotation tool used in this study.

[3]https://github.com/WorshipCookies/RealTimeAnnotation

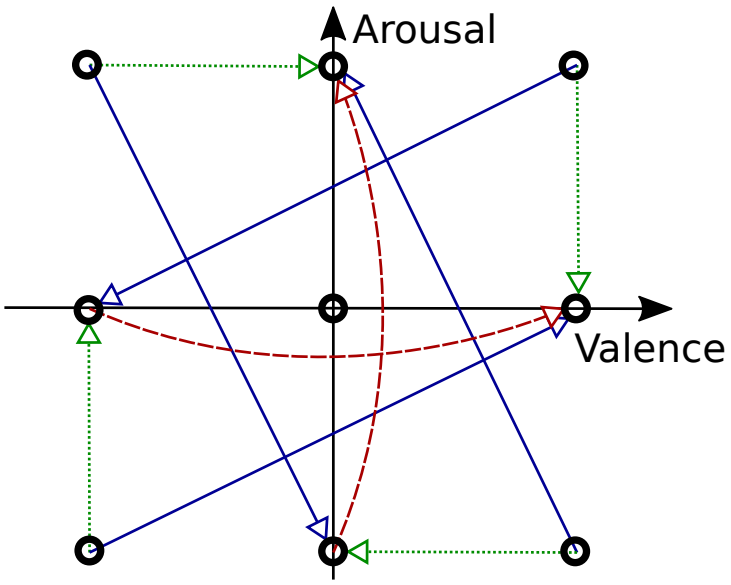

FIGURE 6.2: Visual representation of the mood expression of the generated clips: in red/dashed, the 2 large mono-dimensional transitions; in green/dotted, the 4 small mono-dimensional transitions; in blue/solid, the bi-dimensional transitions. Vertices represent the affective expression of the static clips. A list of which clips correspond to each transition can be accessed at http://msci.itu.dk/gecco/clip_list.txt.

6.3.3 Experiment setup

Two computers were used for the experiment, with identical setup of software (HTML+PHP survey running locally on an Apache webserver) and hardware (Sony headphones and Griffin Technology PowerMate controllers). The volume of the computer audio was adjusted beforehand to the same level on each PC. All tests were conducted in the meeting rooms at the IT University of Copenhagen, which present comparable levels of light and room layout.

6.3.4 Results and analysis

The data collected corresponds to 200 answers and real-time annotations, from 20 participants. Recall that each participant was presented with a randomized selection of 5 music clips (from a possible 10) containing a transition in expressed mood state and 5 clips with static mood expression (from a possible 9). The clips were also presented in random order.

TABLE 6.7: Variations in arousal and valence from survey. Given the categorical nature of the data we include variation in average and mode. The possible answers are on a 5-point Likert scale (range 0-4), transition data is the difference in how participants annotated the affective expression at the start and end of clips. Clips 0-9 present a transition in affective expression, clips 10-18 are static.

Clip No.	Valence average variation	Arousal average variation	Valence mode variation	Arousal mode variation
0	-0.444	-0.556	-1	0
1	2	-2.833	3	-4
2	0.538	-0.154	0	0
3	0.25	1	2	2
4	1.615	-0.154	2	1
5	0.625	2.125	0	3
6	-1.125	-1.875	-2	-1
7	-1	-1.417	-2	-2
8	1.25	1.75	1	3
9	-0.545	-0.182	0	0
10	0.111	0.333	0	0
11	0	-0.333	0	0
12	0	0	0	0
13	0.231	-0.077	0	0
14	0	0.071	0	0
15	0.111	-0.111	0	-1
16	0.111	-0.222	0	1
17	0	0.214	-1	0
18	0.25	-0.125	0	0

TABLE 6.8: Valence, raw answers contingency table. Shows how many times an answer was chosen in respect of the intended valence expression.

Intended/Chosen	0	1	2	3	4
Negative	15	61	34	27	2
Neutral	5	22	27	50	16
Positive	5	21	38	58	19

6.3.5 Survey analysis

Transition perception

Table 6.7 shows the differences in the annotations the participants provided for the start and end of the clips. The clips that presented a static mood-state (clips 10-18) present little variation in annotation. In the *transition* group, two clips have been labelled as having almost no perceivable change in expression (clips 2 and 9). Both these clips have no change in arousal (this seems to align with the results to be discussed in Section 6.3.6). Furthermore, the average variation in valence in these two cases is higher than any of the variations observed in the *static* group, leading us to hypothesize that listeners can indeed perceive variations in affective expression.

It is important to notice however, that while most perceived transitions reflect what would be expected based on the generator parameters, there are three notable exceptions in annotating valence. In clip 3, a transition to a more positive mood has been annotated, while the clip would have been expected to maintain the same valence; in this case it is noteworthy that while the variation in mode makes it seem like a very strong misclassification (+2), the variation in average scores present a much better score (+0.25). Clip 7 shows a decrease in valence where there would be expected to be none, and clip 8 shows an increase in valence where there would rather be expected to be a small decrease. All of these cases connect to, and find a possible explanation, in the results and discussion that follow in Section 6.3.6.

Valence analysis

The raw answers given by the participants can be represented in categorical values from 0 to 4 (answers on a Likert scale). Observing the contingency Table 6.8, it can be observed that there is only a small variation in how clips, that should express neutral and positive valence, are categorized by the participants. Performing a χ^2 test of independence on this data returns a p-value of $2.822e^{-10}$ ($\chi^2 = 61.11, \nu = 8$), so the null hypothesis that the annotations are independent from the expressed valence can be rejected. A series of tests has

TABLE 6.9: Valence contingency table, shows how many times an answer was chosen with what was intended. In this case the answers identifying a negative/positive valence are grouped, no matter the perceived intensity, creating three possible answers: positive, negative and neutral.

Intended\Chosen	Negative	Neutral	Positive
Negative	76	34	29
Neutral	27	27	66
Positive	26	38	77

been conducted on each coupled valence-expression of this experiment to test the independence of the answers' distributions.

Negative vs Neutral Fisher's exact test: $p = 1.188e^{-08}$. Chi-squared $p = 3.082e^{-08}$ ($\chi^2 = 40.713, \nu = 4$)

Neutral vs Positive Fisher's exact test: $p = 0.9039$. Chi-squared $p = 0.9019$ ($\chi^2 = 1.0517, \nu = 4$)

Negative vs Positive Fisher's exact test: $p = 8.69e^{-11}$. Chi-squared $p = 3.994e^{-10}$ ($\chi^2 = 49.79, \nu = 4$)

Because very small numbers appear in Table 6.8 χ^2 might not be producing precise estimates of the p-value. To check the correctness of the results a categorization of {Positive, Neutral, Negative} is achieved (Table 6.9) by grouping the "somewhat positive/negative" and "very positive/negative" answers. Although this removes some of the answers' granularity, repeating the same tests as before, chi-squared test of independence on this data returns a p-value of $6.419e^{-12}$ ($\chi^2 = 58.358, \nu = 4$). Performing the tests on the coupled data we obtain:

Negative vs Neutral Fisher's exact test: $p = 5.264e^{-09}$. Chi-squared $p = 7.826e^{-09}$ ($\chi^2 = 37.332, \nu = 2$)

Neutral vs Positive Fisher's exact test: $p = 0.5992$. Chi-squared $p = 0.5934$ ($\chi^2 = 1.0437, \nu = 2$)

Negative vs Positive Fisher's exact test: $p = 3.79e^{-11}$. Chi-squared $p = 8.17e^{-11}$ ($\chi^2 = 46.456, \nu = 2$)

While we have a very strong statistical significance between Negative valence and the other two levels, the Neutral and Positive levels appear too similar to consistently distinguish between them.

Arousal analysis

As with valence, a contingency table can be created showing how the participants rated the arousal present in the pieces (Table 6.10). This

TABLE 6.10: Arousal raw answers contingency table. Shows how many times an answer was chosen in respect of the intended arousal expression.

Intended\Chosen	0	1	2	3	4
Low	78	41	11	12	5
Neutral	23	41	54	10	1
High	11	23	31	54	5

time a clear difference between the distributions emerges. Applying the chi-squared test a p-value of $2.2e^{-16}$ ($\chi^2 = 152.11, \nu = 8$) can be calculated, which sustains the hypothesis that the answers are not independent of the expressed arousal. Performing the tests on the coupled arousal-expressions we obtain:

Low vs Neutral Fisher's exact test: $p = 1.506e^{-13}$. Chi-squared $p = 2.475e^{-12}$ ($\chi^2 = 60.328, \nu = 4$)

Neutral vs High Fisher's exact test: $p = 1.149e^{-10}$ Chi-squared $p = 7.947e^{-10}$ ($\chi^2 = 48.358, \nu = 4$)

Low vs High [4] Chi-squared $p = 2.2e^{-16}$ ($\chi^2 = 90.451, \nu = 4$)

Again, small numbers can be found in Table 6.10, so the "slightly tense/calm" and "very tense/calm" are combined to obtain Table 6.11. With Chi-squared a p-value smaller than $2.2e^{-16}$ ($\chi^2 = 125.61, \nu = 4$), consistent with the previous result. Performing the same tests on the coupled data we obtain:

Low vs Neutral Fisher's exact test: $p = 3.386e^{-11}$. Chi-squared $p = 1.47e^{-10}$ ($\chi^2 = 45.281, \nu = 2$)

Neutral vs High Fisher's exact test: $p = 7.894^{-12}$. Chi-squared $p = 3.346^{-11}$ ($\chi^2 = 48.242, \nu = 2$)

Low vs High Fisher's exact test: $p < 2.2e^{-16}$. Chi-squared $p < 2.2e^{-16}$ ($\chi^2 = 78.57, \nu = 2$)

A statistically significant difference of the participants' answer given the three arousal levels can be shown for each of the groups, moreover by looking at the answers distributions we can confirm that the arousal levels are perceived as expected. Still we notice that there seems to be a bias towards low arousal.

6.3.6 Real-time annotation

The data recorded with the real-time annotation tool consists of a score representing how much higher/lower people are rating the valence of the clip from the original (the valence at the start of the clip)[5].

[4] Fisher's exact test couldn't be calculated because of a lack of memory

[5] The raw data can be accessed at http://msci.itu.dk/gecco/alllogs.zip.

TABLE 6.11: Arousal contingency table showing how many times an answer was chosen with respect to what we intended. In this case, answers that identify a calm/tense arousal are grouped no matter the perceived intensity, creating three possible answers: high, low and neutral.

Intended\Chosen	Low	Neutral	High
Low	119	11	17
Neutral	64	54	11
High	34	31	59

As there is no limit to how high/low people could score changes, each raw log is pre-processed with min-max normalization. This way each of the measurements will range between 0-1 and the new data will account for personal perception of changes (e.g. one participant might annotate each change with a double-value scale compared to another participant). Finally, for each clip the average and standard deviation has been calculated to obtain the graphs that can be seen in Figure 6.3[6].

The first thing that can be noticed is that the clips that present transitions in mood expression, present a change in affect trend halfway through the clip (where the change in affective expression happens). Yet in some cases (clips 3, 5 and 8) we observe an increase in valence which should not be there. This increase in participant-observed valence is accompanied by an increase in expressed arousal, which might suggest that one or more of the features that we associate with arousal has an effect on valence as well. Interestingly, clip 7, which should not present any change in arousal, shows a very small negative transition which might correspond with the decrease in expressed arousal. Yet the high standard deviation in observed data present throughout the piece, is not as indicative of misclassification of arousal as the previously mentioned clips.

6.3.7 Demographics

From the 20 users that participated in the experiment, 14 are males, 5 females, and 1 participant did not express gender. The participants' age has an average of 27.2 years ($stdev \approx 6.3$). In regards to the other demographic answers, expressed in 5-point Likert scale (0–4), most people self-reported very little experience with playing an instrument ($avg = 1.2$, $stdev \approx 1.2$, $mode = 0$), very little knowledge of music theory ($avg = 1.1$, $stdev \approx 1.1$, $mode = 0$), and a considerable experience with video-games ($avg = 2.5$, $stdev \approx 1.27$, $mode = 3$). No

[6]the complete set can be accessed at http://msci.itu.dk/gecco/graphs.zip

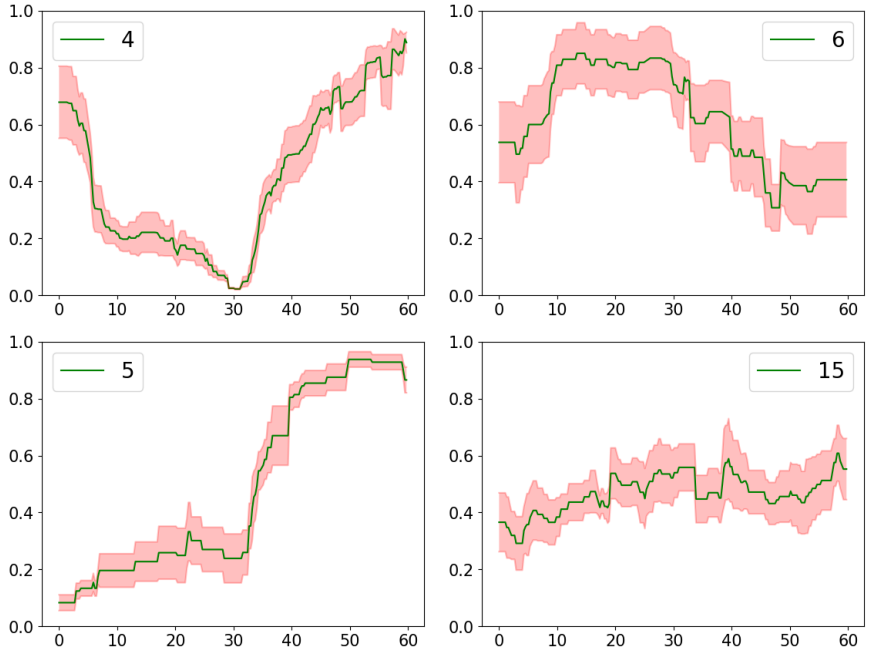

FIGURE 6.3: Examples of averaged real-time annotation for valence, the standard deviation is displayed as the red zone, the complete set can be accessed at http://msci.itu.dk/gecco/graphs.zip. These showcase the main types of annotation that can be observed from the data: e.g., clip 4 presents a correctly annotated clip, an increase in valence halfway through the clip, clip 6 presents a correctly annotated decrease in valence, clip 5 presents an incorrectly annotated increase in valence, and clip 15 correctly shows no transition. The x-axis represents seconds after the start of the clip.

matter how we divide the population the results are not significantly different, possibly because of the limited number of participants.

6.3.8 Conclusions

The main question of this study is: *can* METACOMPOSE *reliably express mood states?* In response, we described an experimental evaluation in which we created music clips from METACOMPOSE (either containing a transition in affective state or not), and asked participants to annotate the pieces, both in real-time and after a complete first listening.

Analysis of the data supports the hypothesis that transitions in affective expression intended in the compositions produced by METACOMPOSE can be recognized by the listeners, and moreover that the sampled levels of arousal are correctly detected with a strong statistical significance. Valence expression seems less well-defined: (i) from

the survey answers we see no strong difference between the annotations provided for *Neutral* and *Positive* pieces, (ii) from the analysis of transition perception we observe some incorrect annotations, and (iii) in the real-time annotation some incorrectly perceived changes can be noticed in affect static clips.

To explain point (i) we hypothesize that the fault lies in the introduction of dissonances: METACOMPOSE seems to only start to include dissonances when expressing negative valence. This means that dissonance-wise there is no difference between *Positive* and *Neutral* valence levels. Points (ii) and (iii) however seem to uncover a more systematic flaw in our expression theory: it seems that one (or more) of the features that they associate with arousal have also an effect on valence, as we can observe perceived increases/decreases in valence in response to relative changes in expressed arousal. We need to acknowledge that our sample size is not very large, yet considering the very strong statistical significance of the results we obtained on arousal, it seems likely that METACOMPOSE does indeed present some deficits in valence expression. A more systematic analysis of each music feature would be recommended to amend the mood expression theory to reliably express valence.

In summary, we show how METACOMPOSE expresses, in a reliable and perceivable way, affect arousal in the music clips it generates. However, there are emergent issues in affect valence expressions, very likely due to some interplay between the musical features associated with arousal and the ones associated with valence.

Chapter 7

Effect on player experience of mood expressive music produced by METACOMPOSE

This chapter describes the results of a study on the effect of using METACOMPOSE, under different configurations, to create the background music for the Checkers game. This represents the final contribution of this thesis, in which we bring a game together with affective music generation and we observe the effects on the player's experience. In section 9.2 we also describe a second study which is planned but not yet conducted.

7.1 Introduction

This chapter explores the use of the METACOMPOSE system in a game context. The game chosen for this experiment is American Checkers (see following section for description of the game), this game was chosen for two main reasons: it has simple rules that are easy to grasp for people unfamiliar with it, and it has a minimal amount of intrinsic narrative. While the former comes from practical considerations, we wanted a game that satisfied that latter requirement to remove as many variables as possible that could influence the perception of the game. The research questions this study addresses are:

1. Can we observe any difference in player experience (emotionally) when presented with affective-dynamic music compared to static music?

2. Can we observe any difference when the music is supporting the game's internal narrative/state?

These research questions are complemented by two hypotheses:

1. Players will prefer background music with dynamic affective expression.

2. Players will prefer background music where the affective expression is consistent with the game state.

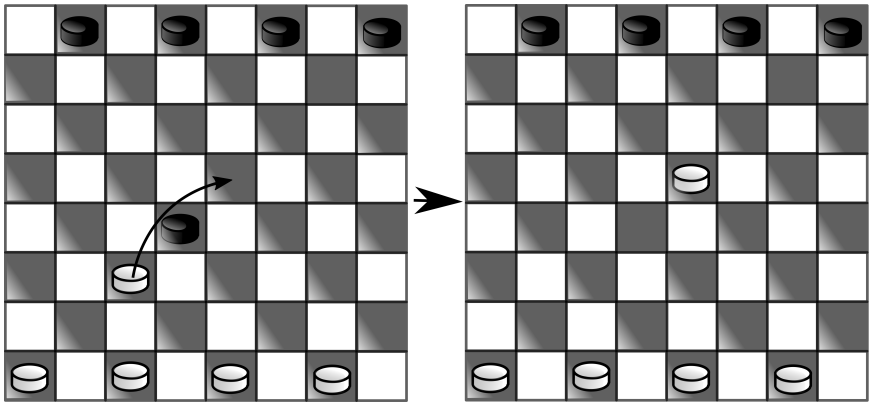

FIGURE 7.1: A capture: captures are achieved when a piece is able to "jump" over an enemy piece. This is only possible when the square beyond the enemy piece is empty.

