A Functional Taxonomy of Music Generation Systems

DORIEN HERREMANS, Singapore University of Technology and Design & Queen Mary University of London
CHING-HUA CHUAN, University of North Florida
ELAINE CHEW, Queen Mary University of London

Digital advances have transformed the face of automatic music generation since its beginnings at the dawn of computing. Despite the many breakthroughs, issues such as the musical tasks targeted by different machines and the degree to which they succeed remain open questions. We present a functional taxonomy for music generation systems with reference to existing systems. The taxonomy organizes systems according to the purposes for which they were designed. It also reveals the inter-relatedness amongst the systems. This design-centered approach contrasts with predominant methods-based surveys, and facilitates the identification of grand challenges so as to set the stage for new breakthroughs.

CCS Concepts: •**Applied computing** → **Sound and music computing;** •**Information systems** → *Multimedia information systems;* •**Computing methodologies** → *Artificial intelligence;* Machine learning;

Additional Key Words and Phrases: music generation, taxonomy, functional survey, survey, automatic composition, algorithmic composition

ACM Reference Format:
Dorien Herremans, Ching-Hua Chuan and Elaine Chew, 2016. A Functional Taxonomy of Music Generation Systems. *ACM Comput. Surv.* 50, 5, Article 69 (September 2017), 33 pages.
DOI: 10.1145/3108242

1. INTRODUCTION

The history of automatic music generation is almost as old as that of computers. That machines can one day generate "elaborate and scientific pieces of music of any degree of complexity and extent" [Lovelace 1843] was anticipated by visionaries such as Ada Lovelace since the initial designs for a general purpose computing device were laid down by Charles Babbage. Indeed, music generation or automated composition was a task accomplished by one of the first computers built, the ILLIAC I [Hiller Jr and Isaacson 1957]. Today, computer-based composition systems are aplenty. The recent announcement of Google Magenta[1], "a research project to advance the state of the art in machine intelligence for music and art generation," underscores the importance and popularity of automatic music generation in artificial intelligence.

[1]http://magenta.tensorflow.org/welcome-to-magenta

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 658914.
Author's addresses: D. Herremans, Information Systems Technology and Design Pillar, Singapore University of Technology and Design, Singapore University of Technology & Design, 8 Somapah Road, 1.502-18, Singapore 487372, for part of the work, D. Herremans was at the School of Electronic Engineering and Computer Science, Queen Mary University of London; E. Chew, School of Electronic Engineering and Computer Science, Queen Mary University of London, Mile End Road, E1 NS4 London, UK; C.-H. Chuan, School of Computing, University of North Florida, 1 UNF Drive, Jacksonville, FL 32224, US.

DOI: 10.1145/3108242

Despite the enthusiasm of researchers, using computers to generate music remains an ill-defined problem. Although several survey papers on automatic music generation [Papadopoulos and Wiggins 1999; Nierhaus 2009; Fernández and Vico 2013] exist, researchers still debate the kinds of musical tasks that can be performed by machines and the degree to which satisfactory outcomes can be achieved. Outstanding questions include: what compositional tasks are solved and which remain challenges? How is each compositional task modeled and how do they connect to each other? What is the relationship between systems proposed for different compositional tasks? What is the goal defined for each task and how can the objective be quantified? How are the systems evaluated? While individual questions or subsets of these questions might be addressed in specific papers, previous surveys fail to provide a systematic comparison of the state of the art.

This paper aims to answer these questions by proposing a functional taxonomy of automatic music generation systems. Focusing on the purpose for which the systems were developed, we examine the manner in which each music composition task was modeled and describe the connection between different tasks within and across systems. We propose a concept map for automatic music generation systems based on the functions of the systems in Section 1.1. A brief history of early automatic music generation systems is provided in Section 1.2, followed by a discussion on the general approach to evaluating computer generated music (Section 1.3). A detailed survey of systems designed based on each functional aspect is then presented in Section 2.

1.1. Function and design concepts in automatic music generation systems

The complexity and types of music generation systems is almost as varied as music itself. It would be a gross simplification to consider and judge all automatic music generation systems in a homogeneous fashion. The easiest way to understand the complexity of these systems and their connections one to another is to examine the functions for which they were designed.

Figure 1 illustrates a concept map showing the functional design aspects that form the proposed taxonomy of music generation systems. The map is centered around two basic concepts crucial to music generation systems: the *composition* (the higher grey node) and the *note* (the lower gray node), which possesses properties such as pitch, duration, onset time, and instrumentation.

Between the *note* and the *composition* lie four essential elements of music composition: *melody*, *harmony*, *rhythm*, and *timbre*. Systems that focus on any one of the four aspects generate a sequence of notes that fulfills a specific set of goals, which can vary widely amongst the systems. For example, for melody generation, a system could be designed to simply produce a monophonic sequence of notes [Brooks et al. 1957], or be constrained to fit a given accompaniment [Pachet and Roy 2001]. For an automatic harmonization system, the goal could involve generating three lines of music for a given melody without breaking music theoretic rules (e.g., harmonizing chorales [Ebcioğlu 1988], or producing substitute chord progressions in jazz [Chemillier 2001]. For rhythm generation, a system could focus on producing rhythmic patterns that sound like rock n' roll [Tokui and Iba 2000], or on changing the timing of onsets to make the rendering of the piece sound more human-like [Tidemann and Demiris 2008].

Timbre is unique in that it is based only on the acoustic characteristic of music. Timbre can be generated either by playing notes on a real instrument or by artificially synthesizing sounds for a note or several notes. In automatic music composition, timbre generation surfaces as a problem in orchestration, which is often modeled as a retrieval problem [Psenicka 2003], or a multi-objective search problem [Carpentier et al. 2010].

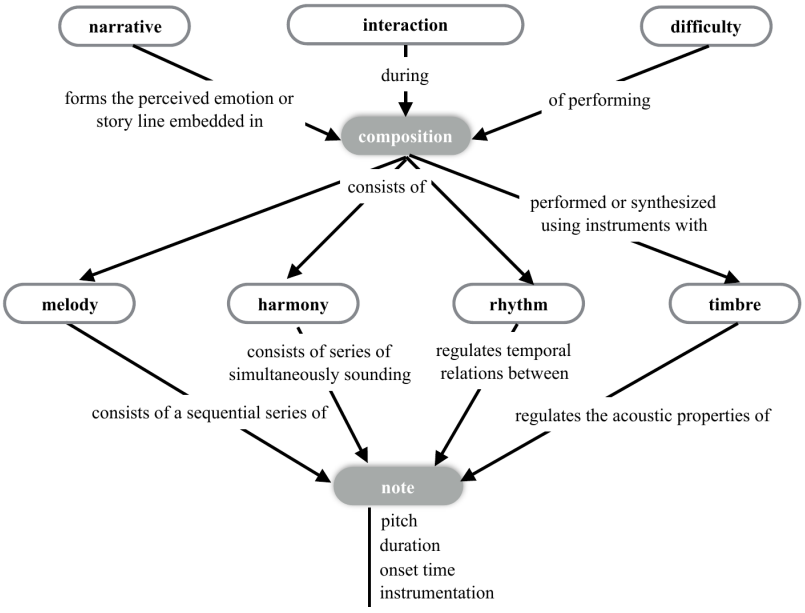

Fig. 1. Concept map for automatic music generation systems.