Essentially, the first question allows us to explore the effect of *dynamic* music, while the second one the effect of *adaptive* music. To this end, we present and discuss the results of a participant-based evaluation study in which participants were tasked to play two Checkers games with different setups. We collected both self-reported – through a questionnaire – and physiological data.

7.2 Checkers

Checkers (or *Draughts*, in British English) is a family of strategy board games for two players. The invariable characteristics of a Checkers-type game are: pieces are uniform (they all display the same gameplay rules), movement is strictly diagonal, captures are achieved by jumping over opponent pieces, and captures are mandatory. There are many variants, which usually change the board size (like the Polish draughts, which plays on a 10x10 board) or the capture constraints (e.g. in the Russian Draughts all captures have to be made), but can also include variations to core rules of the game (in Italian draughts *men* pieces cannot capture *kings*).

The specific version we have used in this study is the *American checkers* (or *straight checkers*). This version is played on an 8x8 chequered board with 12 pieces per side. The objective of the game is to remove all enemy pieces from the game.

There are two types of pieces:

Men: initially all the pieces on the board are "men" (uncrowned pieces). These pieces can move one step forward diagonally or, in case there is an empty space behind an enemy piece, it can perform a capture by "jumping" over it (see Figure 7.1). It is also possible for the man to chain multiple captures if there is the opportunity.

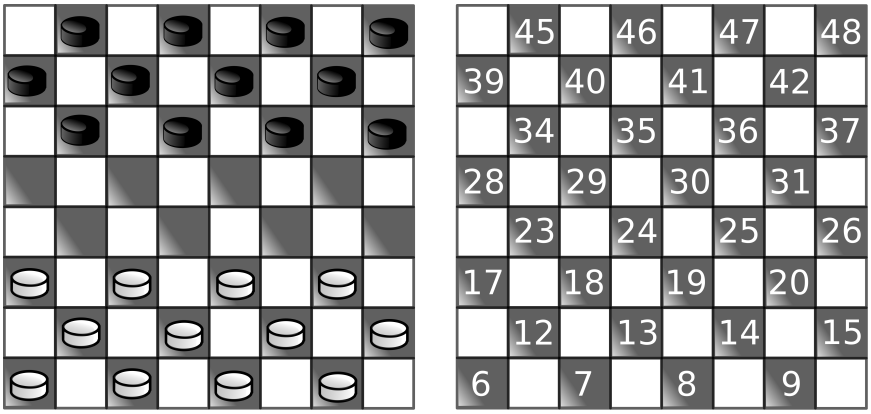

FIGURE 7.2: The initial state of a Checkers game and the numeric notation used by the raven-checkers framework.

Kings: Whenever a *man* reaches the enemy end of the board (*kings row*), it becomes a *king*. In physical games this is marked by the addition of an additional *man* token on top of the promoted one, in the software we use in this experiment a crown sprite is added to the piece. These pieces obtain the new ability of moving and capturing backwards, they also maintain the ability of chaining captures.

7.2.1 Rules

Each player starts with 12 men on the dark squares closest to the player (occupying 3 rows, see Figure 7.2). Tokens can be white or black depending on the player, and the player that is playing black starts. Players take turns making a move until the game is over.

There are two possible moves that each player can take during their turn:

- Simple move: this consists in moving a piece in an adjacent empty square. Men can only move diagonally forward, while kings can move backwards as well.

- Jump (or capture): this move consists in moving a piece from a square adjacent to an enemy piece to the square immediately beyond it on the same line ("jumping" over it). This move can only be taken if the square beyond the enemy piece is empty. Once a capture has been taken, the enemy piece that was jumped is removed from the game. In American checkers any piece can jump any other type of piece, with the constraint that men can only capture forward. Jumps can be "chained" in a single move whenever another capture is possible from the position the piece lands on after a previous capture. Moreover, while captures are mandatory, there is no requirement to maximise

the number of captures in one single turn if there are multiple choices of captures.

The game ends whenever all pieces of a player are removed from the game or whenever a player is left with no legal move. If no side can force a win (e.g. if both players only have one king, and do not fall in simple traps) the game ends in a draw.

7.2.2 AI for Checkers

Checkers was one of the first games to which AI was applied to try to solve it, the earliest program was developed in 1951 by Christopher Strachey. Since the 1990s the Schaeffer's *Chinook* program has been the strongest one, performing at the highest human-levels until 1996, when it won the U.S. National Tournament by an incredible margin [206]. Compared to other games such as Go, Checkers is relatively simple, with its $\approx 10^{20}$ possible positions and game-tree complexity of about $\approx 10^{40}$, so it is not surprising that in 2007 Schaeffer *et al.* published "Checkers is solved" [207], declaring the game to have been (weakly) solved from a computational point of view. Schaeffer [208] clarifies that by solved he means that, given that both players never make mistakes and chose the best possible moves, the game will always end up in a draw.

7.3 Experiment design

The main objective of this study is to explore if any differences in enjoyment and perception of the Checkers game can be observed through different configurations of affective music produced by METACOMPOSE.

An experiment was designed where participants would play two games of checkers, while listening to two (out of three) different setups of generated music. During these games players were asked to wear an E4 Wristband, which allowed us to record various physiological measures such as: Blood Volume Pulse (BVP), galvanic skin resistance (GSR), and peripheral skin temperature. These possible setups were:

- Static expression: METACOMPOSE rendered the piece without any change in affective expression throughout the game. This acts as a control group to find if differences arise between a static piece of music and a dynamic one.
- Consistent affective expression: at the start of the player's turn, an evaluation of the game state is conducted and some representative values of valence and arousal passed to Metacompose. In this way the music should reflect the state of the game.

- Random affective expression: at the start of the player's turn, random values for valence and arousal are passed to Metacompose. This acts as a control group (like group 1), but allows us to differentiate between differences due to random and guided dynamics.

Afterward participants were tasked with answering four comparative questions regarding the two games. As stated before, Martinez and Yannakakis [201] suggest that ranking produces more consistent and reliable data when annotating affect information, therefore the choice of asking the participants to compare two pieces of music. The questions were:

- **Which game did you find more engaging?**
  "The first one"/"The second one"/"Neither"/ "Both Equally"
- **In which game was the music best?**
  "The first one"/"The second one"/"Neither"/ "Both Equally"
- **In which game did the music better match how exciting the game was?**
  "The first one"/"The second one"/"Neither"/ "Both Equally"
- **In which game did the music better match how well you were playing?**
  "The first one"/"The second one"/"Neither"/ "Both Equally"

As in the studies detailed in the previous chapter, we also include the more neutral answers "Neither" and "Both Equally" to avoid randomness in the data from participants who cannot decide which clip satisfies the evaluation criterion better or worse.

A survey was developed with HTML and PHP, using a MySQL database to hold the data collected. The checkers framework we used is an open-source AI framework called *raven-checkers*[1] which includes a computer agent. The PHP code invokes the checkers framework through the *exec()* function, which effectively stops the execution of the PHP code until the annotation terminates.

The experiment was designed for the participants to play two games of Checkers with 2 randomly chosen setups (repetitions of the same setup were not allowed). As each game can take between 5 and 10 minutes, the experiment was designed to last between 10 and 20 minutes for each participant.

7.4 Evaluation of the game state

To pass METACOMPOSE the valence/arousal coordinated to change the affective expression of the music, we have to evaluate the state

[1] https://github.com/bcorfman/raven-checkers

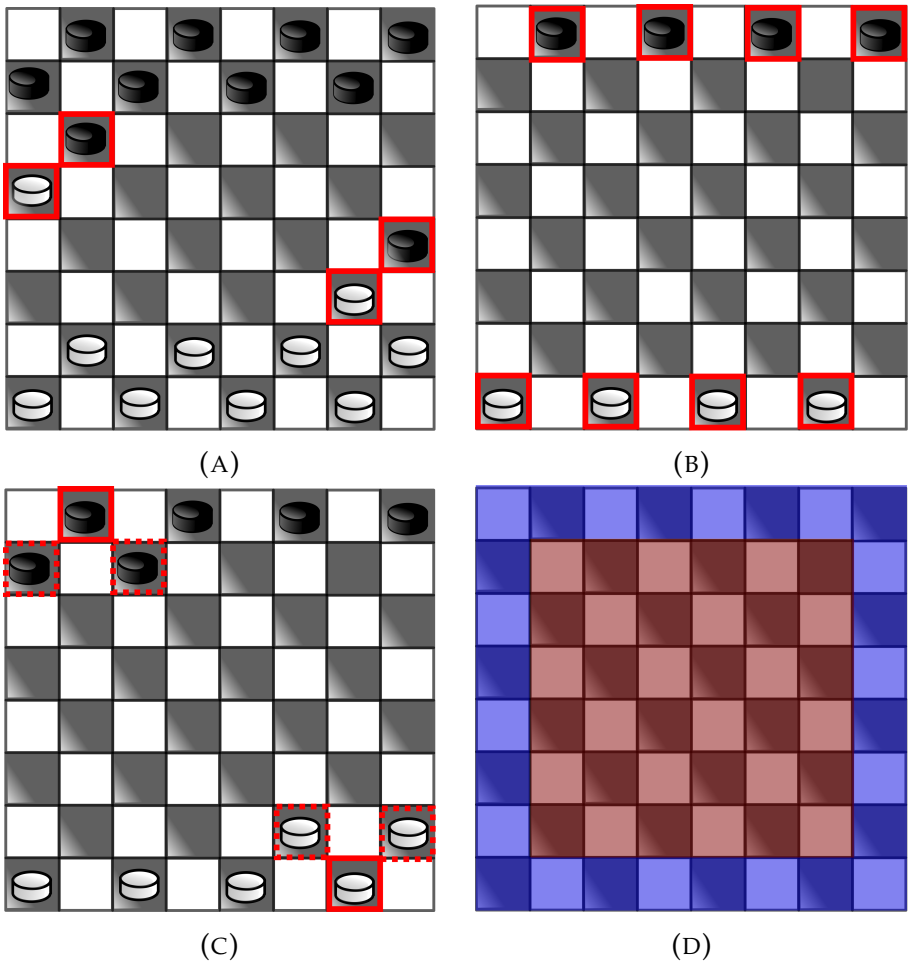

FIGURE 7.3: Visualisations of some of the metrics used to judge the state of the board: (a) a cramp, (b) the backrank guard, (c) the double corner configuration, and (d) the two positioning areas (centre and edge).

of the game. We decided to derive the valence value from a utility value meant to describe how good the current board configuration is for the player, while arousal is based on the possible moves that the player can take ("how much is at stake for the next move?"). In this section we will describe in detail how these calculations are made.

The utility value we use to describe valence is the same used by the *raven-checkers* controller to evaluate the game state. It is derived from various features of the current game state:

- Piece count: men and kings for the two players are counted and weighted (100 for each man, and 130 for each king). The piece count utility is given by the value for the player's pieces minus the value for the opponent pieces:

$$\begin{aligned}\text{PieceCount} = (&100\text{BlackMen} + 130\text{BlackKings}) - \\ &(100\text{WhiteMen} + 130\text{WhiteKings})\end{aligned}$$

- Cramp: in Checkers a "cramp" is a restriction of mobility in a region of the board. A bonus is given to the player if it has a piece in position 28 while the opponent has a piece in position 34 (see Figure 7.3a). In the opposite way a malus is applied in case the opponent has secured a mirrored position.

- Backrank Guard: a bonus is given depending on how many pieces the player still has in the row closest to her. This means that until these are moved no enemy man can become a king and, if it can be maintained, it would often require the opponent to sacrifice a piece to free a spot in the kings row.

- Double Corner: A bonus is given if the player has a man in position 9 and another man in either position 14 or 15 (see Figure 7.3a). Position 9 is relatively more exposed than the other squares in the kings row, as if an enemy piece can get in position 15 it would become virtually untouchable until it becomes a king. The presence of another piece in the two adjacent squares helps alleviate such a problem.

- Centre positions: this utility function looks at how many pieces lie in the central section of the board (not lying on the edges, see Figure 7.3a), giving a bonus based on the difference between the number of player pieces and enemy pieces. Kings are given a higher weight in this calculation. Having pieces in the centre of the board, while possibly exposing them to captures, can avoid most easy traps, especially towards the end game.

  $$\text{Center} = (\text{BlackMen} - \text{WhiteMen}) + 5(\text{BlackKings} - \text{WhiteKings})$$

- Edge positions: this utility function looks at how many pieces lie in on the edges of the board (see Figure 7.3a), giving this time a penalty based on the difference between the number of player pieces and enemy pieces. Kings are given a higher weight in this calculation. A piece lying on the edge has severely limited movement, making it easy to create situations where the opponent can force a capture without exposing her pieces.

  $$\begin{aligned}\text{Edge} = -[&(\text{BlackMen} - \text{WhiteMen}) + \\ &5(\text{BlackKings} - \text{WhiteKings})]\end{aligned}$$

- Tempo: this evaluation function looks at the amount of men left on the board (both players) to estimate if the game is in its opening, midgame, or endgame phase. It then returns a value based on the phase, the amount of men left, and how close they are to the kings row.

To calculate arousal we have decided to use, as a measure of "tension", how many moves the player can take and how much these can

change the course of the game. For example if the player can take five moves but all of these are of little consequence to the game, the arousal expressed in the music will be low, while if there are moves that can improve/worsen the situation the music will reflect this by becoming more stressful. This is implemented through:

1. Calculate all possible moves m from the current state s.
2. For each resulting state $s_i = T(s, m)$ (where T is the transition function that returns a new board state given an initial state and a move), calculate the best move a that the adversary can take, and save the utility value of state $s_i i = T(s_i, a)$.
3. Once we have calculated the worst situation achieved with every possible move, we calculate the standard deviation of such values. This gives us a measure of how much the game could change from the current game board, which we then use as input to METACOMPOSE's arousal parameter.

7.5 Music generation

As opposed to the experiments described in Chapter 6, the music in this experiment is not generated beforehand. Instead the music is generated in real-time by METACOMPOSE, as the system was designed to function. For each participant, METACOMPOSE creates one composition and uses that one as basis for the music generated in both of the player's playthrough. This way we ensure that no difference in the player's response is due to a potential quality difference between two compositions.

7.6 Results and analysis

The data collected corresponds to 26 self-reported comparisons and 34 recordings of physiological measurements, respectively from 26 and 17 participants. We have only gathered physiological measurements from 17 out of the 26 participants due to an issue in the recording software. Recall that each participant was presented with a randomized selection of 2 distinct experimental setups (from the three described in Section 7).

7.6.1 Self-report

This section will discuss the results obtained through the survey portion of the experiment. As shorthand we will refer as the criteria the participants used to evaluate the games they played using the labels: *engage*, *best*, *exciting*, *well*. Refer to Section 7 for the complete text of the questions.

TABLE 7.1: Participants' answers to our criteria when comparing the **consistent** and **random** setups. Also shown are the p-values, calculated using a two-tailed binomial test, and the Binomial Effect Size Display.

	Engage	Best	Exciting	Well
Preferred consistent	6	6	6	4
Preferred random	1	0	2	3
No preference	2	3	1	2
Total succ+fail	7	6	8	7
Binomial test	5.47E-02	**1.56E-02**	1.09E-01	2.73E-01
BESD	71.40%	100%	50.00%	14.30%

For now we only consider definitive answers (i.e. the participant chooses one of the music clips presented); we will look at the impact of the neutral answers at the end of this section. As for the study described in Section 6.1, under the definite choice constraint, the data becomes Boolean: the answers are either *"user preferred the first setup"* or *"user preferred the second setup"*. To analyse this data we use a two-tailed binomial test, with as null hypothesis that both categories are equally likely to occur and, as we have only two possible outcomes, that probability is 0.5. The Binomial Effect Size Display (BESD) [204] is another way of looking on the effects of treatments by considering the increase of success through interventions. This is an interesting measure, as it elucidates how much of an effect is created, in our case, by the introduction of altered tones.

7.6.2 Consistent vs Random expression changes

Out of our participants, nine were shown the comparison between the consistent and static setups. As can be seen in Table 7.1, a strong statistical significance can only be observed for the *best* criterion. If we had adopted less stringent criteria and used p-value cutoffs of $.05$ or $.1$, we would have seen significant effects also for the *engage* ($\approx .05$) and *exciting* ($\approx .10$) criteria. Thus the null hypothesis can be refuted (at least for the *best* criterion) and a difference in distribution can be inferred between preferring the setup with consistent affective expression compared to the random expression setup. This shows how the *consistent* setup is perceived as possessing a better overall music quality, and hints at leading to better engagement and better expression of excitement in the game. The final criterion (*well*) gives us much more inconsistent results. In the next sections we will see how that seems to be a consistent behaviour and we will discuss at the end of the section why this could be such a difficult criterion to evaluate. The BESD values reflect what can be inferred from the p-values especially highlighting how, while we do not have strongly

TABLE 7.2: Participants' answers to our criteria when comparing the **consistent** and **static** setups. Also shown are the p-values, calculated using a two-tailed binomial test, and the Binomial Effect Size Display.

	Engage	Best	Exciting	Well
Preferred consistent	6	5	5	2
Preferred static	1	1	1	1
No preference	1	2	2	5
Total succ+fail	7	6	6	3
Binomial test	5.47E-02	9.38E-02	9.38E-02	3.75E-01
BESD	71.40%	67%	66.70%	33.00%

significant p-values, we see a high increase in successes in the *engage* and *best* criteria.

7.6.3 Consistent vs Static expression

Out of our participants, eight were shown the comparison between the consistent and static setups. As can be seen in Table 7.2, no strong statistical significance can be observed for any criterion. Nonetheless we can observe some relatively low p-values for the *engage* ($\approx .05$), *best* ($\approx .09$), and *exciting*($\approx .09$) criteria. Still, it can be observed that, compared with the static setup, the *consistent* setup seems to be perceived as: providing a better engagement, having higher quality, and better expressing in-game excitement. These results seem to reflect the ones reported in the previous section, showing how the consistent experimental setup seems to be better perceived than the other two. We refer the reader to Section 7.8 for more discussion on the *well* criterion. The BESD values highlight that there is a significant increase in preference for the engage, best, and exciting criteria when using the consistent setup, although the calculated p-values are not so strongly significant.

7.6.4 Random vs Static expression

Out of our participants, nine were shown the comparison between the consistent and static setups. As in the previous section, Table 7.3 shows no p-values with strong statistical significance for any criterion. While the amount of data collected at the time of this writing is not enough to draw reliable conclusions, we can draw some hypotheses about how the games are differently perceived in these two setups. It seems that static music seems to be considered more engaging and – to a smaller degree – have overall better quality than music with random changes in affective expression. Conversely, it appears that random changes in expression are still perceived as more supportive to the excitement of the game. Intuitively we can imagine

TABLE 7.3: Participants' answers to our criteria when comparing the **random** and **static** setups. Also shown are the p-values, calculated using a two-tailed binomial test, and the Binomial Effect Size Display.

	Engage	Best	Exciting	Well
Preferred random	2	2	6	1
Preferred static	5	4	2	1
No preference	2	3	1	7
Total succ+fail	7	6	8	2
Binomial test	1.64E-01	2.34E-01	1.09E-01	5.00E-01
BESD	-42.90%	-33%	50.00%	0.00%

how participants could have found the static music to be less disruptive than the one presenting random expression changes (hence better scores for *engage* and *best*). At the same time, it is likely that at times the random changes in expression might have matched (at least in some part) the current state of the game or the excitement perceived by the player, thus leading to a better score for the random setup in the *exciting* criteria. While in a lesser amount than the results discussed in the previous sections, these hypotheses are corroborated by the BESD values. Nonetheless, with the current data at our disposal we have to consider these two setups as being relatively equivalent.

7.6.5 Physiological data

In this section we will discuss the results obtained from the collection of physiological measurements from the participants. In particular we will analyse how the recorded heart-rate and electromyographic activity measurements relate to the values of arousal/valence we calculated from game states. This means that differences observed in these measurements should be due to the different experimental setups: the music changes consistently with the game states, the music changes randomly independently of the game, and the music presents a static expression throughout the game. This section is divided in two main parts, one where we will analyse differences in the groups compared to the valence values, and a complementary one where we will discuss arousal.

The results presented here are calculated using the Pearson product-moment correlation coefficient. This is a measure of correlation between two variables (e.g. heart-rate and valence) which is obtained by dividing the covariance of the two variables by the product of their standard deviations. The Pearson correlation coefficient r has range $(-1, 1)$, where $+1$ represents a perfect direct increasing linear relationship, -1 represents a perfect decreasing inverse linear relationship, and 0 represents no correlation.

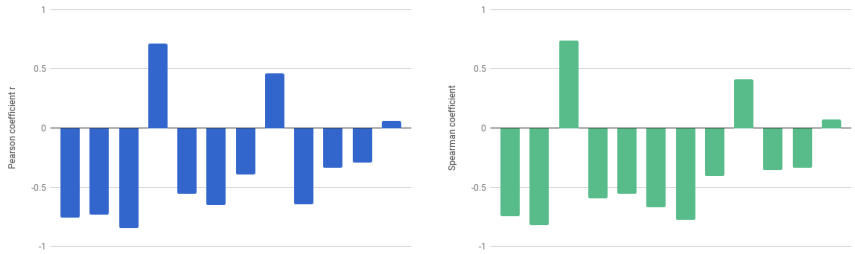

FIGURE 7.4: The Pearson and Spearman correlation coefficients between the valence calculated from the game state and the **heart-rate** measurements for the games with the **consistent** experimental setup. The coefficients are sorted from the lowest (left) to the highest (right) p-value related to them.

We also calculate the Spearman's rank correlation coefficient, which differs from the Pearson coefficient by not assessing linear correlation, but monotonic correlation. This means that it evaluates to what extent the increase of one variable relates to an increase or decrease of the second one. Analogously to the Pearson correlation coefficient, the Spearman coefficient also has range $(-1, 1)$, where $+1$ represents a perfect positive rank correlation (i.e. an increase in one value always corresponds to an increase in the second one), -1 represents a perfect negative rank correlation, and 0 represents no correlation. We decided to include this measurement as the Pearson correlation coefficient assumes a linear correlation between the two variables, which might not necessarily be the case.

It is important to remember that correlation does not imply causation, but what we want to observe here is the differences between different groups rather than proving some direct influence of a variable on another one.

7.6.6 Valence between groups

Consistent group

Figure 7.4 shows the Pearson and Spearman correlation coefficients calculated for the twelve participant data we collected in this setup. As these coefficients can have different degrees of confidence (p-values), we have decided to visualize them ordering them from the most significant to the least. In the following analysis we do not consider any coefficients with a confidence value greater than 0.05, which removes two Pearson coefficient and one Spearman coefficients. We can observe that most measures present a negative correlation between valence and HR, if we calculate the average correlation we obtain ≈ -0.37 (stdev ≈ 0.53) for the Pearson coefficients and ≈ -0.37 (stdev ≈ 0.50) for the Spearman coefficients.

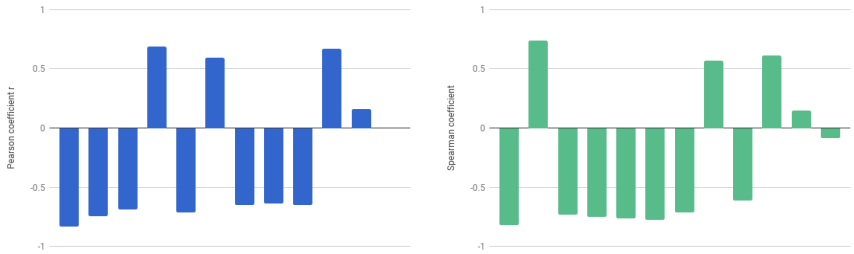

FIGURE 7.5: The Pearson and Spearman correlation coefficients between the valence calculated from the game state and the **skin conductance** measurements for the games with the **consistent** experimental setup. The coefficients are sorted from the lowest (left) to the highest (right) p-value related to them.

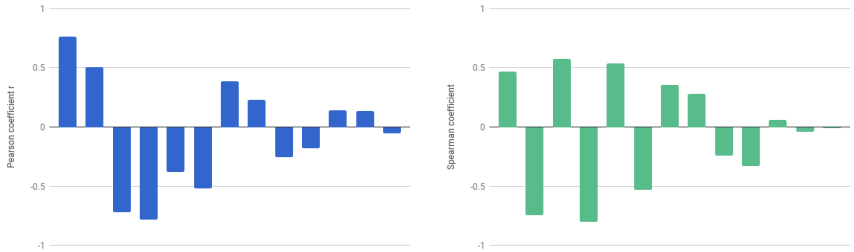

FIGURE 7.6: The Pearson and Spearman correlation coefficients between the valence calculated from the game state and the **heart-rate** measurements for the games with the **random** experimental setup. The coefficients are sorted from the lowest (left) to the highest (right) p-value related to them.

As can be seen in Figure 7.5, EDA measurements show very similar results, with a general trend of negative correlations. In this case we find two Pearson coefficients and two Spearman coefficient with p-value less than 0.05. The average coefficients we obtain are ≈ -0.30 (stdev ≈ 0.65) for the Pearson correlation and ≈ -0.32 (stdev ≈ 0.67) for the Spearman rank correlation.

Given the very high standard deviations found in the coefficients for these variables (always more than 0.5), it is hard to assume that there is a strong correlation between the variables. Nonetheless we note a negative trend in most participants.

Random group

Figure 7.6 shows the Pearson and Spearman correlation coefficients between HR and valence calculated for the thirteen participants data we collected in the **random** experimental setup ordered by their p-values. In this case we observe a higher diversity in coefficients, with

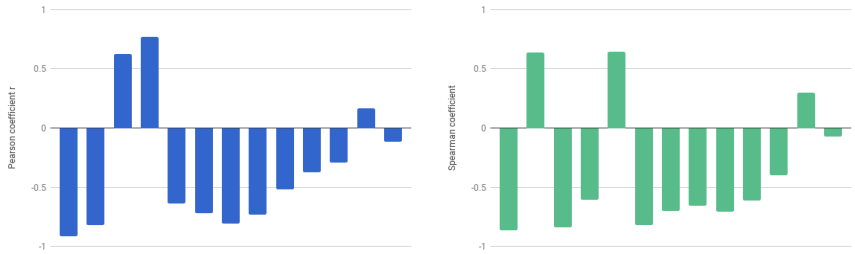

FIGURE 7.7: The Pearson and Spearman correlation coefficients between the valence calculated from the game state and the **skin conductance** measurements for the games with the **random** experimental setup. The coefficients are sorted from the lowest (left) to the highest (right) p-value related to them.

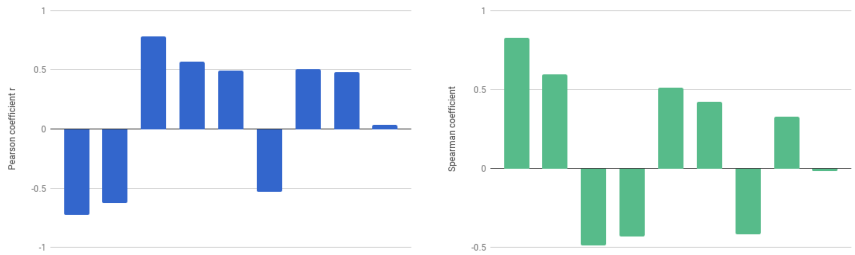

FIGURE 7.8: The Pearson and Spearman correlation coefficients between the valence calculated from the game state and the **heart-rate** measurements for the games with the **static** experimental setup. The coefficients are sorted from the lowest (left) to the highest (right) p-value related to them.

an almost equal separation between positive and negative correlation. There are also more coefficients with not significant p-values (>0.05): six Pearson coefficients and six Spearman coefficients. While we can observe a negative correlation in the average of both Pearson (≈ -0.11, stdev ≈ 0.64) and Spearman (≈ -0.02, stdev ≈ 0.64) coefficients, it is much lower compared to the **consistent** group.

We can observe very different data when looking at EDA (see Figure 7.7), here the negative correlation appears more marked. We find four non statistically significant Pearson coefficients and three Spearman ones. The averages reflect the visual observation, with an average of ≈ -0.42 (stdev ≈ 0.64) for the Pearson coefficients and of ≈ -0.45 (stdev ≈ 0.58) for the Spearman coefficients.

Static group

In the static group we have obtained nine observations, if we analyse the correlation between predicted valence and HR (see Figure 7.8) we can observe that the coefficients are fairly well distributed both

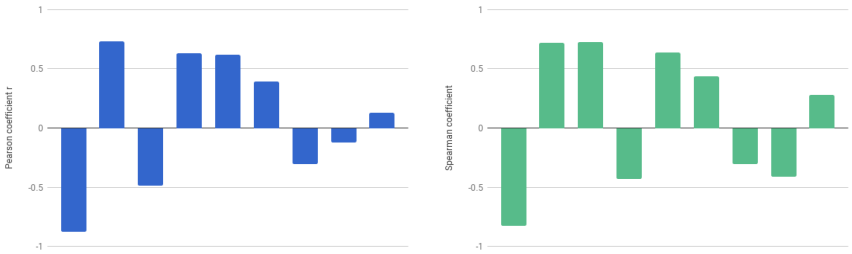

FIGURE 7.9: The Pearson and Spearman correlation coefficients between the valence calculated from the game state and the **skin conductance** measurements for the games with the **static** experimental setup. The coefficients are sorted from the lowest (left) to the highest (right) p-value related to them.

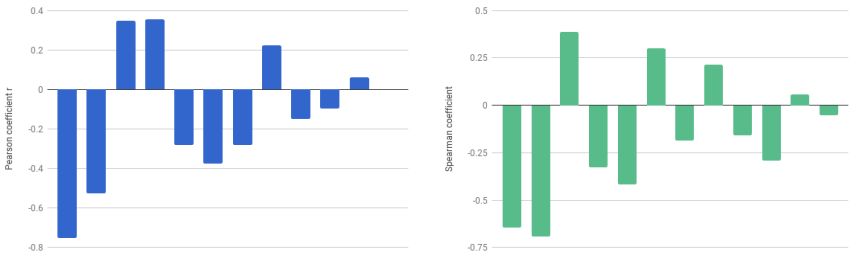

FIGURE 7.10: The Pearson and Spearman correlation coefficients between the arousal calculated from the game state and the **heart-rate** measurements for the games with the **consistent** experimental setup. The coefficients are sorted from the lowest (left) to the highest (right) p-value related to them.

in polarity and magnitude. This observation is corroborated by the averages of the Pearson (≈ 0.12, stdev ≈ 0.63) and the Spearman (≈ 0.15, stdev ≈ 0.57) coefficients. Only few of the coefficients are non-significant, respectively one and two for the Pearson and Spearman correlation tests.

A similar distribution can be observed also in the EDA data, as seen in Figure 7.9), with a slightly higher number of non-significant coefficients (three and two). As would be expected, we also obtain similar averages (Pearson: ≈ 0.17, stdev ≈ 0.68; Spearman: ≈ 0.14, stdev ≈ 0.64). We can then safely state that no apparent relationship between HR and EDA is reflected in the data in the **random** experimental setup.

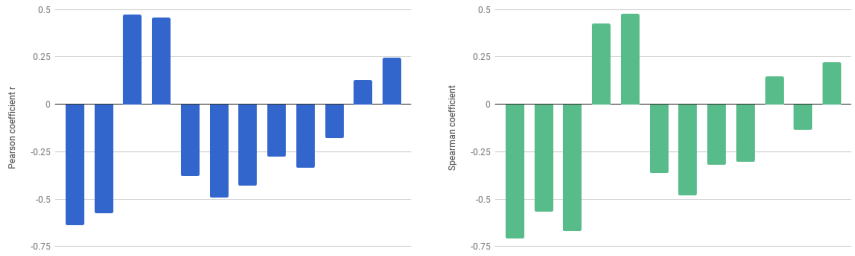

FIGURE 7.11: The Pearson and Spearman correlation coefficients between the arousal calculated from the game state and the **skin conductance** measurements for the games with the **consistent** experimental setup. The coefficients are sorted from the lowest (left) to the highest (right) p-value related to them.

7.6.7 Arousal between groups

Consistent group

As in the previous section, Figure 7.10 shows the Pearson and Spearman correlation coefficients calculated for the twelve participants data we collected in this setup. We can observe a fairly high amount of variance between the coefficients, yet this time only four are statistically significant for both correlation tests. This time the two average correlations are fairly different: ≈ -0.14 (stdev ≈ 0.58) for the Pearson coefficients and ≈ -0.32 (stdev ≈ 0.50) for the Spearman coefficients.

As can be seen in Figure 7.11, EDA measurements show similar results, if with a more marked negative correlation. In this case we find only five Pearson coefficients and five Spearman coefficient with p-value less than 0.05. The average coefficients we obtain are ≈ -0.23 (stdev ≈ 0.48) for the Pearson correlation and ≈ -0.27 (stdev ≈ 0.50) for the Spearman rank correlation.

Random group

Figure 7.12 shows the Pearson and Spearman correlation coefficients between HR and arousal calculated for the thirteen participants' data we collected in the **random** experimental setup ordered by their p-values. In this case we observe how the most significant coefficients are almost symmetrical, with a subsequent decrease in any kind of correlation. About half of the coefficients do not present significant p-values (> 0.05): seven Pearson coefficients and eight Spearman coefficients. Given how the coefficients are distributed, it is unsurprising that the averages are very close to zero for both the Pearson (≈ -0.03, stdev ≈ 0.61) and Spearman (≈ -0.09, stdev ≈ 0.56) correlation tests.

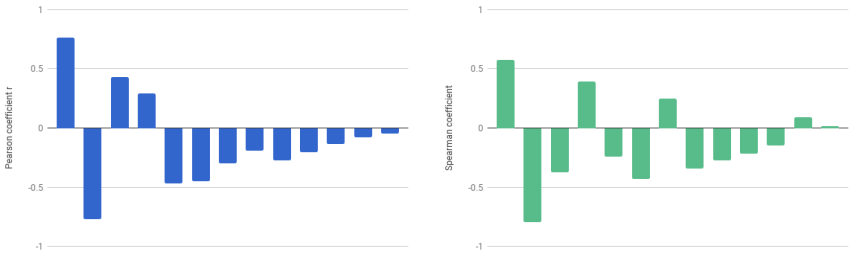

FIGURE 7.12: The Pearson and Spearman correlation coefficients between the arousal calculated from the game state and the **heart-rate** measurements for the games with the **random** experimental setup. The coefficients are sorted from the lowest (left) to the highest (right) p-value related to them.

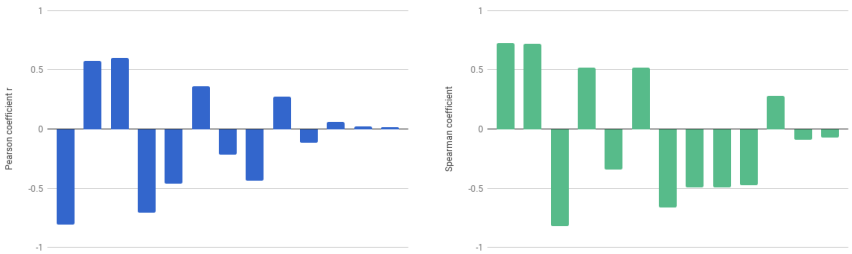

FIGURE 7.13: The Pearson and Spearman correlation coefficients between the arousal calculated from the game state and the **skin conductance** measurements for the games with the **random** experimental setup. The coefficients are sorted from the lowest (left) to the highest (right) p-value related to them.

Very similar data can be observed when looking at EDA (see Figure 7.13), especially with the Pearson correlation test. We find seven non statistically significant Pearson coefficients and three Spearman ones. The averages reflect the HR observations, with an average of ≈ -0.07 (stdev ≈ 0.66) for the Pearson coefficients and of ≈ -0.08 (stdev ≈ 0.62) for the Spearman coefficients.