The objective of a system, such as matching a target timbre, will directly impact the problem definition and prospective solution techniques, such as multi-objective search or retrieval. Also notice that a music generation system can tackle more than one functional aspect—melody, harmony, rhythm, timbre—either by targeting multiple goals at the same time or focusing on one goal with other musical aspects considered constant and provided by the user.

Returning to Figure 1, three high-level concepts are shown above *composition*: *narrative*, *interactive composing*, and *difficulty*. Interactive composing refers to an online problem solving approach, which can be real-time or not, to music generation that employs user input. A system can be designed to generate each of the four essential musical elements, or a combination of them, in an interactive manner. For example, a system can listen to a person's playing and learn her or his style in real time, and improvise with the player in the same style [Pachet 2003; Assayag et al. 2006]. Another type of interactive system incorporates a user's feedback in the music generation process, using it either as critique for reinforcement learning [Franklin 2001] or as a source of parameters in music generation [François et al. 2013].

The *narrative* contributes to the emotion, tension, and/or story line perceived by the listener when listening to music [Huron 2006]. The concept of *difficulty* focuses on physical aspects of playing the instrument. Systems with ergonomic goals must consider the playability of certain note combinations on a particular instrument.

To achieve these goals, the long-term and/or hierarchical structure of the music plays an important role. These high-level goals and the long-term structure have been the focus of recent development in automatic music generation, a trend that will persist into the near future.

As shown in Figure 1, automatic music generation evokes a number of computational problems and demonstrates capabilities that span almost the entire spectrum of artificial intelligence. For example, generating music can be described as a sen-

sorless problem (generating monophonic melody without accompaniment), a partially observable problem (with accompaniment but not the underlying chord progression), or a fully observable problem (accompaniment with labeled chord progression). Different agent types, including model- and knowledge-based [Chuan and Chew 2011], goal-based [Pachet and Roy 2001], utility-based [McVicar et al. 2014], and statistical learning [Tokui and Iba 2000], have been used for music generation.

In music generation, states can be defined in terms of discrete (e.g., pitch, interval, duration, chord) as well as continuous (e.g., melodic contour, acoustic brightness and roughness) features. In addition, various techniques, such as stochastic approaches, probabilistic modeling, and combinatorial optimization, have been applied to music generation. In such a rich problem domain, it is thus especially important to understand the intricacies within each subproblem and the manner in which the subproblems are interconnected one with another.

1.2. Automating composition: early years

The idea of composers relinquishing some degree of creative control and automating certain aspects of composition has been around for a long time. A popular early example is Mozart's *Musikalisches Würfelspiel* (Musical Dice Game), whereby small fragments of music are randomly re-ordered by rolling a dice to create a musical piece. Mozart was not the only one experimenting with this idea. In fact, the first musical dice game, called Der allezeit fertige Menuetten und Polonaisencomponist (The Ever-Ready Minuet and Polonaise Composer) can be traced back to Johann Philipp Kirnberger [Kirnberger 1757]. According to Hedges [1978], at least twenty musical dice games where published between 1757 and 1812, making it possible for musical novices to compose polonaises, minuets, marches, walzes, and more.

John Cage, Charles Dodge, Iannis Xenakis and other avant-garde composers have continued the ideas of chance-inspired composition. John Cage's *Atlas Eclipticalis* was composed by randomly placing translucent paper on a star chart and tracing the stars as notes [Pritchett 1994]. In the piece called "Analogique A", Xenakis uses statistical models (Markov) to determine how musical sections are ordered [Xenakis 1992]. The composer David Cope began his "Experiments in Musical Intelligence" in 1981 as the result of a composer's block; the aim of his resultant software was to model his own composing style, so that at any given point one could request a next note, next bar, and so on. In later experiments, Cope also modeled styles of other composers [Cope 1996]. Some of the music composed using this approach proved to be fairly successful. A more extensive overview of such avant-garde composers is given by Cope [2000].

State-of-the-art music generation systems extend these ideas of mimicking styles and pieces, be it in the form of statistical properties of styles or explicitly repeated fragments. Before Section 2 describes in greater depth music generation systems in terms of their functions, the next section focuses on how the goals of music generation systems are defined and evaluated, depending on the technique used for generation.

1.3. Measuring success

For automatic music generation systems, unless the end goal is the process rather than the outcome, evaluation of the resulting composition is usually desired, and for some systems an essential step in the composition process.

The output of music generation systems can be evaluated by human listeners, using music theoretic rules, or using machine-learned models. The choice of evaluation method is primarily influenced by the goal of the music generation system, such as similarity to a corpus or a style (as encapsulated by rules or machine-learned models) versus music that sounds good. All of these goals are interrelated and impact the implementation of the music generation system and the quality of the generated pieces.

While human feedback may arguably be the most sound approach for evaluating post-hoc if the generated pieces sound good [Pearce and Wiggins 2001; Agres et al. 2017], requiring people to rate the output at each step of the process can take an excessive amount of time. This is often referred to as the human fitness bottleneck [Biles 2001]. A second issue with human evaluation is fatigue. Continuous listening and evaluating can cause significant psychological strain for the listener [Tokui and Iba 2000]. So while it can be useful, and arguably essential, to let human listeners test the final outcome of a system, human ratings aren't practically possible to guide or steer *during* the generation process.

If the goal of the automatic composition process is to create music similar to a given style or body of work by a particular composer, one could look to music theory for well-known rules such as those for music in the style of a composer, say Palestrina. These could be incorporated into an expert system or serve as a fitness function, say of a genetic algorithm. The downside to this approach is that existing rule sets are limited to a few narrowly-defined styles that have been comprehensively analyzed by music theorists or systematically outlined by the composer, which constrains its robustness and wider applicability, or are so generic as to result in music lacking definable characteristics.

The third approach, using machine-learned models, seems to offer a solution to the aforementioned problems. By learning the style of either a corpus of music or a particular piece, music can be generated with characteristics following those in the training pieces. The characteristics may include distributions of absolute or relative pitch sets, durations, intervals, and contours. A large collection of features is suggested by Towsey et al. [2001] and Conklin and Witten [1995]. Markov chains form a class of machine-learned models; they capture the statistical occurrence of features in a particular piece or corpus. Sampling from Markov models results in pieces with similar statistical distributions of the desired musical features. Other machine-learning approaches include neural networks and, more recently, deep-learning methods, which attempt to capture more complex relationships in a music piece.