Static group

Finally, in the static group we have obtained nine observations, if we analyse the correlation between predicted arousal and HR (see Figure 7.14) we can observe that the coefficients appear fairly well distributed, with a small positive tendency. The averages of the Pearson (≈ 0.33, stdev ≈ 0.60) and the Spearman (≈ 0.06, stdev ≈ 0.55) coefficients seem to confirm the observation. Many of the coefficients are non-significant for the Pearson test (five) compared with the Spearman correlation test; this might mean that the positive average found between the Pearson coefficient could be overly optimistic.

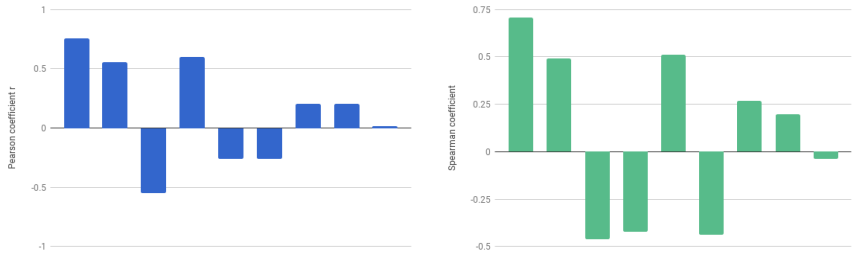

FIGURE 7.14: The Pearson and Spearman correlation coefficients between the arousal calculated from the game state and the **heart-rate** measurements for the games with the **static** experimental setup. The coefficients are sorted from the lowest (left) to the highest (right) p-value related to them.

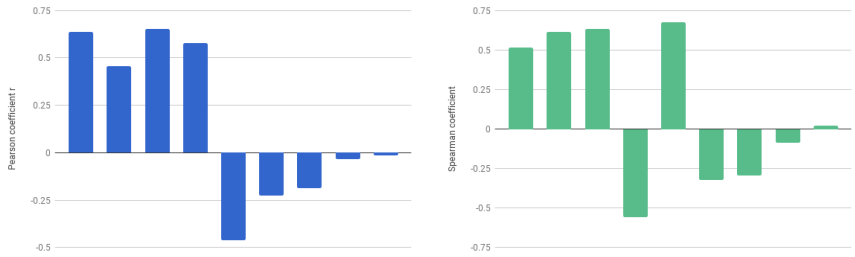

FIGURE 7.15: The Pearson and Spearman correlation coefficients between the arousal calculated from the game state and the **skin conductance** measurements for the games with the **static** experimental setup. The coefficients are sorted from the lowest (left) to the highest (right) p-value related to them.

A much more positive distribution can be observed in the EDA data (as seen in Figure 7.15), we also observe a more balanced amount of non-significant coefficients (four and two). As would be expected we also obtain slightly higher averages (Pearson: ≈ 0.37, stdev ≈ 0.47; Spearman: ≈ 0.18, stdev ≈ 0.54).

7.6.8 Summary

In this section we have observed the apparent relationships between our calculated valence/arousal values and two physiological measures (HR and EDA). Here we provide a short summary of the observed correlations:

Valence

Consistent group: small negative correlations appear between both physiological measures and valence.

Random group: no correlation can be observed with HR, but EDA shows a small negative correlation.

Static group: no correlations can be observed.

Arousal

Consistent group: as with valence, we can observe a slightly negative correlation between arousal and the two physiological measures.

Random group: no correlations can be observed.

Static group: a slightly positive correlation to EDA can be observed.

7.7 Demographics

From the 26 people that participated in the experiment, 16 are males, 9 females, and 1 participant did not express gender. The participants' age has an average of 29.4 years ($stdev \approx 5.9$). In regards to the other demographic answers, expressed in 5-point Likert scale (0–4), most people self-reported little experience with the game of Checker ($avg = 0.84$, $stdev 0.8$, $mode = 1$), and a considerable experience with video-games ($avg = 2.56$, $stdev \approx 1.04$, $mode = 3$). No matter how we divide the population the results are not significantly different, possibly because of the limited number of participants.

7.8 Discussion

The main questions we wanted to explore through this study are: can any difference in player experience be observed when presented with affective-dynamic music compared to static music, and can any difference be observed when the music is supporting the game's internal narrative/state. To this end we designed an experiment where participants were tasked with playing two games of Checkers in three different experimental setups.

From the self-reporting task of the experiment we can show how music with affect expression that is consistent with the game state appears to be better perceived than the other two setups. In particular we observe that it is generally perceived as having better overall quality and –to a less significant degree– lead to a more engaging experience, and to a better match for the perceived excitement in the game. The static setup and the random expression one appear to be much more equivalent, although we can observe some non-significant differences between the two: the static setup seems to be generally better perceived (more engaging and overall better quality), while the random one seems to better match the perceived excitement of the game. We hypothesize that the random group has too many disruptive changes in expression to be particularly liked by the listener, while the static group by definition does not present

any changes that might match the game leading to the excitement results. When looking at the answers for the last criterion ("in which game did the music better match how well you were playing?") we find very inconsistent results between each of the groups. This might be caused by the complexity of the question, which requires the participant to evaluate her own play-through of the game and compare it with a second one.

The physiological data, analysed in Section 7.6.5, provided mostly inconclusive results, this is mainly due to the very high variance we can observe between the observations. To summarize:

Heart-rate:

- For the **static** group we can observe no correlation between HR and valence/arousal.
- The **consistent** group shows a slightly negative correlation between HR and both valence and arousal. While the negative correlation between valence and HR seems intuitively reasonable (the worse the player's situation, the more she gets stressed), the negative correlation between arousal and HR is surprising, as a positive correlation would be more expected.
- No correlation can be found between the **random** group and valence/arousal.

Electromyographic activity:

- For the **static** group we can observe no correlation between EDA and valence. A slight positive correlation can be seen in regards to arousal but, considering the very high valence, this is probably unreliable.
- The **consistent** group, as with the HR results, shows a slightly negative correlation with both valence and arousal.
- The **random** group shows a small negative correlation with valence, more or less with the same magnitude of the consistent group. No correlation with arousal can be observed.

Interestingly there seems to be some differences between the consistent group and the other ones, which would reinforce the results from the self-report data: both HR and EDA show a negative correlation between valence. This result seems intuitive in showing that as the situation of the player gets worse, her stress in increased. We also observe a counter-intuitive relationship between arousal and the two physiological measures, we remind the reader that both HR and EDA are measures of stress. Still, given the high variance observed throughout these measurements, it is unlikely these are reliable averages.

We have to recognise some limitations of this experiment: first and foremost the amount of participants at the time of this writing is

not high enough to obtain strongly significant results, although some trends can be observed in the self-report answers. Moreover, while it seems reasonable, there is no insurance that the estimation of valence/arousal (see Section 7.4) is necessarily correct in expressing the state of the game. A preliminary study could have been designed to validate this state-evaluation, for example by asking expert checkers players to give us an evaluation of specific states and comparing it to our measurements. Finally, when looking at the physiological measurements, there is the problem that these are often slow to react and can have delays ranging between one and three seconds depending on the individual. The analysis presented here does not take this into account, instead comparing the sensor data with the state evaluations at the exact time the state was reached. We have also conducted analysis of the data delaying it by one, two, and three seconds, but found no significantly different results. As we have no way of knowing how much the delay in responses is for each participant (note that response delay is also variable within the same participant) we decided it was better not to add any assumptions of such delay and instead present the results as they are.

To conclude, through this experiment we found that participants self-reportedly preferred the dynamic affective music provided by METACOMPOSE when trying to reflect the current game-state in three out of four criteria. This result is especially significant for the perceived quality of the music. The physiological data shows some difference between the consistent group and the others, but the unreliability of these results makes it hard to reach any kind of conclusions.

Chapter 8

PRIMAL-IMPROV, a co-evolutionary modular music improviser

This chapter describes the Primal-Improv system, a system for co-evolution of improvisational modules. The modules learn how to play with a human, but in a very different way compared to usual machine-learning techniques: each module is comprised of two Artificial Neural Networks, and the evolutionary process does not just change the weights of the connection but also the topology of the networks.

Primal-Improv has also been designed to be a partial replacement to the *real-time affective music composer* part of METACOMPOSE. As described in section 5.2, this part of METACOMPOSE is composed of a number of very simple human-written improvisational modules, which could be replaced by modules evolved with Primal-Improv. This system is still a work in progress, and as such it hasn't been integrated with METACOMPOSE yet. We better describe how we think it will fit in in section 9.2.2.

8.1 PRIMAL-IMPROV

Improvements in computer science, evolutionary computation, and music informatics have enabled many creative performance systems producing music. The goal of most such systems is to exhibit musicality, which is determined by a listening audience or a performing musician. Many diverse approaches have been applied to this domain, due to the variety of methods in artificial intelligence (reinforcement learning, evolutionary algorithms, statistical modelling, etc.) and to different interpretations and genres of music applied to a specific method [107], [209].

Another goal of artistic expression is to create interesting structures that one cannot immediately imagine. In musical history we find plenty of rules and limitations (for example the 12-tone series), which musicians use to create interesting content. In a way, simple

limitations help to support the creative process. Interactive evolution, where human aesthetic judgment is included in the fitness function, has often been used to guide evolutionary creative systems. Another approach is to base the fitness function on music theory to emulate the way a human would play/compose. The PRIMAL-IMPROV, while still in its infancy, presents a different approach by introducing very simple rules in the fitness function, which do not necessarily need to rely on musical theory. The most interesting aspect is how, even though the system generates music according to rules of our own construction, musical structures emerge that are not obvious to human creators. These structures arise from the freedom we give to the Artificial Neural Networks (ANNs) to musically express characteristics of their morphology, thanks to the multitude of ways that the fitness functions can be tackled. Moreover, by avoiding domain knowledge our system can create music that transcends usual harmony, which can lead to some unexpected improvisations.

Part of the architecture described in this paper is inspired by *MaestroGenesis* [73], a tool for computer-assisted composition that is based on interactive evolution. The main difference between PRIMAL-IMPROV and *MaestroGenesis* is that our system does not require human input, but instead co-evolves its multiple voices according to "primitive" fitness functions.

The system presented in this paper is based on the idea of evolving improvisational modules, each capable of creating monophonic melodies. The system can be used to create accompaniments to predefined melodic phrases or, more interestingly, provide an adaptive improvisational companion to a human player. While there are other evolutionary real-time improvisers (GenJam [54], Bown [103]), the proposed system presents a novel modular architecture that allows for the creation of an arbitrary music "instrument".

Finally, when considering the real-time application of PRIMAL-IMPROV, we can note how the feedback loop between the player and the system closes: as the musician plays, he/she finds out how the system reacts to the music played and starts to use them; at the same time the system adapts to the music the human is playing, this change will likewise influence the musician, and so on.

8.1.1 System description

PRIMAL-IMPROV is an evolutionary system which uses NEAT in combination with co-evolution to create a real-time (and offline) improvisational system. In the current implementation of Primal-Improv two modules are evolved, but the structure of the system will allow for an arbitrary amount of these to be added, possibly creating a very large and diverse instrumentation. The system is developed in C#,

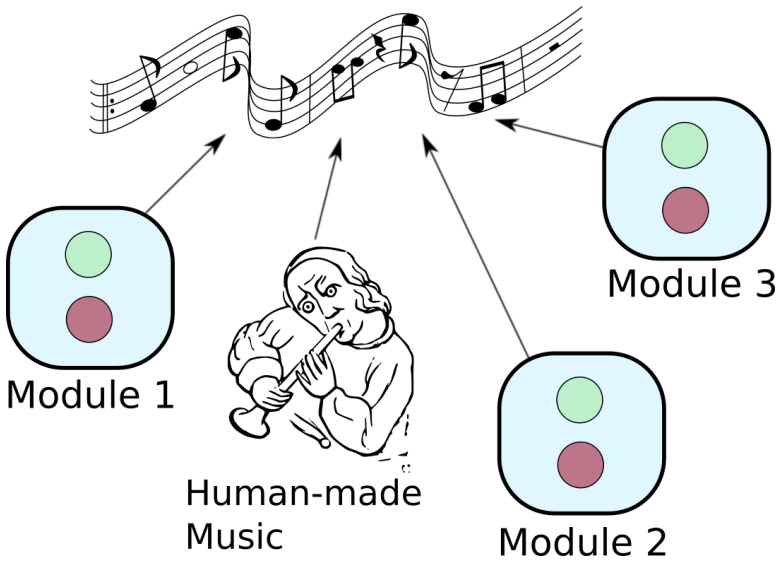

FIGURE 8.1: A high level representation of how the human and the modules by playing together create the final musical output. Note that the *human-made music* can either be a pre-written melody or a musician playing in real-time.

using the *SharpNeat*[1] library and the *C# MIDI toolkit*[2].

Module architecture

A module is defined by a collection of components needed to produce a monophonic melody in response to the environment's state (see Figure 8.2). Each module "listens" to both the human and all the other modules currently active, and according to such inputs, decides what to play next. The components of each module are:

- Rhythm neural network: this ANN controls the duration of the notes played by the module. The note duration is represented as a decaying function, where the decay depends on the length of the note (see the *Representation* subsection for a detailed explanation). The ANN has as many inputs as the number of currently active modules, plus an additional one for the human player. It presents only one output, which can be represented as the same type of function as those in input. A real-time peak recognition algorithm (described below) is applied to this function to identify when a note should end, in order to allow the next note to begin.

- Pitch neural network: this ANN determines the pitch that should be played when the *rhythm ANN* determines that a new note

[1] http://sharpneat.sourceforge.net/
[2] https://www.codeproject.com/articles/6228/c-midi-toolkit

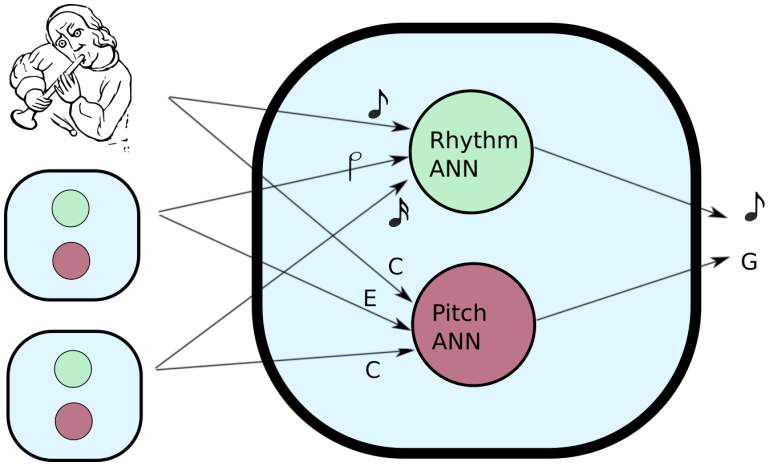

FIGURE 8.2: Module components: the rhythm ANN that controls the durations of the notes, and the pitch ANN that controls what pitch should the new notes have. Each module *listens* (has as inputs) to the human-made melody and to all the other modules to decide what to play (output).

should be played. The ANN has as many inputs as the number of currently active modules + one additional input for the human player. These inputs are a representation of the pitches being currently played. Note that the pitches are represented as absolute pitches (without octave information), meaning that a C4 and a C6 would have the same representation.

Evolutionary Algorithm

In this section we describe the function of the GA, with the assumption that only two modules are being evolved, yet the structure should be expandable to an arbitrary amount of these. In the *Conclusion* section we discuss what kind of challenges will need to be addressed to achieve this.

For each module the system has to run two instances of a NEAT GA: one for the *Rhythm ANN* and one for the *Pitch ANN*. In the case described here, this means the system is running four separate evolutionary algorithms. The four are almost identical, apart from the fitness functions, which differ between *Rhythm* and *Pitch* ANNs. As stated before, this is a cooperative co-evolutionary algorithm, which means that the individuals of the four populations are not evaluated by themselves, but instead:

1. Individuals are chosen from each of the populations.
2. The sampled ANNs are used together to produce results.
3. These results are evaluated and the computed fitness value is propagated to the ANNs.

To illustrate via a practical example: each generation the population of *Pitch ANNs for module 1* will be used in conjunction with the highest scoring *Pitch ANNs for module 2* to create some outputs using the following formula:

$$f(P_i^{'}) = \frac{\sum_{j=1}^{n} f_{Pitch}(P_i^{'}, PChamp_j^{''})}{n},$$

where f is the fitness function of a specific individual, $P_i^{'}$ is an individual i of the *module 1* Pitch GA, f_{Pitch} is the fitness evaluation of the pitches produced by the input ANNs, and $PChamp_j^{''}$ is one of the n champions of the *module 2* Pitch GA.

Likewise, the fitness for a specific individual of the Rhythm GAs is calculated as:

$$f(R_i^{'}) = \frac{\sum_{j=1}^{n} f_{Rhythm}(R_i^{'}, RChamp_j^{''})}{n},$$

where $R_i^{'}$ is an individual i of the *module 1* Rhythm GA, f_{Rhythm} is the fitness evaluation of the durations produced by the input ANNs, and $RChamp_j^{''}$ is one of the n champions of the *module 2* Rhythm GA.

The fitnesses for the *module 2* GAs are constructed analogously using the champions of the *module 1* GAs.

Fitness Function

As stated earlier, the fitness functions are calculated on the outputs of the Rhythm ANNs (and Pitch ANNs) of all the modules. These fitnesses are calculated using a human-made reference phrase, in the real-time application: this is a recording of the last notes played by the user, while in the static case it uses the first five notes of a provided piece. We decided to keep these functions simple to observe emergent behaviours and give more freedom to the ANNs.

The fitness for the Rhythm ANNs is calculated as:

$$f_{Rhythm}(R_i^{'}, RChamp_j^{''}) = -(Few + SameDuration),$$

where *Few* is a function that measures if the notes produced are less than a threshold (in the current implementation three), and *SameDuration* is a function that discourages the modules from producing the same rhythmic pattern.

The fitness for the Pitch ANNs is calculated as:

$$f_{Pitch}(P_i^{'}, PChamp_j^{''}) = -(SamePitch + Identity + OutOfKey),$$

where *SamePitch* is a function that discourages the modules from producing the same pitches, *Identity* counts the number of times the

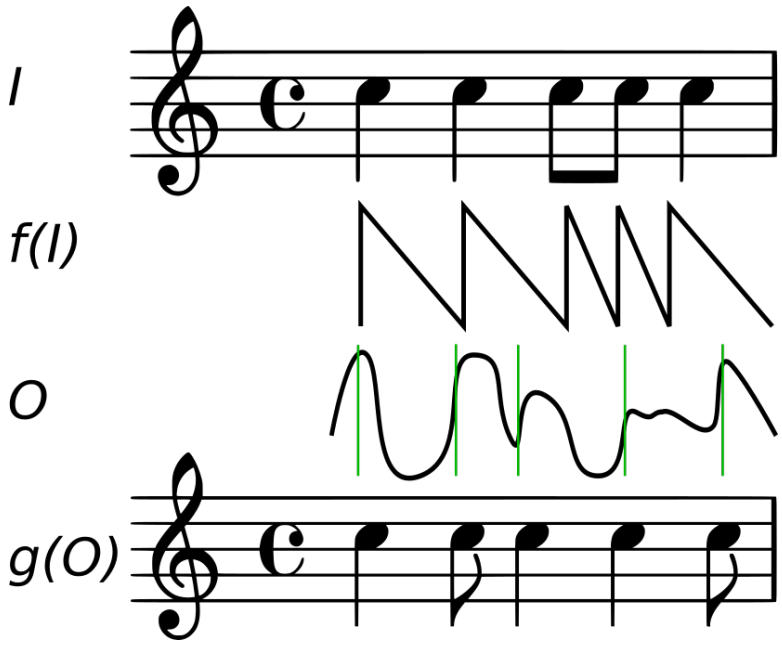

FIGURE 8.3: Representation of rhythm, from the top: input rhythm, decaying function $f(I)$ representing the input, output of the network (in green the recognized peaks), output rhythm $g(O)$.

pitches produced by each module are the same, throughout the sequence, and *OutOfKey* counts the times the produced pitches are out of key, with respect to a predefined key.

Representation

SharpNeat represents neural networks as a collection of nodes and weights connecting the nodes. As SharpNeat has not been changed in this project, we refer the reader to the SharpNeat page for more details.

What is more interesting is how the system interprets inputs and outputs: the Pitch ANNs translate the absolute pitches in values in the range $[0, 1]$:

$$\mathit{PitchInput}(P) = P/12,$$

where P is an absolute pitch, which can range from $[0, 11]$ and $\in N$, this procedure divides the interval in twelve parts, each of them representing one of the twelve notes commonly used in Western music. In this way a function to interpret the outputs can easily be determined:

$$\mathit{PitchOutput}(o) = 12o,$$

where $\mathit{PitchOutput}(o) \in N$, o is the output value of the ANN which has range $[0, 1]$.

Durations are instead represented as a sawtooth function, which generates a spike that decays according to the duration of the note

(Figure 8.3). This is a function of time, which is represented as MIDI ticks (24 per quarter note). This representation is inspired by Hoover's *functional scaffolding for composing additional music voices* [73], which allows the networks to learn when new notes are played while allowing for more freedom in creating non-standard durations. The decay corresponding to the input duration can be calculated as:

$$Decay = \frac{1}{duration \cdot PPQN \cdot 4},$$

where PPQN is the ticks (or pulses) per quarter note.

Once the decay is calculated, the values for each tick can be found by subtracting the decay from the previous value: $x_n = x_{n-1} - Decay$.

The Rhythm ANN outputs a value for each tick, and peaks in this function are interpreted as the start of a new note. The peak detection algorithm is called *Smoothed z-score*[3]. It is based on the principle of dispersion: when a data-point is x standard deviations away from the moving mean a new peak is signalled. The algorithm creates a separate moving mean and standard deviation, meaning that the signals do not risk corrupting the threshold. The main parameters of the algorithm are:

- *Lag*: the lag of the moving window, as in how many observations in the past should be used to smooth the data.
- *Threshold*: how many standard deviations does the new datapoint have to be distant from the moving mean to be recognized as a peak.
- *Influence* is a value between [0,1] that determines how much the new signals should be counted to calculate the new threshold. If the *influence* is 0 the algorithm ignores the signals while recalculating the threshold, which is the most robust option.

8.1.2 Real-Time

In the real-time application of the system the user is able to use a computer keyboard to play notes; as notes are played the modules receive them as inputs for the Rhythm and Pitch ANNs and calculate if and what notes to play. As the user is playing, a *recording* module keeps a queue of the last notes played and, after a predefined passage of time, a message is sent to update the reference phrase in the evolutionary algorithm with said recording. At the same time, as the reference phrase is updated, the current best individuals are pushed to the synthesizer where they will replace the previous best individuals. The program allows for minimal user control over the system:

[3]https://stackoverflow.com/questions/22583391/peak-signal-detection-in-realtime-timeseries-data/22640362

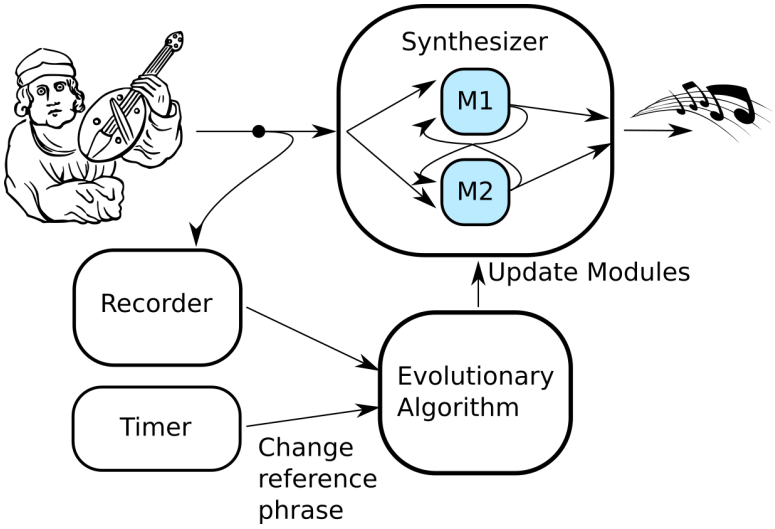

FIGURE 8.4: Representation of the Real-time architecture of Primal-Improv. The human player is recorded by a component which periodically updates the reference phrase driving the evolutionary algorithm. At the same time the modules in the GA are passed to the synthesizer to use until the next time interval has passed.

mainly switching on/off the modules. The current implementation keeps the last ten notes played by the user and switches the reference phrase every ten seconds.

8.2 Evaluation of PRIMAL-IMPROV

This section describes a quantitative user study designed as a preliminary evaluation of the musical product of the system. As music is (at least partly) a subjective matter, we evaluate it through a preference questionnaire to listeners distributed over the Internet. Participants are asked to indicate which of two pieces of music presented to them they prefer; one of these is generated with the complete system (as described in the previous section) and the other using a random fitness function. We realize that comparing against a random fitness function doesn't prove music quality, but allows us to identify (i) if the emergent music patterns are caused by the architecture of the system and (ii) if the introduction of small evolutionary pressure is enough to perceptively improve the result. Participants do not know which is which, and order is randomized. Quality is evaluated according to four criteria: *pleasantness*, *randomness*, *harmoniousness* and *interestingness* (as described in section 6.1). We chose these criteria to cover different aspects or dimensions of perceived quality of music. Our intended interpretation of the criteria is presented below. Note that no definition of these terms is offered in the survey, and there

TABLE 8.1: Number of correct, incorrect and neutral answers to our criteria for the complete system against the version with random fitness function. Also included the p-values calculated using a two-tailed binomial test and the Binomial Effect Size Display (BESD). Note that in the case of the random criteria the listener is asked to select the clip that he/she feels the most random, so it is expected that less people preferred the complete system.

Choice	Pleasing	Random	Interesting	Harmonious
Preferred the complete system	307	185	261	294
Preferred the random fitness	151	226	151	164
Neutral answer	77	124	123	77
Total non-neutral answers	458	411	412	458
Binomial test p-value	2.60E-13	0.04836	6.62E-08	1.30E-09
BESD	34.10%	-10%	26.70%	28.40%

is therefore no guarantee that participants interpret these criteria the same way we do.

An online survey was developed with HTML and PHP, using a MySQL database to hold the data collected. Participants were presented with paired music clips and asked to evaluate them using the four criteria described. Each of the four criteria has a multiple choices question structured as:

> **Which piece do you find more pleasing?** *"Clip A"/"Clip B"/"Neither"/ "Both Equally"*

Where the last word (e.g. "pleasing") is dependent on the criteria. We also include the more neutral answers "Neither" and "Both Equally" to avoid randomness in the data from participants who cannot decide which clip satisfies their evaluation, according to the criteria better or worse. Other benefits of doing this are: avoiding participant frustration, and giving us potential information on interesting individual pairs, where the pieces are considered equally good/bad.

8.2.1 Music Piece Generation

Eight clips were created for each group (*normal* and *random fitness*), for a total of 18 pieces[4]. The neural networks were allowed to evolve for 100 generations, using the first ten notes of the melody as a reference point for the evolution. The clips used in the experiment were

[4]The list and the generated music for the various groups can be accessed at *http://msci.itu.dk/Primal-ImprovFiles/*.

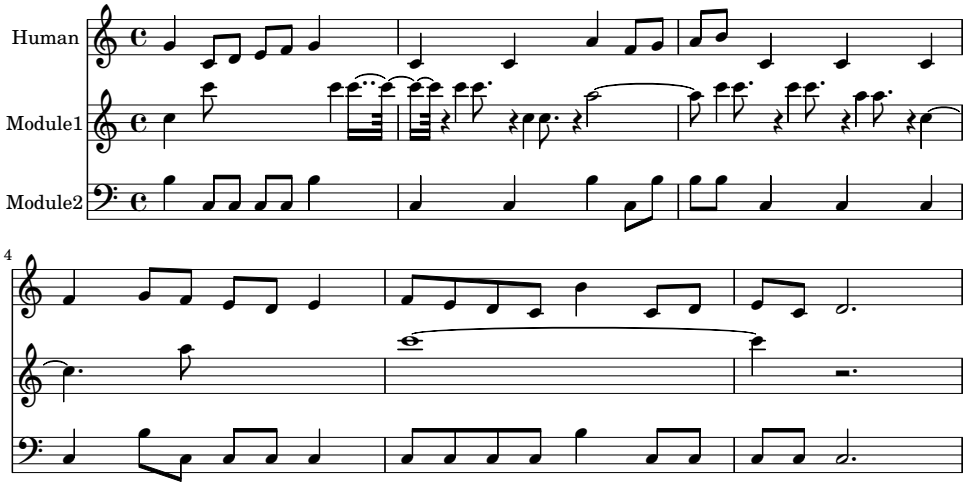

FIGURE 8.5: Example of the outputs of two modules based on Bach's *Minuet in G*.

not selected from a pool, but were the result of a singular execution of the program. The melodies used are a mix of classical melodies, traditional songs and one melody composed by the author.

8.2.2 Results and Analysis

The data collected amounts to a total of 538 answers for each of the four evaluation criteria from 87 participants. Table 8.1 shows how many responses were obtained for each criteria and how many neutral answers were collected.

For now we only consider definitive answers (i.e. the participant chooses one of the music clips presented); we will look at the impact of the neutral answers at the end of this section. Under the definite choice constraint, the data becomes Boolean: the answers are either *"user chose the clip generated with the complete system"* or *"user chose the generated with the random fitness function"*. To analyse this data we use a two-tailed binomial test, which is an exact test of the statistical significance of deviations from a theoretically expected random distribution of observations in the two categories. The null hypothesis is that both categories are equally likely to occur and, as we have only two possible outcomes, that probability is 0.5. The Binomial Effect Size Display (BESD) [204] is another way of looking at the effects of treatments by considering the increase of success through interventions. This is an interesting measure, as it elucidates how much of an effect is created, in our case, by the introduction of even very simple fitness functions.

As it can be seen in Table 8.1, there is a strong statistical significance ($p < 0.05$) for all the criteria, although there is a clearly smaller effect on the *randomness* criteria ($p \approx 0.048$) This means that the null hypothesis can be refuted and a difference in distribution can be inferred between choosing the music generated with (and without) the

described fitness functions. This indicates that music generated by the complete system is considered more pleasing, less random, more interesting and more harmonious than music generated by the degenerate system with random fitness. The BESD values reflect what can be inferred from the p-values; it can be observed that for the *randomness* criterion the effect is small (-10%) confirming that, while the p-value is significant, people cannot easily distinguish emergent musical structures. This result could be explained by the high freedom left to the ANNs and by the short length of the generated pieces, which might not have been long enough for participants to discern emergent patterns and make a judgement.

8.2.3 Demographics

Our participant's population is composed by 62 males, 14 females, and 7 people that did not specify their gender. The average age is 37.46 (stdev 17.72). Participants were asked to rate their skill with a music instrument and their knowledge of music theory according to a five point Likert scale (0 to 4). The participants reported a similar level of musical training (avg: 1.11, stdev: 1.01, mode: 0) and instrument skill (avg: 1.36, stdev: 1.19, mode: 0). The homogeneity of the population may explain how, however we partition the population, we find no significant difference in the results.

8.2.4 Conclusions

This paper describes a system for co-evolution of melody-producing modules, which can work both as an arranger for a predefined melody, or a real-time improvisational system. The main features of the system are: emergence of musical structures out of very simple rules, not requiring direct human input, and adaptiveness to a human musician.

A quantitative user study has been described, comparing the complete system with one using a random fitness function. As described in the *Results and Analysis* session we have shown a statistical significant preference in all criteria (*interestingness*, *less randomness*, *pleasantness* and *harmoniousness*) for the complete system. This shows that the fitness function evaluated, while extremely simple, does guide the evolution of the modules to create something more interesting than the networks themselves could otherwise produce. Clearly this does not prove that the system actually produces interesting music, but we believe it is a first step in showing the potential of PRIMAL-IMPROV.

This system is still in its infancy and there are many improvements that can be implemented, yet we believe that there are a couple of crucial problems that we will address immediately. The first is that the current implementation of the system is limited to two

modules; it would be interesting to find out how the system reacts with a larger number. The fitness functions will likely require adjustment to avoid a cacophony effect, due to too many modules playing simultaneously. An interesting phenomenon that appears in some of the music pieces created for the user study is a "ringing" effect (like an old time phone): this is due to the neural networks producing very short notes (1/96th of a measure) repeatedly. This is a very "non-human" way of playing and, while emulating a human player is not an objective of the system, this effect seems to be negatively perceived by listeners, so it might need to be addressed and eliminated. Finally, maybe the greatest challenge of the current system is that the evolutionary process is too slow to effectively work in real-time. This is due to the rhythm representation (see Figure 8.3), which for each evaluation forces us to construct very large arrays (in the default MIDI standard there are 24 pulses per quarter note). Once these issues are resolved, we plan to conduct qualitative experiments with musicians playing with the system in real-time, in order to find out how the interaction between the players unfolds, and what kind of creative improvisations can result.

In summary, we present (i) a novel co-evolutionary system for improvisation and melody arrangement and (ii) a quantitative study showing how even from very simple rules some interesting content seems to emerge.

Chapter 9

Discussion

This thesis set out to explore how music can be automatically generated to express affect in order to influence player experience in games. Towards that end, we have described our initial studies in expressing moods through the generation of ambient music in (Chapter 4). While these initial tests showed some promise, we quickly realised the quality of the produced music was not enough to maintain the listener's interest for the longer periods of time that the game domain would require. The METACOMPOSE system was designed to create more structured and varied music in real-time, through a combination of a graph traversal-based chord sequence generator, an evolutionary search-based melody generator, a pattern-based accompaniment generator, and affective improvisation. MetaCompose achieves this mood expressive music by creating a *composition* (a chord sequence, a melody, a basic rhythm, and basic arpeggio), which is then rendered in real-time by a number of improvisational modules. These modules use the mood expression theory described in Section 5.2.2 to change the way they express the information contained in the *composition*, achieving different affective expression. Several studies have been conducted to validate the system's structure (Section 6.1) and the expressive capabilities of the system (Sections 6.2 and 6.3). The results of this last study highlight how, while our mood expression theory is far from perfect, it allows for reliable expression of arousal and of strong differences in valence. Moreover transitions between affective states are generally correctly perceived which means that, while there might be disagreement between how to annotate the expression of a particular piece, participants distinguish changes in expression (e.g. more negative).

To explore the final research question of this thesis – How do players react to exposition to such music while playing a game? – we performed an experiment where METACOMPOSE has been applied to create background music for the game of Checkers. Results show how the usage of the METACOMPOSE system while trying to express affect consistently with the current game state is better perceived by the players compared with random affective states and static expression. Data from physiological sensors did not yield significant results, yet they suggests that the music with consistent expression could have a different effect on the players compared with

the other groups. It is important to note that a low number of participants and high variance in the analysis does not allow us to strongly support this hypothesis.

This thesis has thus explored how to create a dynamic, adaptive, and affective music system and provides early results on how can this kind of music can influence players. During this exploration we have, however, found a number of limitations to the system, expression theory, and to the generality of the conclusions. In this chapter we discuss the limitations, insights, and possible improvements, which we have observed based on these studies. In Section 9.2 we will delineate some future avenues of research.

9.1 Limitations

In this section we discuss how the design choices and the domain in which this thesis was developed create some limitations on the generality of the conclusions reached. The limitations discussed in this section include: design decisions, implementation of specific algorithms, and domain specific problems.

9.1.1 Limitations of METACOMPOSE

In order to develop METACOMPOSE we had to make a number of design decisions, among these the choice of the various techniques applied to the *composition* generation. While the study described in Section 6.1 shows how each part of the system adds to the quality of the produced music, it is noteworthy that this gives us no comparison with music composed by humans.

While creating music that would be perceived to have the same overall quality as music written by a human was not one of the objectives of this thesis, it is important to note that listeners might dislike music that can be easily identified as not made by a human. A way to understand how much of an impact this has on the perception of METACOMPOSE's music, would be to conduct a study to compare the music generated by METACOMPOSE to music created by a human. The intrinsic problem of this type of experiment is that METACOMPOSE presents a "compositional style" (i.e. small loopable compositions) and would very likely be very easy to distinguish from most human-composed music. This could create a polarization of the participants' answers, with a likely bias against the system. A solution would be to have the human compose in a similar style to the generator, yet this raises the ethical question of how much should we "constrain" the human.