A concept that directly relates to the task of evaluating the generated music, regardless of which of the above three methods are used, is *similarity*. In the first case, human listeners have formed their frame of reference through previous listening experiences [Peretz et al. 1998; Krumhansl 2001] and will judge generated pieces based on their similarity to pieces with which they are familiar. Secondly, a piece generated with music theoretic rules will possess attributes characteristic of those in the target style. Finally, pieces generated by machine-learned models will have features distributed in ways similar to the original corpus.

Since similarity is central to metrics of success in music generation systems, an important challenge then becomes one of finding the right balance between similarity and novelty or creativity. In the words of Hiller [1989]: "It is interesting to speculate how much must be changed to create a new work." For example, music based on fragments of an already existing composition, as in the case with high-order Markov models, run the risk of crossing the fine line between stylistic similarity and plagiarism [Papadopoulos et al. 2014]. Evaluating the creativity, which is sometimes equated to novelty, of the generated music is a complex topic treated in greater length in Agres et al. [2017].

In order to facilitate the comparison of results from different music generation systems, the authors have set up an online computer generated music repository[2]. This repository allows researchers to upload both audio files and sheet music generated by their systems. This will facilitate dissemination of results and promote research

[2]http://dorienherremans.com/cogemur

transparency so as to better assess the impact of different systems. Access to concrete examples through the website will allow visitors to better understand the behavior of the music generation systems that created them.

In the remainder of this paper, we will discuss each of the functional areas on which music generation systems can focus. Rather than aiming to provide an exhaustive list of music generation systems, we choose to focus on research that presented novel ideas which were later adopted and extended by other researchers. This function and design-based perspective stands in contrast to existing survey papers, which typically categorize generation systems according to the techniques that they employ, such as Markov models, genetic algorithms, rule-based systems, and neural networks—see [Papadopoulos and Wiggins 1999; Nierhaus 2009; Fernández and Vico 2013]. By offering a new taxonomy inspired by the function and design foci of the systems, we aim to provide deeper insights into the problems that existing systems tackle and the current challenges in the field, thereby inspiring future work that pushes the boundaries of the state-of-the-art.

2. A FUNCTIONAL INDEX OF MUSIC GENERATION SYSTEMS

This section explores functional aspects addressed in different music generation systems which form the taxonomy proposed in this paper; example systems are given for each aspect. The functional aspects discussed, in order of appearance, are *melody*, *harmony*, *rhythm*, *timbre*, *interaction*, *narrative*, and *difficulty*. We also touch upon *long-term structure* in relation to some of these categories.

It is worth pointing out that the aspects, while separate in their own right, can often be conflated; for example, rhythm is inherent in most melodies. Therefore, a system mentioned in the context of one aspect may also touch upon other functional aspects.

In Table I an overview is given of the different techniques used within these functional aspects. Systems are classified by their main technique and listed with their most prominent aspect. Typically, music generation systems can belong to more than one category. In this paper (and therefore also in Table I), the most important contribution of the systems is emphasized and only the systems with a clear contribution are listed. In the next subsections, the individual functional aspects will be discussed in greater detail.

Table I: Functional overview of selected music generation systems by their main technique.

Markov models	
Melody	[Pinkerton 1956; Brooks et al. 1957; Moorer 1972; Conklin and Witten 1995; Pachet and Roy 2001; Davismoon and Eccles 2010; Pearce et al. 2010; Gillick et al. 2010; McVicar et al. 2014; Papadopoulos et al. 2014]
Harmony	[Hiller Jr and Isaacson 1957; Xenakis 1992; Farbood and Schoner 2001; Allan and Williams 2005; Lee and Jang 2004; Yi and Goldsmith 2007; Simon et al. 2008; Eigenfeldt and Pasquier 2009; De Prisco et al. 2010; Chuan and Chew 2011; Bigo and Conklin 2015]
Rhythm	[Tidemann and Demiris 2008; Marchini and Purwins 2010; Hawryshkewich et al. 2011]
Interaction	[Thom 2000]
Narrative	[Prechtl et al. 2014a,b]
Difficulty	[McVicar et al. 2014]

Factor oracles	
Interaction	[Assayag et al. 2006; Weinberg and Driscoll 2006; François et al. 2007; Assayag et al. 2010; Dubnov and Assayag 2012; François et al. 2013; Nika et al. 2015]
Rhythm	[Weinberg and Driscoll 2006]
Incremental parsing	
Interaction	[Pachet 2003]
Reinforcement learning	
Interaction	[Franklin 2001]
Rule/Constraint satisfaction/Grammar-based	
Melody	[Keller and Morrison 2007; Gillick et al. 2010; Herremans and Sörensen 2012]
Harmony	[Hiller Jr and Isaacson 1957; Steedman 1984; Ebcioğlu 1988; Cope 1996; Assayag et al. 1999b; Cope 2004; Huang and Chew 2005; Anders 2007; Anders and Miranda 2009; Aguilera et al. 2010; Herremans and Sörensen 2012, 2013; Tanaka et al. 2016]
Narrative	[Rutherford and Wiggins 2002]
Difficulty	[Lin and Liu 2006]
Interaction	[Lewis 2000; Chemillier 2001; Morales-Manzanares et al. 2001; Marsden 2004]
Narrative	[Casella and Paiva 2001; Farbood et al. 2007; Brown 2012; Nakamura et al. 1994]
Neural networks/Restricted Boltzmann machines/ LSTM	
Harmony	[Lewis 1991; Hild et al. 1992; Eck and Schmidhuber 2002; Boulanger-Lewandowski et al. 2012; Herremans and Chuan 2017]
Melody	[Todd 1989; Duff 1989; Mozer 1991; Lewis 1991; Toiviainen 1995; Eck and Schmidhuber 2002; Franklin 2006; Agres et al. 2009; Boulanger-Lewandowski et al. 2012]
Interaction	[Franklin 2001]
Narrative	[Browne and Fox 2009]
Evolutionary/Population-based optimization algorithms	
Melody	[Horner and Goldberg 1991; Towsey et al. 2001; WASCHKA II 2007; Herremans and Sörensen 2012]
Harmony	[McIntyre 1994; Polito et al. 1997; Phon-Amnuaisuk and Wiggins 1999; Geis and Middendorf 2007; WASCHKA II 2007; Herremans and Sörensen 2012]
Rhythm	[Tokui and Iba 2000; Pearce and Wiggins 2001; Ariza 2002]
Interaction	[Biles 1998, 2001]
Difficulty	[Tuohy and Potter 2005; De Prisco et al. 2012]
Timbre	[Carpentier et al. 2010]
Local search-based optimization	
Melody	[Herremans and Sörensen 2012]

Harmony	[Herremans and Sörensen 2012; Herremans et al. 2015a]
Narrative	[Browne and Fox 2009; Herremans and Chew 2016a]
Timbre	[Carpentier et al. 2010]
Integer Programming	
Melody	[Cunha et al. 2016]
Other optimization methods	
Melody	[Davismoon and Eccles 2010]
Harmony	[Tsang and Aitken 1999; Farbood and Schoner 2001; Bemman and Meredith 2016]
Timbre	[Hummel 2005; Collins 2012]
Difficulty	[Radisavljevic and Driessen 2004]

2.1. Melody

Melody constitutes one of the first aspects of music subject to automatic generation. This section explores the range of automatic systems for generating melody. The generation of simple melodies is studied first, followed by the transformation of existing ones, then the more constrained problem of generating melodies that fit an accompaniment or chord sequence.

2.1.1. Melodic generation. When considering the problem of generating music, the simplest form of the exercise that comes to mind is the composition of monophonic melodies without accompaniment.

Problem description. In most melody generation systems, the objective is to compose melodies with characteristics similar to a chosen style—such as Western tonal music or free jazz—or corpus—such as music for the Ethiopian lyre the bagana [Herremans et al. 2015b], a selection of nursery rhymes [Pinkerton 1956], or hymn tunes [Brooks et al. 1957].

These systems depend on a function to evaluate the fitness of output sequences or to prune candidates. Such a fitness function, as discussed in Section 1.3 is often based on similarity to a given corpus, style, or piece. The music is often reduced to extracted features; these features can then be compared to that of the exemplar piece or corpus, a model, or existing music theoretic rules. Example features include absolute or relative pitch [Conklin 2003], intervals [Herremans et al. 2015a], durations [Conklin 2003], and contours [Alpern 1995]. Not all studies provide details of the extracted features, which makes it difficult to compare the objectives and results.

Early work. Building on the ideas of the aforementioned avant garde composers, some early work on melody generation uses stochastic models. These models capture the statistical occurrence of features in a particular song or corpus to generate music having selected feature distributions similar to the target song or corpus.

The first attempts at generating melodies with computers date back to 1956, when Pinkerton built a first order Markov model, the "Banal Tune-Maker", based on a corpus of 39 simple nursery rhymes. Using a random walk process, he was able to generate new melodies that "sound like nursery rhymes". The following year, Brooks et al. [1957] built Markov models from order one up to eight based on a dataset of 37 hymns. When using a random walk process, they noted that melodies generated by higher order

models tend to be more repetitive and those generated by lower order models had more randomness.

The trade-off between composing pieces similar to existing work and novel, creative input is a delicate one. Although Stravinsky is famously quoted as having said, "good composers borrow and great composers steal" [Raines 2015], machines still lack the ability to distinguish between artful stealing and outright plagiarism. Concepts of irony and humor can also be difficult to quantify. In order to avoid plagiarism and create new and original compositions, an automatic music generation system needs to find the balance between generating pieces similar to a given style, yet not too similar to individual pieces.

Papadopoulos et al. [2014] examined problems of plagiarism arising from higher order Markov chains. Their resulting system learns a high order model, but introduces MaxOrder, the maximum allowable subsequence order in a generated sequence, to curb excessive repeats of material from the source music piece. The sequences are generated using finite-domain constraint satisfaction. The idea of adding control constraints when generating music using Markov models was further explored by Pachet and Roy [2001]. Examples of applications of such control constraints include requirements that a sequence be globally ascending or follows an arbitrary pitch contour. Although there have been some tests of using control constraints with monophonic melodies, the research of Pachet and Roy [2001] focuses on the even more constrained problem of generating jazz solos over accompaniment, a topic that is explored in greater detail in Section 2.1.3.

Structure and patterns. Composing a monophonic melody may seem like a simple task compared to the scoring of a full symphony. Nevertheless, melodies are more than just movements between notes, they normally possess long term structure. This structure may result from the presence of motives, patterns, and variations of the patterns. Generating music from a Markov model with a random walk or Gibbs sampling typically does not enforce patterns that lead to long term structure. In recent years, some research has shown the effectiveness of using techniques such as optimization and deep learning to enforce long-term structure.

Davismoon and Eccles [2010] were some of the first researchers to frame music generation as a combinatorial optimization problem with a Markov model integrated in its objective function. In order to evaluate the music generated, their system builds a (second) Markov model based on the generated music so as to enable to system to minimize a Euclidean distance between the original model and the new model. They used simulated annealing, a metaheuristic inspired by a metallurgic technique used to cool a crystalline solid [Kirkpatrick et al. 1983], to solve this distance-minimization problem. This allowed them to pose some extra constraints to control pitch drift and solve end-point problems.

Pearce et al. [2010]'s IDyOM system uses a combination of long- and short-term Markov models. A dataset of modern Western tonal-style music was used to train a long-term model, combined with a short-term model trained incrementally on the piece being generated. The short-term model captures the changes in melodic expectation as it relates to the growing knowledge of the current fragment's structure. Local repeated structures are more likely to recur; this model will therefore recognize and stimulate repeated structures within a piece. The result is an increase in the similarity of the piece with itself, which can be considered a precursor to form.

A recent study by Roig et al. [2014] generates melodies by concatenating rhythmic and melodic patterns sampled from a database. Selection is done based on rules combined with a probabilistic method. This approach allows the system to generate melodies with larger-scale structure such as repeated patterns, which causes the piece

to have moments of self-similarity. Cunha et al. [2016] adopt a similar approach, using integer programming with structural constraints to generate guitar solos from short existing licks. The objective function consists of a combination of rules. Bemman and Meredith [2016] mathematically formalized a problem posed by composer Milton Babbitt. Babbit is famous for composing twelve-tone serial music and formulated the "all-partition array"-problem, which consists of finding a rectangular area of pitch class integers that can be partitioned into regions whereby each region represents a distinct integer partition of 12. There are only very few solutions to this computationally hard composition problem with a very structured nature, one of which was found by Tanaka et al. [2016] through constraint programming.

Herremans et al. [2015b] investigates the integration of Markov models in an optimization algorithm, exploring multiple ways in which a Markov model can be used to construct an objective function that forces the music to have the same statistical distribution of features as a corpus or piece. This optimization problem is solved using a variable neighborhood search (VNS). The main advantage of this approach is that it allows for the inclusion of any type of constraint. In their paper, the generated piece is constrained to an AABCA structure. The approach was implemented and evaluated by generating music for the bagana, an Ethiopian lyre. Since this system uses the semiotic pattern from a template piece, the newly generated pieces can be considered as having structure like the template.

The MorpheuS system [Herremans and Chew 2016a] expands on the VNS method, adding constraints on recurring (transposed) patterns and adherence to a given tension profile. Repeated patterns are detected using the compression algorithm COSIATEC [Meredith 2013]. COSIATEC finds the locations where melodic fragments are repeated in a template piece, thus supplying higher-level information about repetitions and structural organization. Tonal tension is quantified using measures [Herremans and Chew 2016b] based on the spiral array [Chew 2014].