9.1.2 Limitations of our Mood Expression Theory

As discussed in Chapter 6, and specifically in Section 6.3, our mood expression theory has shown adequate success in expressing identifiable mood states in music. Yet, while expression of arousal seems to be almost always correctly perceived, we find some problems in the expression of valence. This problem mostly appears as difficulties in distinguishing between gradations of valence, and changes in valence expression caused by variables connected with arousal in our theory.

To properly identify the source of the problem we plan to conduct separate studies on each musical feature. Apart from allowing us to build a better model, this would also allow us to better explore the nature of the relationships between each feature and the two axes of the affect model. The main interest point would be to find out if these relationships are linear (as we have assumed until now) or not, as is more likely. It is worth noting that, even if we could properly describe the relationships between each feature and the valence/arousal axes, it is likely that interplay between features might change the affect expression.

Another limitation of this theory can be found in the model of affect that we have adopted. While Russell's bi-dimensional model of affect is the most accepted one, there is no real guarantee that it is the best fit for the capabilities of METACOMPOSE, or is the best one to describe game emotions. Since Russell described his circumplex model of affect in 1980, many more models have been proposed, often presenting different relationships between valence and arousal (Vector model [210], Positive activation - negative activation [115]), or the introduction of other dimensions (Pleasure, Arousal and Dominance emotional state model [211], Lövheim's cube of emotion [212]).

9.1.3 Limitations of the domain

Cultural limitations

A clear limitation of this work is the little cultural diversity included in our studies. The mood expression theory, as well as the theory behind METACOMPOSE, is strongly based in Western music, which makes all our results limited to the Western cultural framework. We have tried to conduct the experiment described in section 4.4 in different cultures (South Korea, Russia, and Arabic countries), yet we had little success in gathering enough data to provide a comparative analysis. Another issue of this study is that the main purpose was to allow participants to describe the expression of the clips through free-text answers, and the analysis of these kinds of answers for very different languages provided many unexpected challenges (e.g. in Korean the "word stemming" process is not really applicable).

Moreover that experiment was based on the Moody Music Generator, which created ambient music that might be culturally more easily accepted due to its lack of clear structure compared to the music created by METACOMPOSE. For this reason we believe that these experiments would tell us little (even if we had enough data) of the effect of METACOMPOSE's affect expression across multiple cultures. It is our belief that our theory would not transfer well to other cultures, given the great differences that can arise between music, both in accepted rhythms and harmony.

Perceived affect vs. Elicited affect

In this thesis we explore how to express perceivable moods in music; it has to be underlined that this is a very different problem than arousing such affective states in the listener. While research shows that listeners can perceive the emotional content of specific types of music [213], the study of elicited (or evoked) music has more ambiguous results. The main problem of this type of research is that evoked emotions happen inside the listener, making the decision of a criterion or measurement to assess the emotional state complicated. The most common measurements (often used in combination) for both perceived and elicited affect are: self-report, physiological responses, and expressive behaviour. In most studies presented in this thesis, self-report and comparative methodologies have been used. In Chapter 7, we also included physiological measures. Expressive behaviour is the observation of outwards manifestations of affective states, the most common approaches being: electromyography (EMG), head shakes observation, posture observation, and eye-tracking. Electromyography is especially interesting as it has been shown to correlate with valence [214], [215]. Nonetheless we have not included such measurements in any of the studies mostly for two reasons: EMG is fairly invasive and possibly detrimental to the experience, and such measurements require an experimental setup unsuitable for larger studies.

While we cannot show any results on elicited affect we believe that, in the games domain, perception should be the most important aspect to study. In fact there are many more factors at play than just the music while playing a game, such as aesthetics, current state of the game, game genre, game mechanics, etc.. It is worth noting that research seems to point at a correlation between perceived and elicited affect [216], yet it is clear that there is a distinction between the two, as different studies show inconclusive and conflicting relationships between the two [217], [218]. It is clear that there is still much research to do on this topic, moreover there is a disagreement over the validity of self-reporting of affective states as Zentner *et al.* hypothesize how participants might not be reporting "true" emotion, but perceived emotion [219].

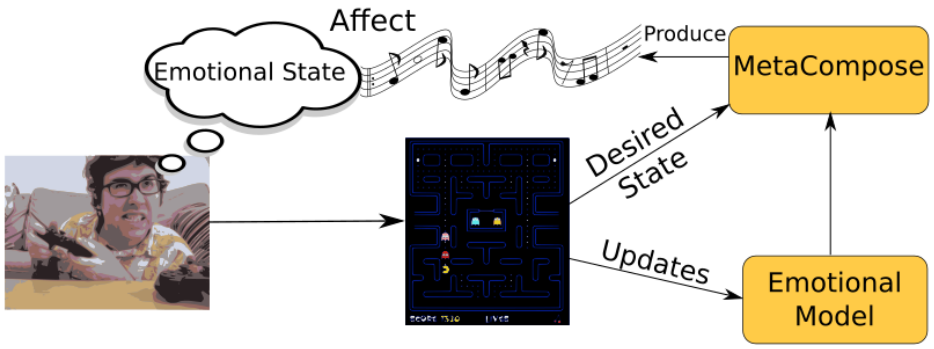

FIGURE 9.1: A player interacts with a game and through this interaction updates an emotional model and an affective music generator (MetaCompose). The game, potentially through designer annotation, tells the music generator in what emotional state it would like the player to be. At the same time an emotional model of the player estimates the current emotional state of the player. The music composer, by consulting the emotional model decides what affective expression to put in the music. Finally by listening to the music the player's emotional state is influenced.

9.2 Future work

9.2.1 Extensibility of affective music generation for games

The work described in this thesis could be applied and extended to affect the emotional state of a player. We propose a system with four main components, as can be seen in Figure 9.1: a player, a game, an emotional model, and an affective music generator (MetaCompose). The player, being human, possesses an emotional state which might vary depending on the game. Of course this emotional state depends on many external factors besides the game but, as these factors are uncontrollable, when looking at this as a closed system we assume that changes are triggered by the interaction of the player with the other components of the system. Players can get frustrated, find solace, relax, get excited, and much more by playing a game; what we would like to explore is the effect of affective music on this process, specifically if the use of specific affective expressive music could influence the player's experience and emotional state. A clear problem arising from this system is how to detect the player's emotional state; an emotional model of the player could be created through data mining techniques on a combination of in-game data, self-report, and physiological measures. This would allow us to experiment on how different expression in music might affect the player depending on her current state.

We realize that reliably evoking emotional states is a difficult (and possibly impossible) task, as other factors could overshadow the music's effect, for example the gameplay itself or the graphic aesthetics. Still, we believe that this approach would be interesting to explore fully, as it would allow us to explore many complex and interesting questions such as: how does perception of affective music change when the listener is in different emotional states, how much can music influence the user of an interactive application, and how can emotional models be used to generate affective content.

9.2.2 Extensibility of METACOMPOSE

Metacompose has been designed with a modular structure, as described in chapter 5, to allow for easy improvement of specific parts. While the system is functional and complete, many improvements to different parts of the generation or improvisational modules could be implemented.

The main improvement that the system would benefit from is better handling of transitions between one composition (or affective expression) and the next. These are currently very drastic, with clean cut transitions between different compositions, and linear gradients between parameters. While the musical product of these very simple transition mechanisms is not necessarily jarring, there is no guarantee of a pleasant transition. One of the possible solutions to this problem is for the system to generate very small *compositions* to act as "bridges" between the two compositions. This would be mainly an harmonic problem, as the two compositions could be in different keys and we would want the "bridge" to smooth this harmonic change. This problem could be approached by extending the chord sequence generation module. Of course harmony shouldn't be the only factor to consider, as voicing and rhythm would also play a part.

Another clear improvement to the chord sequence module is that the graph we currently utilize only allows for the creation of sequences in major keys. An expansion of such a graph to allow for minor keys would likely increase the "musical range" of the generator – the amount of potential musical pieces that can be generated. This extension would require some rethinking of the dissonance introduction technique, as it currently assumes that the piece is in a major key. A possible solution would be to randomly choose which degrees to alter, but that might lead to a loss of all harmonic structure. This could be mitigated by a weighted selection of degrees to alter based on harmonic distance from the original key.

When considering a practical application of METACOMPOSE, we cannot imagine that each game music (or interactive environment) composer would be happy to just use the basic improvisational modules. In fact we immediately realized that, as different applications

would require different aesthetics, it was important to give the composer a measure of control over the style of the produced music. The improvisational modules can be rewritten to change the way the basic composition is rendered, moreover the combination and number of such modules can be fully customized. We expect that this will satisfy most users to adapt METACOMPOSE to their purposes. Sadly we haven't been able to explore how composers might use this system, which at the moment will at least require a small amount of programming experience. We expect the improvisational modules class would require some re-designing to make it more accessible and easily modifiable. We discuss some possible future improvements in more detail in section 9.2.3.

Another extensibility of the improvisational modules of METACOMPOSE lies in the substitution of these through ANNs. An example already briefly discussed in the thesis would be to use modules evolved with the Primal-Improv system. While we haven't been able to explore this application yet, we expect that the structured form of METACOMPOSE's composition generation would temper the inhuman emergent music characteristics of Primal-Improv. A different approach would be to train ANNs on a specific corpus with the objective of extrapolating a specific style. This approach would create a series of challenges, especially: what would the inputs/outputs of the ANNs be, and how can these be related to the *compositions* generated by METACOMPOSE. This would be a highly complex problem, as we would need to extrapolate a structure resembling the *composition abstraction* from each musical piece in the training corpus. A better solution might be to use artificial evolution to evolve the modules and then evaluating the musical product of the modules (based on many METACOMPOSE compositions) with a specific genre or corpus. A possible way to define such evaluation function – which is a non-trivial problem, as defining the characteristics of a genre is very difficult – could be approached by training a classifier (ANN) on a corpus of representative pieces of a musical genre. This method, especially with the development of Deep Learning, has shown some good results, as discussed in section 2.4.5.

Another clear improvement for the system lies in its real-time synthesizers: currently METACOMPOSE uses the basic Java MIDI synthesizers which, apart from being pretty limited, do not produce very high quality audio. What would be extremely interesting is to give METACOMPOSE access to fully programmable synthesizers, and then including timbre-generation as part of the affective expression. Through this, the system would be able to create appropriate timbres for each affective expression required. One of the main challenges of this approach would be making the transition between timbres non disruptive to the listeners. The introduction of "transitional" compositions (as discussed previously in this section) might be used to alleviate the transition between different timbres, either through stylistic

expedients or through intermediate timbres.

A final consideration is that, while METACOMPOSE's main objective is to be used in the games domain, the current implementation presents some technical issues that limit its integration. The main problem is that it is currently implemented in Java, which is not easily integrable with the most common game engines. The decision to use this specific programming language was motivated by it being the most used language in our research group when this PhD study began. Currently an interactive application (game or other) can communicate with METACOMPOSE through the TCP/IP protocol. This way, applications written in any language are able to control the METACOMPOSE system, but this also means that the application and METACOMPOSE are two different programs running at the same time on the user's computer. Obviously this makes applications using METACOMPOSE hard to distribute and creates unnecessary complexity that would be unwanted by the developers. The best solution, although time consuming, would be to reimplement METACOMPOSE in a language more easily integrable with common game engines. Even better, the creation of a library and APIs would make the integration and use of METACOMPOSE in commercial projects more attractive and viable.

9.2.3 Composers and METACOMPOSE

An interesting question that arises from the METACOMPOSE is how well it could be used in game-production, especially how easily can the improvisational modules be rewritten or improved on by a composer. We would want to involve some composers in the process and using their feedback find out how METACOMPOSE could be improved to become a product that can be integrated in commercial applications. Currently some programming knowledge and understanding of the system architecture is needed, we assume that changes to the structure of the improvisational modules will be necessary to "hide" all parameters unnecessary to the definition of an improvisational style. Still, only through conversation with the end-users – the composers – can we properly define the level of abstraction and control needed. Moreover currently METACOMPOSE does not have a graphical user interface, as it is designed as a back-end process. We expect that a minimal interface for testing, if not even an interface for module production, will be strongly requested by the composers.

9.2.4 Extensibility of Primal-Improv: Modular Semi-Autonomous Acoustic Robots

In this section we discuss the usage of Primal-Improv as a controller for an embodied robotic system, with the primary objectives of creating a flexible robotic instrument/improviser and exploring how

users react to this type of systems. Throughout our lives we, as humans, exhibit in many different ways some aspects of creativity: we create books, music, movies, paintings, etc. The field of computational creativity has been active for decades studying how computers can also display creative behaviours or interact with humans to facilitate the creation of art (or any kind of content).

Computers are obviously very different from us, we have biological bodies that work very differently compared to the components of machines. In this field computers are often synonymous with software, little attention is given to the fact that when humans create they also exist in a physical space, while the algorithm only acts inside a virtual confine. We feel that there is great potential in embodying these creative machines in physical robots. Moreover in robotics the focus is often on achieving an objective, while disregarding that in the interaction with humans there needs to be a social aspect. The field of Computer/Human interaction has been working on such types of interaction for a long while, but the biggest efforts can be seen in humanoid robots and simulation of facial expressions [220], [221]. While these approaches are completely valid, arguably at the present time they are stuck in the "uncanny valley", the phenomenon where humans find disturbing when things look almost-human but lack some features that we recognize in each other [222]. Various examples of music making robots can be found in the literature and art community [223], [224]. Yet, these are mostly systems designed to just work for their specific task; through the extension of Primal-Improv we intend to define general design rules that would allow other researchers and the industry to make their robots more creative and social through the creation of acoustic feedback, be it music or just sounds.

The main system that we propose is a modular robot able to create music autonomously and in collaboration with a human. This system is driven by Primal-Improv, as described in Chapter 8. Each robotic module is designed to analogically create monophonic sounds; these can be achieved in different ways (strings, percussions, and less conventional methods) to create a possibly unique *ensemble*.

The system described as above is an autonomous system but, by giving the system the ability to listen to its surroundings, interaction with a user can be achieved. In this case the modules will not only be able to tell what each other is doing, but also what the human is playing. The evolutionary algorithm, by definition, will continue to try to optimize the modules to play well with each other and the human, leading to a form of guided improvisation. In this scenario the human actor can be considered as a module over which the system has no influence. An interesting aspect to explore is how this kind of interaction affects the human and his/her perception of the robot. We believe that this knowledge might be generalizable, leading to

a better understanding of the interaction relationships between humans and creative robots.

The modular design has two main objectives: one is purely artistic, allowing the human to create many different morphologies, the other is to explore scalability and integration of the system with other robots. For example, our system could be integrated in other robots lacking social skills to better give feedback to human actors.

Robotics and Modularity in Instruments

Modular robotics is a research paradigm wherein robots are constructed through assembling, usually reconfigurable, modular parts. Modularity meaning that the "robot is built from several physically independent units that encapsulate some of the complexity of their functionality" [225]. In our system this means that we encapsulate instrumental features and allow them to be connected to an instrument with a docking slot. Modular robots are categorized as being either homogeneous (one module type) or heterogeneous (multiple module types) [225]. How different compartments of a heterogeneous modular robot can work together and how these compartments – be it strings, keys, snares, drums – form effective compliances. A lot of work has been done on modular robots for various purposes but there is almost no implementation of modular robotic elements in the generation of acoustics. One prominent approach is presented by Rogers *et al.* [226] where monochord and cylindrical aerophone robotic musical instruments are independently controllable robotic units that make up an instrument ensemble. Aside from the concept of modularity in instruments, a lot of work has been done on the implementation of robotic instruments [227].

A notable robotic implementation called Mechbass comes very close to our proposed idea of the modular instrument [228]. Mechbass is composed of four bass strings that can be plucked individually. While Mechbass is a platform that can only reproduce predefined music, our system will allow for improvisation.

9.2.5 Influence on player experience in more varied contexts

The reader might have been wondering about the choice of the game of Checkers in the study described in Chapter 7: why choose this classical boardgame compared to other games? We thought Checkers would provide a good first use-case of METACOMPOSE thanks to its extremely minimal narrative and aesthetics; in fact, when comparing it to similar games like Chess, Checkers only has two types of pieces and even these pieces have weaker narrative-intrinsic names: men and kings (in some languages "ladies"). A natural next step in exploring the effect of the music generated by METACOMPOSE on

player experience would be to apply it to an opposite case: a so-called adventure game (the genre is also referred to as 'point-and-click').

We plan to conduct a similar experiment to the one described in Chapter 7 using the game briefly described in Section 4.3.2. This would mean observing how differently people would perceive the game based on three experimental settings: 1) the background music reflects and supports the in-game narrative, 2) the changes in mood expression happen randomly while during gameplay, and 3) the music never changes expression. An additional setup that would also be interesting to observe is a dynamic and adaptive music, but contrasting the narrative of the game. This study would not only give us invaluable insight in the effect of music on narrative perception, but – coupled with the Checkers experiment – would create the basis for analysing the effect of affective expressive music in more complex games.

9.3 Summary

This chapter concludes this thesis by discussing the insights obtained through our various work on exploring how dynamic and affective music generation can be achieved and used in games. During the various experiments we conducted, a number of limitations of the generality of the results obtained were found. The most important of these is the lack of a culturally diverse setting for our evaluations, and the use in METACOMPOSE of Western music conventions. This means that the results here described are specific to Western culture/music, and would likely change in a different cultural and musical setup. We have also presented in this chapter a number of possible improvements and future work that could address some of these limitations and would, in general, be interesting to pursue.

Bibliography

[1] K. Collins, "An Introduction to Procedural Music in Video Games", *Contemporary Music Review*, vol. 28, no. 1, pp. 5–15, Feb. 2009.

[2] R. Stevens and D. Raybould, *The game audio tutorial: A practical guide to creating and implementing sound and music for interactive games*. Taylor & Francis, 2013.

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

[4] V. J. Konečni, "Does music induce emotion? a theoretical and methodological analysis.", *Psychology of Aesthetics, Creativity, and the Arts*, vol. 2, no. 2, p. 115, 2008.

[5] K. Miller, "Schizophonic performance: Guitar hero, rock band, and virtual virtuosity", *Journal of the Society for American Music*, vol. 3, no. 04, pp. 395–429, 2009.

[6] D. Arsenault, "Guitar hero:" not like playing guitar at all"?", *Loading...*, vol. 2, no. 2, 2008.