In recent years, more complex deep learning models such as recursive neural networks have gained in popularity. The trend is due in part to the fact that such models can learn complex relationships between notes given a large-enough corpus. Some of these models also allow for the generation of music with repeating patterns and notions of structure. The next paragraphs examine research on neural network-based melody generation.

Deep learning and structure. The first computational model based on artificial neural networks (ANNs) was created by McCulloch and Pitts [1943]. Starting in the eighties, more sophisticated models have emerged that aim to more accurately capture complex properties of music. The first neural network for music generation was developed by Todd [1989], who designed a three-layered recurrent artificial neural network, whose output (one single pitch at a time) forms a melody line. Building on this approach, Duff [1989] created another ANN using relative pitches instead of absolute pitches to compose music in J.S. Bach's style. Recurrent neural networks are a family of neural networks built for representing sequences [Rumelhart et al. 1988]. They have cyclic connections between nodes that create a memory structure.

Mozer [1991] implemented a recurrent connectionist network (called CONCERT), that was used in an experiment to generate music that sounds like J.S. Bach's minuets and marches. Novel in this approach was the representation of pitches in a psychologically-grounded multidimensional space. This representation enabled the system to capture a notion of similarity between pitches. Although CONCERT is able to learn some structure, such as that of diatonic scales, its output lacks long-term coherence such as that produced by repetition and the statement of the theme at the beginning and its return near the end. While the internal memory of recursive neural

networks [Rumelhart et al. 1985] can, in principle, deal with the entire sequence history. It remains a challenge, however, to efficiently train long term dependencies [Bengio et al. 1994]. x In the same year, Lewis [1991] designed another ANN framework with a slightly different approach. Instead of training the ANN on a corpus, he mapped a collection of patterns—drawn from music ranging from random to very good—to a musicality score. To create new pieces, the mapping was inverted and the musicality score of random patterns was maximized with a gradient-descent algorithm to reshape the patterns. Due to the high computational cost, the system was only tested on simple and short compositions. Agres et al. [2009] built a recurrent neural network that learned the tonal structure of melodies, and examined the impact of the number of epochs of training on the quality of newly generated melodies. They showed that better-liked melodies were the result of models that had more sparse internal representations. Conceptually, this sort of sparse representation may reflect the way in which the human cortex encodes musical structure.

Since these initial studies, deep learning networks have increased in popularity. Franklin [2006] developed a Long Short-Term Recurrent Neural Network (LSTM) that generates solos over a reharmonization of chords. She suggests that hierarchical LSTM networks might be able to learn sub-phrase structures in future work. LSTM was developed in 1997 by [Hochreiter and Schmidhuber 1997]. It is a recurrent neural network architecture that introduces a memory structure in its nodes. More recently, Boulanger-Lewandowski et al. [2012] used a piano roll representation to create Recurrent Temporal Restrictive Boltzmann Machine (RT-RBM)-based models for polyphonic pieces. An RBM, originally called Harmonium by the original developer [Smolensky 1986], is a type of neural network that can learn a probability distribution over its inputs. While the model of Boulanger-Lewandowski et al. [2012] is intended mainly to improve the accuracy of transcription, it can equally be used for generating music. The RBM-based model learns basic harmony and melody, and local temporal coherence. Long-term structure and musical meter are not captured by this model.

The capability for RBM's to recognize long-term structures such as motives and phrases is acknowledged in a recent paper by Lattner et al. [2015], in which an RBM is used to segment musical pieces. The model reaches an accuracy rate that competes with current state-of-the-art segmentation models. Recent work by Herremans and Chuan [2017] takes a different approach inspired by linguistics. They use neural networks to evaluate the ability of semantic vector space models (word2vec) to capture musical context and semantic similarity. The results are promising and show that musical knowledge such as tonality can be modeled by solely looking at the context of a musical segment.

2.1.2. Transformation. Horner and Goldberg [1991], pioneers in applying genetic algorithms (GAs) to music composition, tackle the problem of thematic bridging, the transformation of an initial musical pattern to a final one over a specified duration. A genetic algorithms is a type of metaheuristic that became popular in the 70s through the work of [Holland 1992]. It typically maintain a set (called population) of solutions and combine solutions from this set to form new ones. In the work of Horner and Goldberg [1991], based on a set of operators, an initial melodic pattern is transformed to resemble the final pattern using a GA. The final result consists of a concatenation of all patterns encountered during this process.

Ralley [1995] uses the same technique (GA) for melodic development, a process in which key characteristics of a given melody are transformed to generate new material. The results are mixed as no interesting transformed output were found. According to Ralley [1995], the problem lies in the high subjectivity of the desired outcome.

GenDash, a compositional tool developed by composer Rodney Waschka II [WASCHKA II 2007], is not a fully automated composition system but works in tandem with a human composer. The genetic algorithm does not have any type of fitness function (human or other); it simply evolves measures of music at random. In this process, each measure is treated as a different population for evolution. Using GenDash, Waschka composed the opera *Sappho's Breath* by using a population that consists of twenty-six measures from typical Greek and Medieval songs [Dostál 2013].

Recently, Sony Computer Science Labs' Flow Composer has been used to reorchestrate Ode to Joy, the European Anthem, in seven different styles, including Bach chorales and Penny Lane by The Beatles [Pachet 2016]. The reorchestrations are based on max-entropy models, which are often used in fields such as physics and biology to model probability distributions with observed pairwise correlations [Lezon et al. 2006].

2.1.3. Chord constraints. A melody is most often paired either with counterpoint or with chords that harmonize the melody. While there exists much work on generating chords given a melody (see Section 2.2.3), some studies focus on generating a melody that fit a chord sequence.

Moorer [1972], for instance, first generates a chord sequence, then a melodic line against the sequence. The melody notes are restricted to only those in the corresponding chord at any given point in time. At each point, a decision is made, based on a second-order Markov model, to invert melodic fragments based on the chord, or to copy the previous one. The resulting short melodies have a strangely alien sound, which the author attributes to the fact that the "plan" or approach is not one that humans use, and the system does not discriminate against unfamiliar sequences.

The generation of jazz solos over an accompaniment is a popular problem [Pachet and Roy 2001; Toiviainen 1995; Keller and Morrison 2007]. The improvisation system (Impro-Visor) designed by Keller and Morrison [2007] uses probabilistic grammars to generate jazz solos. The model successfully learns the style of a composer, as reflected in an experiment described by Gillick et al. [2010], where human listeners correctly matched 95% of solos composed by Impro-Visor in the style of the famous performer Clifford Brown to the original solo. The accuracy was 90% for Miles Davis, and slightly less, 85% for Freddie Hubbard. They state that "The combination of contours and note categories seems to balance similarity and novelty sufficiently well to be characterized as jazz". The system does not capture long-term structure, which the authors suggest might be solved by using the structure of an existing solo as a template.