[7] D. Birchfield, "Generative model for the creation of musical emotion, meaning, and form", in *Proceedings of the 2003 ACM SIGMM Workshop on Experiential Telepresence*, 2003, pp. 99–104.

[8] M. Eladhari, R. Nieuwdorp, and M. Fridenfalk, "The soundtrack of your mind: Mind music-adaptive audio for game characters", in *Proceedings of Advances in Computer Entertainment Technology*, 2006.

[9] S. R. Livingstone and A. R. Brown, "Dynamic response: Real-time adaptation for music emotion", in *Proceedings of the 2nd Australasian Conference on Interactive Entertainment*, 2005, pp. 105–111.

[10] G. N. Yannakakis and J. Togelius, "Experience-driven procedural content generation", *IEEE Transactions on Affective Computing*, vol. 2, no. 3, pp. 147–161, 2011.

[11] M. Cook, S. Colton, and J. Gow, "Automating game design in three dimensions", in *Proceedings of the AISB Symposium on AI and Games*, 2014, pp. 20–24.

[12] J. A. Russell, "A circumplex model of affect", *Journal of Personality and Social Psychology*, vol. 39, no. 6, pp. 1161–1178, 1980.

[13] M. Scirea, "Mood dependent music generator", in *Proceedings of Advances in Computer Entertainment*, 2013, pp. 626–629.

[14] M. Scirea, G. A. Barros, N. Shaker, and J. Togelius, "Smug: Scientific music generator", in *Proceedings of the Sixth International Conference on Computational Creativity June*, 2015, p. 204.

[15] M. Scirea, Y.-G. Cheong, and B. C. Bae, "Mood expression in real-time computer generated music using pure data", in *Proceedings of the International Conference on Music Perception and Cognition*, 2014.

[16] C. Darwin, "The origin of species by means of natural selection: Or, the preservation of favored races in the struggle for life", 1859.

[17] K. A. De Jong, *Evolutionary computation: A unified approach*. MIT press, 2006.

[18] M. Schoenauer and Z. Michalewicz, "Evolutionary computation at the edge of feasibility", in *International Conference on Parallel Problem Solving from Nature*, Springer, 1996, pp. 245–254.

[19] Z. Michalewicz, "Do not kill unfeasible individuals", in *Proceedings of the Fourth Intelligent Information Systems Workshop*, 1995, pp. 110–123.

[20] C. A. Coello Coello, "Constraint-handling techniques used with evolutionary algorithms", in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, ACM, 2016, pp. 563–587.

[21] A. Homaifar, C. X. Qi, and S. H. Lai, "Constrained optimization via genetic algorithms", *Simulation*, vol. 62, no. 4, pp. 242–253, 1994.

[22] K. Deb, "An efficient constraint handling method for genetic algorithms", *Computer methods in applied mechanics and engineering*, vol. 186, no. 2, pp. 311–338, 2000.

[23] G. Hadjeres and F. Pachet, "Deepbach: A steerable model for bach chorales generation", *ArXiv preprint arXiv:1612.01010*, 2016.

[24] S. O. Kimbrough, G. J. Koehler, M. Lu, and D. H. Wood, "On a feasible–infeasible two-population (fi-2pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch", *Eur. J. Operational Research*, vol. 190, no. 2, pp. 310–327, 2008.

[25] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results", *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.

[26] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii", *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[27] C. D. Rosin and R. K. Belew, "New methods for competitive coevolution", *Evolutionary computation*, vol. 5, no. 1, pp. 1–29, 1997.

[28] W. D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure", *Physica D: Nonlinear Phenomena*, vol. 42, no. 1-3, pp. 228–234, 1990.

[29] C. C. Coello and M. R. Sierra, "A coevolutionary multi-objective evolutionary algorithm", in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, IEEE, vol. 1, 2003, pp. 482–489.

[30] M. A. Potter and K. A. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents", *Evolutionary computation*, vol. 8, no. 1, pp. 1–29, 2000.

[31] P. Dahlstedt and M. G. Nordahl, "Living melodies: Coevolution of sonic communication", *Leonardo*, vol. 34, no. 3, pp. 243–248, 2001.

[32] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey", *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.

[33] K. Perlin, "An image synthesizer", *ACM Siggraph Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.

[34] L. Cardamone, D. Loiacono, and P. L. Lanzi, "Interactive evolution for the procedural generation of tracks in a high-end racing game", in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, ACM, 2011, pp. 395–402.

[35] S. Risi, J. Lehman, D. B. D'Ambrosio, R. Hall, and K. O. Stanley, "Combining search-based procedural content generation and social gaming in the petalz video game.", in *Aiide*, Citeseer, 2012.

[36] E. J. Hastings, R. K. Guha, and K. O. Stanley, "Automatic content generation in the galactic arms race video game", *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 4, pp. 245–263, 2009.

[37] P. Pasquier, A. Eigenfeldt, O. Bown, and S. Dubnov, "An introduction to musical metacreation", *Computers in Entertainment (CIE)*, vol. 14, no. 2, p. 2, 2016.

[38] M. Peter, "Milieus of creativity: The role of places, environments, and spatial contexts", in *Milieus of Creativity*, Springer, 2009, pp. 97–153.

[39] M. A. Boden, *The creative mind: Myths and mechanisms*. Psychology Press, 2004.

[40] O. Bown, "Generative and adaptive creativity: A unified approach to creativity in nature, humans and machines", in *Computers and creativity*, Springer, 2012, pp. 361–381.

[41] G. Ritchie, "Some empirical criteria for attributing creativity to a computer program", *Minds and Machines*, vol. 17, no. 1, pp. 67–99, 2007.

[42] S. Colton, "Creativity versus the perception of creativity in computational systems.", in *AAAI spring symposium: Creative intelligent systems*, vol. 8, 2008.

[43] O. Bown, "Empirically grounding the evaluation of creative systems: Incorporating interaction design.", in *ICCC*, 2014, pp. 112–119.

[44] S. Colton, J. W. Charnley, and A. Pease, "Computational creativity theory: The face and idea descriptive models.", in *ICCC*, 2011, pp. 90–95.

[45] G. A. Wiggins, "A preliminary framework for description, analysis and comparison of creative systems", *Knowledge-Based Systems*, vol. 19, no. 7, pp. 449–458, 2006.

[46] E. R. Miranda, *Readings in music and artificial intelligence*. Routledge, 2013, vol. 20.

[47] P. Doornbusch, "A brief survey of mapping in algorithmic composition", in *Proceedings of the International Computer Music Conference*, 2002.

[48] M. Edwards, "Algorithmic Composition: Computational Thinking in Music", *Commun. ACM*, vol. 54, no. 7, pp. 58–67, Jul. 2011. (visited on 07/11/2016).

[49] D. Cope, "Algorithmic Music Composition", en, in *Patterns of Intuition*, G. Nierhaus, Ed., Springer Netherlands, 2015, pp. 405–416. (visited on 07/11/2016).

[50] A. Kirke and E. R. Miranda, "A Survey of Computer Systems for Expressive Music Performance", *ACM Comput. Surv.*, vol. 42, no. 1, 3:1–3:41, Dec. 2009.

[51] A. Alpern, "Techniques for algorithmic composition of music", *On the web: http://hamp.hampshire.edu/adaF92/algocomp/algocomp*, vol. 95, p. 120, 1995. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.9364&rep=rep1&type=pdf (visited on 07/11/2016).

[52] R. Wooller, A. R. Brown, E. Miranda, J. Diederich, and R. Berry, "A framework for comparison of process in algorithmic music systems", in *Generative Arts Practice 2005 — A Creativity & Cognition Symposium*, 2005.

[53] S. Abrams, D. V. Oppenheim, D. Pazel, J. Wright, *et al.*, "Higher-level composition control in music sketcher: Modifiers and smart harmony", in *Proceedings of the ICMC*, Citeseer, 1999.

[54] J. Biles, "Genjam: A genetic algorithm for generating jazz solos", in *Proceedings of the International Computer Music Conference*, INTERNATIONAL COMPUTER MUSIC ACCOCIATION, 1994, pp. 131–131.

[55] M. Puckette *et al.*, "Pure data: Another integrated computer music environment", *Proceedings of the Second Intercollege Computer Music Concerts*, pp. 37–41, 1996.

[56] J. Robertson, A. de Quincey, T. Stapleford, and G. Wiggins, "Real-time music generation for a virtual environment", in *Proceedings of ECAI-98 Workshop on AI/Alife and Entertainment*, Citeseer, 1998.

[57] A. Smaill, G. Wiggins, and M. Harris, "Hierarchical music representation for composition and analysis", *Computers and the Humanities*, vol. 27, no. 1, pp. 7–17, 1993.

[58] G. Wiggins, M. Harris, and A. Smaill, *Representing music for analysis and composition*. University of Edinburgh, Department of Artificial Intelligence, 1990.

[59] K. Monteith, T. Martinez, and D. Ventura, "Automatic generation of music for inducing emotive response", in *Proceedings of the International Conference on Computational Creativity*, Citeseer, 2010, pp. 140–149.

[60] H. Chan and D. A. Ventura, "Automatic composition of themed mood pieces", 2008.

[61] R. Loughran, J. McDermott, and M. O'Neill, "Tonality driven piano compositions with grammatical evolution", in *IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2015, pp. 2168–2175.

[62] P. Dahlstedt, "Autonomous evolution of complete piano pieces and performances", in *Proceedings of Music AL Workshop*, Citeseer, 2007.

[63] E. R. Miranda and A. Biles, *Evolutionary computer music*. Springer, 2007.

[64] A. P. Rigopulos and E. B. Egozy, *Real-time music creation system*, US Patent 5,627,335, May 1997.

[65] S. K. Meier and J. L. Briggs, *System for real-time music composition and synthesis*, US Patent 5,496,962, Mar. 1996.

[66] A. Horner and D. E. Goldberg, "Genetic algorithms and computer-assisted music composition", *Urbana*, vol. 51, no. 61801, pp. 437–441, 1991.

[67] R. A. McIntyre, "Bach in a box: The evolution of four part baroque harmony using the genetic algorithm", in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, IEEE, 1994, pp. 852–857.

[68] G. Papadopoulos and G. Wiggins, "A genetic algorithm for the generation of jazz melodies", *Proceedings of STEP*, vol. 98, 1998.

[69] G. Wiggins, G. Papadopoulos, S. Phon-Amnuaisuk, and A. Tuson, "Evolutionary methods for musical composition", *Dai Research Paper*, 1998.

[70] B. Jacob, "Composing with genetic algorithms", 1995.

[71] D. Horowitz, "Generating rhythms with genetic algorithms", in *AAAI*, Cambridge, MA, vol. 94, 1994, p. 1459.

[72] D. Ralley, "Genetic algorithms as a tool for melodic development", 1995.

[73] A. K. Hoover, P. A. Szerlip, and K. O. Stanley, "Functional scaffolding for composing additional musical voices", *Computer Music Journal*, 2014.

[74] ——, "Interactively evolving harmonies through functional scaffolding", in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, ACM, 2011, pp. 387–394.

[75] C. Ames, "The markov process as a compositional model: A survey and tutorial", *Leonardo*, pp. 175–187, 1989.

[76] K. Jones, "Compositional applications of stochastic processes", *Computer Music Journal*, vol. 5, no. 2, pp. 45–61, 1981.

[77] D. Zicarelli, "M and jam factory", *Computer Music Journal*, vol. 11, no. 4, pp. 13–29, 1987.

[78] C. Ames and M. Domino, "Cybernetic composer: An overview", in *Understanding music with AI*, MIT Press, 1992, pp. 186–205.

[79] J. Pressing, "Nonlinear maps as generators of musical design", *Computer Music Journal*, vol. 12, no. 2, pp. 35–46, 1988.

[80] S. Bell and L. Gabora, "A music-generating system inspired by the science of complex adaptive systems", *ArXiv preprint arXiv:1610.02475*, 2016.

[81] M. J. Steedman, "Grammar, interpretation, and processing from the lexicon", in *Lexical representation and process*, MIT Press, 1989, pp. 463–504.

[82] G. Rozenberg and A. Salomaa, *The mathematical theory of l systems*. Academic press, 1980, vol. 90.

[83] D. Cope and M. J. Mayer, *Experiments in musical intelligence*. AR editions Madison, WI, 1996, vol. 12.

[84] C. Roads and P. Wieneke, "Grammars as representations for music", *Computer Music Journal*, pp. 48–55, 1979.

[85] K. Ebcioğlu, "An expert system for harmonizing four-part chorales", *Computer Music Journal*, vol. 12, no. 3, pp. 43–51, 1988.

[86] P. Tsang and M. Aitken, "Harmonizing music as a discipline of constraint logic programming.", in *Proceedings of the International Computer Music Conference*, 1991.

[87] F. Pachet and P. Roy, "Formulating constraint satisfaction problems on part-whole relations: The case of automatic musical harmonization", in *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI 98)*, 1998, pp. 1–11.

[88] ——, "Musical harmonization with constraints: A survey", *Constraints*, vol. 6, no. 1, pp. 7–19, 2001.

[89] F. Pachet, G. Ramalho, and J. Carrive, "Representing temporal musical objects and reasoning in the muses system", *Journal of new music research*, vol. 25, no. 3, pp. 252–275, 1996.

[90] G. Ramalho and J.-G. Ganascia, "Simulating creativity in jazz performance", in *AAAI*, vol. 94, 1994, pp. 108–113.

[91] P. M. Todd, "A connectionist approach to algorithmic composition", *Computer Music Journal*, vol. 13, no. 4, pp. 27–43, 1989.

[92] M. C. Mozer, "Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing", *Connection Science*, vol. 6, no. 2-3, pp. 247–280, 1994.

[93] M. I. Bellgard and C. P. Tsang, "Harmonizing music the boltzmann way", *Connection Science*, vol. 6, no. 2-3, pp. 281–297, 1994.

[94] P. Toiviainen, *Symbolic ai versus connectionism in music research.* 2000.

[95] S. M. Schwanauer and D. A. Levitt, *Machine models of music*. MIT Press, 1993.

[96] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[97] A. R. Burton and T. Vladimirova, "Genetic algorithm utilising neural network fitness evaluation for musical composition", in *Artificial Neural Nets and Genetic Algorithms*, Springer, 1998, pp. 219–223.

[98] L. Spector and A. Alpern, "Induction and recapitulation of deep musical structure", in *Proceedings of International Joint Conference on Artificial Intelligence, IJCAI*, vol. 95, 1995, pp. 20–25.

[99] H. Hild, J. Feulner, and W. Menzel, "Harmonet: A neural net for harmonizing chorales in the style of js bach", in *Advances in neural information processing systems*, 1992, pp. 267–274.

[100] J. Feulner, "Neural networks that learn and reproduce various styles of harmonization", in *Proceedings of the International Computer Music Conference*, INTERNATIONAL COMPUTER MUSIC ACCOCIATION, 1993, pp. 236–236.

[101] T. Blackwell, O. Bown, and M. Young, "Live algorithms: Towards autonomous computer improvisers", in *Computers and Creativity*, Springer, 2012, pp. 147–174.

[102] M. Young, "Nn music: Improvising with a 'living'computer", in *International Symposium on Computer Music Modeling and Retrieval*, Springer, 2007, pp. 337–350.

[103] O. Bown and S. Lexer, "Continuous-time recurrent neural networks for generative and interactive musical performance", in *Workshops on Applications of Evolutionary Computation*, Springer, 2006, pp. 652–663.

[104] J. M. Bradshaw, R. R. Hoffman, D. D. Woods, and M. Johnson, "The seven deadly myths of" autonomous systems"", *IEEE Intelligent Systems*, vol. 28, no. 3, pp. 54–61, 2013.

[105] D. Conklin, "Music generation from statistical models", in *Proceedings of the AISB 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, AISB Society London, 2003, pp. 30–35.

[106] G. Nierhaus, *Algorithmic composition: Paradigms of automated music generation*. Springer Science & Business Media, 2009.

[107] P. M. Todd and G. M. Werner, "Frankensteinian methods for evolutionary music", *Musical networks: Parallel distributed perception and performace*, pp. 313–340, 1999.

[108] C. G. Lange and W. James, *The emotions*. Williams & Wilkins, 1922, vol. 1.

[109] K. R. Scherer, A. Schorr, and T. Johnstone, *Appraisal processes in emotion: Theory, methods, research*. Oxford University Press, 2001.

[110] P. Ekman, "An argument for basic emotions", *Cognition & emotion*, vol. 6, no. 3-4, pp. 169–200, 1992.

[111] ——, "Are there basic emotions?", 1992.

[112] S. S. Tomkins, *Affect imagery consciousness: Volume i: The positive affects*. Springer publishing company, 1962, vol. 1.

[113] J. Posner, J. A. Russell, and B. S. Peterson, "The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology", *Development and psychopathology*, vol. 17, no. 03, pp. 715–734, 2005.

[114] H. Schlosberg, "Three dimensions of emotion.", *Psychological review*, vol. 61, no. 2, p. 81, 1954.

[115] D. Watson and A. Tellegen, "Toward a consensual structure of mood.", *Psychological bulletin*, vol. 98, no. 2, p. 219, 1985.

[116] R. S. Lazarus, *Emotion and adaptation.* Oxford University Press, 1991.

[117] C. R. Brewin, "Cognitive change processes in psychotherapy.", *Psychological review*, vol. 96, no. 3, p. 379, 1989.

[118] J. S. Lerner and D. Keltner, "Beyond valence: Toward a model of emotion-specific influences on judgement and choice", *Cognition & Emotion*, vol. 14, no. 4, pp. 473–493, 2000.

[119] B. A. Martin, "The influence of gender on mood effects in advertising", *Psychology & Marketing*, vol. 20, no. 3, pp. 249–273, 2003.

[120] C. D. Batson, L. L. Shaw, and K. C. Oleson, "Differentiating affect, mood, and emotion: Toward functionally based conceptual distinctions.", 1992.

[121] C. Beedie, P. Terry, and A. Lane, "Distinctions between emotion and mood", *Cognition & Emotion*, vol. 19, no. 6, pp. 847–878, 2005.

[122] K. Hevner, "Expression in music: A discussion of experimental studies and theories.", *Psychological review*, vol. 42, no. 2, p. 186, 1935.

[123] L. B. Meyer, *Music, the arts, and ideas: Patterns and predictions in twentieth-century culture*. University of Chicago Press, 2010.