Eck and Schmidhuber [2002] tackle a similar problem, the generation of a blues melody following the generation of a chord sequence. They use a Long Short Term Memory RNN, which the authors claim handles long-term structure well. However, the paper does not provide examples of output for the evaluation of the long-term structure.

In the next section, we review music generation systems that focus on harmony.

2.2. Harmony

Besides melody, harmony is another popular aspect for automatic music generation. This section describes automatic systems for harmony generation, focusing on the manner in which harmonic elements such as chords and cadences are computationally modeled and produced in accordance to a specific style.

In the generation of harmonic sequences, the quality of the output depends primarily on similarity to a target style. For example, in chorale harmonization, this similarity is defined explicitly by adherence to voice-leading rules. In popular music, where chord progressions function primarily as accompaniment to a melody, the desired harmonic

progression is achieved mostly by producing patterns similar to existing examples having the same context. The context is defined by the vertical relation between melody and harmony (i.e., notes sounding at the same time) as well as horizontal patterns of chord transitions (i.e., the relationship of notes over time).

In addition to direct comparisons of harmonic similarity, the output of a chord generation system can also be evaluated under other criteria such as similarity to a genre or to the music of a particular artist.

The system must generate sequences recognizably in the target genre or belonging to a particular corpus, yet not "substantially similar" to it [Liebesman 2007] so as to avoid accusations of plagiarism. It is only a short step from similarity and plagiarism to copyright infringement. On copyright protection of ubiquitous patterns such as harmonic sequences, Gherman [2008] argues that: "When determining whether two musical works are substantially similar . . . the simple, basic harmony or variation should not be protectable as it is functional . . .. The harmony that goes beyond the triviality of primary tonal level and blocked chords is and should be protectable under copyright law."

The next sections discuss the task of counterpoint generation, followed by harmonization of chorales, general harmonization, and the generating of chord sequences.

2.2.1. Counterpoint. Counterpoint is a specific type of polyphony. It is defined by a strict set of rules that handle the intricacies that occur when writing music that has multiple independent (yet harmonizing) voices [Siddharthan 1999].

In Gradus Ad Parnassum, a pedagogical volume written in 1725, Johann Fux documented a comprehensive set of rules for composing counterpoint music [Fux and Mann 1971], which forms the basis of counterpoint texts up to the present day. Counterpoint, as defined by Fux, consists of different "species", or levels of increasing complexity, which include more rhythmic possibilities [Norden 1969].

Problem description. The process of generating counterpoint typically begins with a given melody called the *cantus firmus* ("fixed song"). The task is then to compose one or more melody lines against it. As the rules of counterpoint are strictly defined, it is relatively easy to use rules to generate or evaluate if the generated sequence sounds similar to the style of the original counterpoint music. The Palestrina-Pal system developed by Huang and Chew [2005] offers an interactive interface to visualize violations of these harmonic, rhythmic and melodic rules.

Automatic counterpoint composition systems typically handle two to four voices. The systems for generating four-part counterpoint are grouped together with four-part chorale harmonization in the next section because they follow similar rules. The systems and approaches described below handle fewer than four voices.

Approaches. Three main approaches exist for emulating counterpoint style: the first uses known rules to generate counterpoint; the second uses the rules in an evaluation function of an optimization algorithm; and, the last uses machine learning to capture the style.

In the first category, Hiller Jr and Isaacson [1957] uses rules for counterpoint to generate the first and second movements of the Illiac Suite. David Cope composes first species counterpoint given a cantus firmus in his system "Gradus." Gradus analyses a set of first species counterpoint examples and learns the best settings for 6 general counterpoint goals or rules. These goals are used to sequentially generate the piece, using a rule-based approach [Cope 2004].

Another system, developed by Aguilera et al. [2010] uses logic based on probability rules to generate counterpoint parts in C major, over a fixed cantus firmus. In the generation process, the system evaluates only the harmony characteristics of the coun-

terpoint, but not the melodic aspects. The original theory of Johann Fux contains rules that focus both melodic and harmonic interaction [Fux and Mann 1971].

The second approach, using counterpoint rules as tools for evaluation, is employed in the system called GPmuse, a GA developed by Polito et al. [1997]. GPmuse composes fifth species (mixed rhythm) counterpoint starting from a given cantus firmus. It extracts rules based on the homework problems formulated by Fux and uses the rules to define the fitness functions for the GA. The music generated by GPmuse sounds similar to the original style of counterpoint music. A problem with the system is that some "obvious" rules were not defined by Fux, such as the need for the performer (singer) to breathe. Since these rules were not explicitly programmed in GPmusic, one example output contained very long phrases which solely contained eight notes without any rests.

Strasheela is a generic constraint programming system for composing music. Anders [2007] uses the Strasheela system to compose first species counterpoint based on six rules from music theory. Other constraint programming languages, such as PWConstraints developed at IRCAM can be used to generate counterpoint, provided the user inputs the correct rules [Assayag et al. 1999b].

Herremans and Sörensen [2012] uses a more extensive set of eighteen melodic and fifteen harmonic rules based on Johann Fux's theory to generate a cantus firmus and first species counterpoint. The authors implement the rules in an objective function and optimize (increase) the adherence to these rules using a variable neighborhood search algorithm (VNS). VNS is a combinatorial optimization algorithm based on local search proposed by Mladenović and Hansen [1997]. Herremans and Sörensen [2012]'s system was also implemented as a mobile app [Herremans and Sorensen 2013], and later extended by adding additional rules based on Fux to generate fifth species counterpoint [Herremans et al. 2015a].

A final approach to the counterpoint generation problem can be seen in the application of a machine-learning method to Palestrina-style counterpoint. Farbood and Schoner [2001] implemented a Hidden Markov Model to capture the different rules of such counterpoint; they found the resulting music to be "musical and comparable to those created by a knowledgeable musician." Hidden Markov Models, first described by Baum and Petrie [1966], are used to model systems that are Markov processes with unobserved (hidden) states, and have since become known for their application in temporal pattern recognition [Yamato et al. 1992].

2.2.2. Harmonizing chorales. The harmonizing of chorales is one of the most popular music generation tasks pertaining to harmony. Chorale harmonization produces highly structured music that has been widely studied in music theory, and a rich body of theoretical knowledge offers clear directions and guidelines for composing in this style.

Problem definition. The problem of chorale harmonization has been formulated computationally in a variety of different ways. The most common form is to generate three voices designed to harmonize a given melody, usually the soprano voice [Allan and Williams 2005; Ebcioğlu 1988; Geis and Middendorf 2007; Hild et al. 1992; Phon-Amnuaisuk and Wiggins 1999; Tsang and Aitken 1999; Yi and Goldsmith 2007].

In contrast, the Bach-in-a-Box system proposed by McIntyre [1994] aims to harmonize a user-created melody, which can form one of any four possible voices. Given a monophonic sequence, the system must generate, using GA, three other notes to form a chord with each melody note while ensuring that the given melodic notes are not mutated in the process. The quality of a generated four-part sequence is then measured via fitness functions related to the construction of the chord, the pitch range and motion, the beginnings and endings of chords, smoothness of the chord progressions and chord resolution.

Some systems simplify the process by assuming that all melody notes are chord tones and chords exist at every melody note, i.e. that the polyphony is homophonic [Phon-Amnuaisuk and Wiggins 1999; McIntyre 1994; Yi and Goldsmith 2007]. In many systems, non-chord tones such as passing notes are added as an after-thought following the establishing of the chord progression; others incorporate explicit considerations of non-chord tones in the generation process [Allan and Williams 2005; Hild et al. 1992].

Solution approach. As described in [Hild et al. 1992], the results of chorale harmonization can be expressed in multiple ways, including: as a harmonic skeleton, a chord skeleton, or as four full parts complete with passing tones. A harmonic skeleton describes the chord progression as a sequence of symbols—such as roman numerals—that represent the functional role of each chord in the progression; the rhythm is implied or considered as given. A chord skeleton shows the constituent notes of each chord without passing tones. Most chorale harmonization systems aim to generate chord skeletons; few cover all three kinds of abstractions.

Some systems generate only a harmonic skeleton. For example, De Prisco et al. [2010]'s system produces functional harmonizations represented as roman numerals with indications of whether the chord is in the root position or some inversion. The system proposed by Anders and Miranda [2009] generates the harmonic backbone without requiring melodic input.

It is worth noting that the terminology for the abstractions are not used consistently in the literature. For example, Ebcioğlu [1988] describes chord skeletons as sequences of rhythmless chords with fermatas, like the harmonic skeleton in Hild et al. [1992]. The actual notes including passing tones and suspensions are generated by a fill-in view object that takes the chord skeleton as input in [Ebcioğlu 1988].

Search space and context. The complexity of the harmonization problem is defined by the size of the search space for viable chords. This size is in turn determined by the problem description, which includes the number of chord types and of chords to be generated. The size of the search space is relevant to the number of states in a hidden Markov models [Allan and Williams 2005], the number of nodes in a neural network [Hild et al. 1992], and the length of the chromosome in a genetic algorithm [McIntyre 1994].

In chorale harmonization, for a given key, the basic set of chords consists of: I, ii, iii, IV, V, V^7, vi, and vii^o and their positions (root or inversions). The size of the basic set is significantly increased if details such as secondary dominant, pivot chords, and key modulations are considered. The size of the search space can also be determined by examining composed examples.

Generating chorale harmonization can be approached as an iterative process. Given a melody, the system first generates possible configurations for the chord progression, then modifies the patterns based on certain criteria. For example, the expert system in [Ebcioğlu 1988] takes the generate-and-test approach using three types of rules implemented as first-order logic: production rules, constraints, and heuristics. Systems that use genetic algorithms also follow this iterative nature: the "chromosomes" or "population" are modified iteratively to improve the quality based on fitness functions [McIntyre 1994]. However, this iterative nature becomes computationally expensive when the chorales become longer.

To overcome this problem of search space explosion, many researchers focus on local patterns instead of the entire compositions. This is not only a practical solution for computational reasons, but also a reasonable approach because many of the voice-leading rules in chorales are concerned only with local movements in and between individual voices.

The modeling of local or short-term patterns is even more prominent in approaches that use neural networks and Markov models. In general, to determine a chord at the current time point, such systems define the local context by considering the recent chord sequences, and the melody note in the previous, current, and immediate future time points [Allan and Williams 2005]. For example, De Prisco et al. [2010] discuss three models—one considering only the current chord, one incorporating the current and the immediately preceding chord, and one considering the current and the two closest preceding chords—and their combinations to determine the current chord. Eigenfeldt and Pasquier [2009] also proposed a third-order Markov model for chord generation.

Cadences. While the cadence is a key harmonic feature used in the delineation of phrase boundaries, the modeling of cadences is not always explicitly addressed in chorale harmonization systems. Even for systems that account for phrase structure, cadences are handled to varying degrees of detail.

The generation of cadences is typically achieved through constraints. For example, ending each phrase with a cadence can be set as a hard constraint for any chord progression [Anders and Miranda 2009]. Cadences can also be induced through heuristics [Ebcioğlu 1988] or preferences, say, in cost functions [Phon-Amnuaisuk and Wiggins 1999; McIntyre 1994] that bias the system towards producing more desirable cadential patterns. For example, in [Phon-Amnuaisuk and Wiggins 1999], wrong cadences are penalized up to 100 points, 10 times more than any other rules governing voice leading, while McIntyre [1994] awarded points for proper tritone resolution, including transitions from V^7 and vii^o to I or vi.

In the work of Tsang and Aitken [1999], cadence formation is realized through four rules (out of a total of nine). Also using a rule-based approach, Geis and Middendorf [2007] included a resolution rule as a part of the harmonic score calculation. The generated cadence is then constrained through the rules to be similar to the chosen style. The modeling of cadences as constraints or preference rules can be readily incorporated in systems that use combinatorial approaches such as genetic algorithms and constraint programming.

In contrast, cadential closure is discussed almost as a by-product in systems using statistical approaches such as neural networks and Markov models. For example, in [Hild et al. 1992], harmonic closure relies on explicit coding of the beginnings and endings of phrases. Allan and Williams [2005] report that their HMM with the Viterbi algorithm generates plausible cadences similar to those in the chosen corpus; little information is provided regarding how the system ensures correct cadences, especially the ones midstream, when seeking the most likely chord progression.

Recently, Yi and Goldsmith [2007] proposed an interesting Markov model-based approach: instead of generating the most probable sequence, the authors modeled the harmonization problem as a Markov decision process so that sequences with the highest rewards, including those considering cadences, are selected. The reward is produced by a utility function, which can be either formulated based on music theory or learned from a dataset. In [Yi and Goldsmith 2007], only two rules are encoded in the utility function: chords for which melodic notes are chord tones are preferred, and authentic cadences are preferred while plagal cadences are acceptable.

2.2.3. General harmonization. The general harmonization problem can be considered as one of determining multiple synchronized-note events to fit certain user-defined criteria. Compared to chorale harmonization, the problem of generating harmonic sequences in other genres is less well defined. Unlike chorales in which fitness functions can be established based on well-studied music theoretic rules, the style, cadences, har-

monic quality, and even chord labels can often be unclear in the general harmonization problem.

A number of studies focus on generating harmonizations to user-created melodies in a popular style. Most studies adopt data-driven approaches to determine possible chords for a given melodic segment and to ensure chord-to-chord transitions are commonly observed in the examples. The systems typically produce a harmonic skeleton and use predefined patterns for creating rhythmic textures and instrumental arrangements.