[124] M. Clynes, "Communication and generation of emotion through essentic form", *Emotions-Their parameters and measurement, ed. L. Levi*, pp. 561–602, 1975.

[125] M. Clynes and N. Nettheim, "The living quality of music", in *Music, mind, and brain*, Springer, 1982, pp. 47–82.

[126] M. Budd *et al.*, *Music and the emotions: The philosophical theories*. Routledge, 2002.

[127] H. Katayose, M. Imai, and S. Inokuchi, "Sentiment extraction in music", in *Proceedings of the 9th International Conference on Pattern Recognition*, 1988, pp. 1083–1087.

[128] R. E. Thayer, *The biopsychology of mood and arousal*. Oxford University Press, 1989.

[129] W. Wundt, *Outlines of psychology*. Springer, 1980.

[130] G. Kreutz, U. Ott, D. Teichmann, P. Osawa, and D. Vaitl, "Using music to induce emotions: Influences of musical preference and absorption", *Psychology of Music*, vol. 36, no. 1, pp. 101–126, 2008.

[131] E. Lindström, P. N. Juslin, R. Bresin, and A. Williamon, "'Expressivity comes from within your soul": A questionnaire study of music students' perspectives on expressivity", *Research Studies in Music Education*, vol. 20, no. 1, pp. 23–47, 2003.

[132] R. H. Gundlach, "Factors determining the characterization of musical phrases", *The American Journal of Psychology*, vol. 47, no. 4, pp. 624–643, 1935.

[133] M. G. Rigg, "Speed as a determiner of musical mood.", *Journal of Experimental Psychology*, vol. 27, no. 5, p. 566, 1940.

[134] K. Hevner, "Experimental studies of the elements of expression in music", *The American Journal of Psychology*, vol. 48, no. 2, pp. 246–268, 1936.

[135] L. Wedin, "A multidimensional study of perceptual-emotional qualities in music", *Scandinavian journal of psychology*, vol. 13, no. 1, pp. 241–257, 1972.

[136] K. B. Watson, "The nature and measurement of musical meanings.", *Psychological Monographs*, vol. 54, no. 2, p. i, 1942.

[137] K. R. Scherer and J. S. Oshinsky, "Cue utilization in emotion attribution from auditory stimuli", *Motivation and emotion*, vol. 1, no. 4, pp. 331–346, 1977.

[138] J. I. Alpert and M. I. Alpert, "Music influences on mood and purchase intentions", *Psychology & Marketing*, vol. 7, no. 2, pp. 109–133, 1990.

[139] G. C. Bruner, "Music, mood, and marketing", *The Journal of marketing*, pp. 94–104, 1990.

[140] G. N. Yannakakis, P. Spronck, D. Loiacono, and E. André, "Player modeling", in *Dagstuhl Follow-Ups*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, vol. 6, 2013.

[141] R. W. Picard, "Affective computing: Challenges", *International Journal of Human-Computer Studies*, vol. 59, no. 1, pp. 55–64, 2003.

[142] J. L. Andreassi, *Psychophysiology: Human behavior & physiological response*. Psychology Press, 2013.

[143] L.-O. Lundqvist, F. Carlsson, P. Hilmersson, and P. N. Juslin, "Emotional responses to music: Experience, expression, and physiology", *Psychology of music*, vol. 37, no. 1, pp. 61–90, 2009.

[144] G. E. Schwartz, P. L. Fair, P. Salt, M. R. Mandel, and G. L. Klerman, "Facial muscle patterning to affective imagery in depressed and nondepressed subjects", *Science*, vol. 192, no. 4238, pp. 489–491, 1976.

[145] P. Ekman and E. L. Rosenberg, *What the face reveals: Basic and applied studies of spontaneous expression using the facial action coding system (facs)*. Oxford University Press, USA, 1997.

[146] G. Tom, P. Pettersen, T. Lau, T. Burton, and J. Cook, "The role of overt head movement in the formation of affect", *Basic and Applied Social Psychology*, vol. 12, no. 3, pp. 281–289, 1991.

[147] J. M. Kivikangas, G. Chanel, B. Cowley, I. Ekman, M. Salminen, S. Järvelä, and N. Ravaja, "A review of the use of psychophysiological methods in game research", *Journal of gaming & virtual worlds*, vol. 3, no. 3, pp. 181–199, 2011.

[148] P. Hoffert, *Music for new media: Composing for videogames, web sites, presentations, and other interactive media*. Berklee Press, 2007.

[149] P. Langston, "Six techniques for algorithmic music composition", in *Proceedings of the International Computer Music Conference*, 1989.

[150] S. A. Hedges, "Dice music in the eighteenth century", *Music & Letters*, vol. 59, no. 2, pp. 180–187, 1978.

[151] J. D. Kramer, "New temporalities in music", *Critical Inquiry*, vol. 7, no. 3, pp. 539–556, 1981.

[152] D. Bernstein, *Creating an interactive audio environment*, Gamasutra, http://www.gamasutra.com/view/feature/131646/creating_an_interactive_audio_.php, [Online; accessed 27-August-2017], 1997.

[153] B. A. Lagim, "The music of anarchy online: Creating music for mmogs", *Gamasutra. The Art &Business of making Games. Posted on September*, vol. 16, p. 2002, 2002.

[154] D. Brown, "Mezzo: An adaptive, real-time composition program for game soundtracks", in *Proceedings of the AIIDE 2012 Workshop on Musical Metacreation*, 2012, pp. 68–72.

[155] W. Mkaouer, M. Kessentini, A. Shaout, P. Koligheu, S. Bechikh, K. Deb, and A. Ouni, "Many-objective software remodularization using nsga-iii", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 24, no. 3, p. 17, 2015.

[156] A. Liapis, G. N. Yannakakis, and J. Togelius, "Sentient sketchbook: Computer-aided game level authoring.", in *FDG*, 2013, pp. 213–220.

[157] ——, "Adapting models of visual aesthetics for personalized content creation", *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 3, pp. 213–228, 2012.

[158] ——, "Neuroevolutionary constrained optimization for content creation", in *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, IEEE, 2011, pp. 71–78.

[159] K. Deb, A. Pratap, and T. Meyarivan, "Constrained test problems for multi-objective evolutionary optimization", in *Evolutionary Multi-Criterion Optimization*, Springer, 2001, pp. 284–298.

[160] D. Chafekar, J. Xuan, and K. Rasheed, "Constrained multi-objective optimization using steady state genetic algorithms", in *Genetic and Evolutionary Computation—GECCO 2003*, Springer, 2003, pp. 813–824.

[161] F. Jimenez, A. F. Gómez-Skarmeta, G. Sánchez, and K. Deb, "An evolutionary algorithm for constrained multi-objective optimization", in *Proceedings of the Congress on Evolutionary Computation*, IEEE, 2002, pp. 1133–1138.

[162] A. Isaacs, T. Ray, and W. Smith, "Blessings of maintaining infeasible solutions for constrained multi-objective optimization problems", in *IEEE Congress on Evolutionary Computation*, IEEE, 2008, pp. 2780–2787.

[163] F. Jiménez and J. L. Verdegay, "Constrained multiobjective optimization by evolutionary algorithms", in *University of La*, Citeseer, 1998.

[164] T. Ray, K. TAI, and K. C. SEOW, "Multiobjective design optimization by an evolutionary algorithm", *Engineering Optimization*, vol. 33, no. 4, pp. 399–424, 2001.

[165] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[166] T. Kohonen, "The self-organizing map", *Neurocomputing*, vol. 21, no. 1, pp. 1–6, 1998.

[167] P. J. Werbos, "Backpropagation through time: What it does and how to do it", *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[168] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[169] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development", *Genetic programming and evolvable machines*, vol. 8, no. 2, pp. 131–162, 2007.

[170] P. A. Szerlip, A. K. Hoover, and K. O. Stanley, "Maestrogenesis: Computer-assisted musical accompaniment generation", 2012.

[171] B. Þ. Jónsson, A. K. Hoover, and S. Risi, "Interactively evolving compositional sound synthesis networks", in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ACM, 2015, pp. 321–328.

[172] J. Secretan, N. Beato, D. B. D Ambrosio, A. Rodriguez, A. Campbell, and K. O. Stanley, "Picbreeder: Evolving pictures collaboratively online", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2008, pp. 1759–1768.

[173] J. Clune and H. Lipson, "Evolving three-dimensional objects with a generative encoding inspired by developmental biology", in *Proceedings of the European Conference on Artificial Life*, 2011, pp. 144–148.

[174] E. J. Hastings, R. K. Guha, and K. O. Stanley, "Evolving content in the galactic arms race video game", in *2009 IEEE Symposium on Computational Intelligence and Games*, IEEE, 2009, pp. 241–248.

[175] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, "Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding", in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, ACM, 2013, pp. 167–174.

[176] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies", *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[177] F. Silva, P. Urbano, S. Oliveira, and A. L. Christensen, "Odneat: An algorithm for distributed online, onboard evolution of robot behaviours", *Artificial Life*, vol. 13, pp. 251–258, 2012.

[178] M. Brinkmann, *Pure data patches*, http://www.martin-brinkmann.de/.

[179] A. Mehrabian, "Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament", *Current Psychology*, vol. 14, no. 4, pp. 261–292, 1996.

[180] S. B. Chatman, *Story and discourse: Narrative structure in fiction and film*. Cornell University Press, 1980.

[181] G. Genette and J. E. Lewin, *Narrative discourse: An essay in method*. Cornell University Press, 1983.

[182] R. Barthes, *S/z*. Collins Publishers, 1974.

[183] A. Rollings and E. Adams, "Fundamentals of game design", *New Challenges for Character-Based AI for Games. Chapter 20: Artificial Life and Puzzle Games. Prentice Hall*, pp. 573–590, 2006.

[184] R. Likert, "A technique for the measurement of attitudes.", *Archives of psychology*, 1932.

[185] S. Mugglin, *Chord charts and maps*, http://mugglinworks.com/chordmaps/chartmaps.htm, Accessed: 2015-09-14.

[186] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001, vol. 16.

[187] T. Byron and C. Stevens, "Steps and leaps in human memory for melodies: The ef-fect of pitch interval magnitude in a melodic contour dis-crimination task", in *9th International Conference on Music Perception and Cognition (ICMPC9), Bologna, Italy*, Citeseer, 2006.

[188] M. Scirea, J. Togelius, P. Eklund, and S. Risi, "Metacompose: A compositional evolutionary music composer", in *International Conference on Evolutionary and Biologically Inspired Music and Art*, Springer, 2016, pp. 202–217.

[189] G. T. Toussaint *et al.*, "The euclidean algorithm generates traditional musical rhythms", in *Proceedings of BRIDGES: Mathematical Connections in Art, Music and Science*, 2005, pp. 47–56.

[190] E. Bjorklund, "A metric for measuring the evenness of timing system rep-rate patterns", *SNS ASD Tech Note, SNS-NOTE-CNTRL-100*, 2003.

[191] D. Liu, L. Lu, and H.-J. Zhang, "Automatic mood detection from acoustic music data", in *Proceedings of the International Symposium on Music Information Retrieval*, 2003, pp. 81–7.

[192] J.-J. Aucouturier, F. Pachet, and M. Sandler, ""' the way it sounds": Timbre models for analysis and retrieval of music signals", *Multimedia, IEEE Transactions on*, vol. 7, no. 6, pp. 1028–1035, 2005.

[193] T. Langlois and G. Marques, "A music classification method based on timbral features.", in *ISMIR*, 2009, pp. 81–86.

[194] J. M. Grey and J. W. Gordon, "Perceptual effects of spectral modifications on musical timbres", *The Journal of the Acoustical Society of America*, vol. 63, no. 5, pp. 1493–1500, 1978.

[195] C. P. E. Bach, W. J. Mitchell, and W. John, *Essay on the true art of playing keyboard instruments*. WW Norton, 1949.

[196] L. B. Meyer, *Emotion and meaning in music*. University of Chicago Press, 2008.

[197] L. J. Trainor and B. M. Heinmiller, "The development of evaluative responses to music:: Infants prefer to listen to consonance over dissonance", *Infant Behavior and Development*, vol. 21, no. 1, pp. 77–88, 1998.

[198] G. Husain, W. F. Thompson, and E. G. Schellenberg, "Effects of musical tempo and mode on arousal, mood, and spatial abilities", *Music Perception: An Interdisciplinary Journal*, vol. 20, no. 2, pp. 151–171, 2002.

[199] B. De Haas, R. C. Veltkamp, and F. Wiering, "Tonal pitch step distance: A similarity measure for chord progressions.", in *ISMIR*, 2008, pp. 51–56.

[200] F. Lerdahl, "Tonal pitch space", *Music Perception*, pp. 315–349, 1988.

[201] H. P. Martinez, G. N. Yannakakis, and J. Hallam, "Don't classify ratings of affect; rank them!", *Affective Computing, IEEE Transactions on*, vol. 5, no. 3, pp. 314–326, 2014.

[202] G. Perle, *Serial composition and atonality: An introduction to the music of schoenberg, berg, and webern*. Univ of California Press, 1972.

[203] M. Scirea, M. J. Nelson, and J. Togelius, "Moody music generator: Characterising control parameters using crowdsourcing", in *Evolutionary and Biologically Inspired Music, Sound, Art and Design*, Springer, 2015, pp. 200–211.

[204] R. Rosenthal and D. B. Rubin, "A simple, general purpose display of magnitude of experimental effect.", *Journal of educational psychology*, vol. 74, no. 2, p. 166, 1982.

[205] A. Clerico, C. Chamberland, M. Parent, P.-E. Michon, S. Tremblay, T. Falk, J.-C. Gagnon, and P. Jackson, "Biometrics and classifier fusion to predict the fun-factor in video gaming", in *IEEE Conference on Computational Intelligence and Games*, IEEE, 2016, pp. 233–240.

[206] J. Schaeffer, R. Lake, P. Lu, and M. Bryant, "Chinook the world man-machine checkers champion", *AI Magazine*, vol. 17, no. 1, p. 21, 1996.

[207] J. Schaeffer, N. Burch, Y. Björnsson, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen, "Checkers is solved", *Science*, vol. 317, no. 5844, pp. 1518–1522, 2007.

[208] J. Schaeffer, "Game over: Black to play and draw in checkers.", *ICGA Journal*, vol. 30, no. 4, pp. 187–197, 2007.

[209] E. Miranda, *Composing music with computers*. CRC Press, 2001.

[210] M. M. Bradley, M. K. Greenwald, M. C. Petry, and P. J. Lang, "Remembering pictures: Pleasure and arousal in memory", *Journal of experimental psychology: Learning, Memory, and Cognition*, vol. 18, no. 2, pp. 379–390, 1992.

[211] A. Mehrabian, "Basic dimensions for a general psychological theory implications for personality, social, environmental, and developmental studies", 1980.

[212] H. Lövheim, "A new three-dimensional model for emotions and monoamine neurotransmitters", *Medical hypotheses*, vol. 78, no. 2, pp. 341–348, 2012.

[213] K. R. Scherer and M. R. Zentner, "Emotional effects of music: Production rules", *Music and emotion: Theory and research*, pp. 361–392, 2001.

[214] J. A. Sloboda and P. N. Juslin, "Psychological perspectives on music and emotion.", 2001.

[215] J. A. Sloboda and S. A. O'Neill, "Emotions in everyday listening to music.", 2001.

[216] P. G. Hunter, E. G. Schellenberg, and U. Schimmack, "Feelings and perceptions of happiness and sadness induced by music: Similarities, differences, and mixed emotions.", *Psychology of Aesthetics, Creativity, and the Arts*, vol. 4, no. 1, p. 47, 2010.

[217] S. O. Ali and Z. F. Peynircioğğlu, "Intensity of emotions conveyed and elicited by familiar and unfamiliar music", *Music Perception: An Interdisciplinary Journal*, vol. 27, no. 3, pp. 177–182, 2010.

[218] K. Kallinen and N. Ravaja, "Emotion perceived and emotion felt: Same and different", *Musicae Scientiae*, vol. 10, no. 2, pp. 191–213, 2006.

[219] M. Zentner, D. Grandjean, and K. R. Scherer, "Emotions evoked by the sound of music: Characterization, classification, and measurement.", *Emotion*, vol. 8, no. 4, p. 494, 2008.

[220] H. Miwa, K. Itoh, M. Matsumoto, M. Zecca, H. Takanobu, S. Rocella, M. C. Carrozza, P. Dario, and A. Takanishi, "Effective emotional expressions with expression humanoid robot we-4rii: Integration of humanoid robot hand rch-1", in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, IEEE, vol. 3, 2004, pp. 2203–2208.

[221] T. Hashimoto, S. Hitramatsu, T. Tsuji, and H. Kobayashi, "Development of the face robot saya for rich facial expressions", in *2006 SICE-ICASE International Joint Conference*, IEEE, 2006, pp. 5423–5428.

[222] M. Mori, K. F. MacDorman, and N. Kageki, "The uncanny valley [from the field]", *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 98–100, 2012.

[223] A. Lim, T. Mizumoto, L.-K. Cahier, T. Otsuka, T. Takahashi, K. Komatani, T. Ogata, and H. G. Okuno, "Robot musical accompaniment: Integrating audio and visual cues for real-time synchronization with a human flutist", in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, IEEE, 2010, pp. 1964–1969.

[224] T. Mizumoto, H. Tsujino, T. Takahashi, T. Ogata, and H. G. Okuno, "Thereminist robot: Development of a robot theremin player with feedforward and feedback arm control based on a theremin's pitch model", in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2009, pp. 2297–2302.

[225] K. Stoy, D. Brandt, D. J. Christensen, and D. Brandt, *Self-reconfigurable robots: An introduction*. Mit Press Cambridge, 2010.

[226] T. Rogers, T. Kemper, and S. Barton, "Marie: Monochord-aerophone robotic instrument ensemble", in *15th International Conference on New Interfaces for Musical Expression (NIME)*, 2015, pp. 408–411.

[227] A. Kapur, "A history of robotic musical instruments", in *Proceedings of the International Computer Music Conference*, Citeseer, 2005, pp. 21–28.

[228] J. McVay, D. A. Carnegie, J. W. Murphy, and A. Kapur, "Mechbass: A systems overview of a new four-stringed robotic bass guitar", *School of Engineering Victoria University, Tech. Rep*, 2012.