Lee and Jang [2004] used the first-order Markov model with dynamic programming to determine the harmonic skeleton for a user-hummed tune; the state transition probabilities are learned from 150 songs. Simon et al. [2008] took a similar approach; training their system on 298 songs from various genres such as jazz, rock, pop, blues, and others. Both systems are evaluated via subjective feedback from listening experiments. A drawback of this approach is that the chord sequences generated tend to be generic and indistinct in style.

To preserve a recognizable style, rather than training on multi-style datasets, Chuan and Chew [2011] focused on music composed by only one artist/band, or even a single piece in the extreme case; the problem of data sparsity is overcome through a chord tone determination module that generates a set of possible chords, and the use of neo-Riemannian operations to fill in missing transitions between chords. The generated chord sequence was evaluated subjectively, and quantitatively using cross entropy.

A system for reharmonizing uplifting trance music based on a newly generated chord sequence was developed by [Bigo and Conklin 2015]. The system was extensively tested in an empirical study, in which Agres et al. [2016] found that repetitiveness in harmonic structure and tension, not solely rhythmic structure, is a contributor to listener enjoyment in this form of electronic dance music.

2.2.4. Jazz Chord Sequences. The problem of generating chord sequences unconstrained by melodic considerations is more frequently seen in jazz. Research on the generation of jazz chord progressions has focused on chord substitution and variation.

Steedman [1984] studied 12-bar blues and defined a small set of rules using a generative grammar that produces recognizable 12-bar blues chord progressions. As noted in the article, there was no explicit attempt to generate good chord progressions or to avoid bad ones. Instead, Steedman examined the harmonically meaningful chord progressions and substitutions in order to generate sequences to accompany melodies.

Chemillier [2001] provides a scenario in which a chord sequence of n bars is repeated as a loop with variations as a foundational jazz accompaniment to explain why identifying substitutions to the original sequence is crucial in jazz improvisation. Although the substitution module in Chemellier's system randomly applies Steedman's rules to the chord sequence to generate variations, the author suggests ways that the user could interact with the system to steer the selection.

In the next section, we discuss systems for rhythm generation, without regard for pitch.

2.3. Rhythm

This section discusses systems that automatically generate rhythm. While some of the above mentioned systems already include aspects of rhythm, such as duration, the focus of this section lies on research that focuses on music generation for percussion instruments.

In music generation systems, rhythm is often considered as given or embedded as an attribute of note events. Overall, there exists far fewer systems that solely gen-

erate rhythm than systems focusing on melody and harmony, but similar modeling approaches have been applied to rhythm generation.

Tokui and Iba [2000] proposed the CONGA system which combines genetic algorithms and genetic programming to produce rhythmic patterns that evolve, with user feedback as fitness function. Short fragments of rhythmic patterns form the chromosome elements in the genetic algorithm; the manner in which these patterns are concatenated into a sequence is determined by genetic programming. For evaluation, participants use the system to produce rhythmic progressions that sound like rock n' roll.

A genetic algorithm was also implemented by Ariza [2002] to generate rhythms that consists of a sequence of genetic variations. His fitness function consists of calculating the distance between a rhythm and a user-provided "fit-rhythm" through five distance measures. The results were not evaluated in the paper.

Tidemann and Demiris [2008] used hidden Markov models to learn and generate core patterns and variations similar to examples played by different drummers. Core patterns and variations are defined by the supermaximal repeats (i.e., a repeated pattern that is not part of another pattern) in the melody that correspond to structural parts such as the verse, chorus, and bridge. To produce more "human-like" drum patterns, note onset times and velocities are modeled as Gaussian distributions with noise. The generated patterns were evaluated via a classification task to determine if the generated patterns belong to the same class as the training corpus.

Hawryshkewich et al. [2011] also applied statistical approaches to generate rhythmic patterns. The Beatback system uses variable-length Markov models to store users' input via a MIDI drum interface and to generate rhythmic patterns consistent with the users' styles and pattern complexity. Each drum event is described by its duration, velocity, and instrument such as hi-hat or snare; drum patterns are generated by reproducing highly-likely sequences as observed in the user's playing.

Marchini and Purwins [2010] used variable-length Markov models to generate percussion sequences, while their system learned and reproduced sequences from audio examples. Percussion sounds are first segmented into note events using onset detection; each event is then mapped to symbolic sequences via hierarchical clustering based on acoustic similarity. Preference is given to symbolic labels that maximize temporal regularity. To generate future events that respect the metrical structure, a temporal grid is created via beat and tempo detection.

Similarity in a social context is explored by Weinberg and Driscoll [2006], who created Haile, an interactive robot that plays the drums. Haile analyses, in real time, perceptual aspects of human players, and decides on one of six interaction modes in which to generate rhythms to play with the human player based on this analysis. The six interaction modes are: imitation, stochastic transformation, perceptual transformation, beat detection, simple accompaniment, and perceptual accompaniment.

The next section moves away from typical music generation systems and deals with the more complex tasks of orchestrating music and of accounting for timbre in music generation.

2.4. Timbre

This section considers the aspect of timbre in music generation. The timbre, also referred to as tone color of sound, is the property that distinguishes different voices and musical instruments. In the context of music generation it forms an important aspect to consider when, for instance, composing music for an orchestra because the timbre of each individual voice has an effect on the perception of the composite sound.

The problem of orchestration with a target timbre is often modeled as a combinatorial problem in which the system aims to search a database of instrument sound

samples to retrieve some combination of a subset of the sounds that produce a similar perceived timbre.

In early systems, timbral closeness was only measured using similarity in the frequency spectrum. For example, Psenicka [2003] proposed the SPORCH system, which provides orchestration for any acoustic instrument ensemble to match as best possible any arbitrary sound file. The database consists of descriptive instrument features such as pitch range, the loudest/softest dynamic levels, and notation information (e.g. clef, transposition, etc.) about sound samples. Orchestration is then determined through iterative search for the best instrument sample mix with frequency spectral peaks most similar to those of the target file.

Similarly, in Hummel [2005], the system searches iteratively for virtual music instruments that when synthesized together create speech-like timbres. The algorithm minimizes the difference in spectral envelope between the current sound and the target timbre by iteratively adding sounds that minimize the residual error.

McCormack [1996] developed an L-grammar that generates polyphonic music. Their model learns features based on pitch, duration and timbre. The details of the timbral characteristics were not disclosed in the paper.

Carpentier et al. [2010] points out that acoustic instrument orchestration is significantly different from, and more complex than, sound synthesis. They model orchestration as a constrained multi-objective search problem wherein the system aims to find combinations of sounds similar to a target timbre. Sound samples stored in a database are represented by their sound attributes and features. Sound attributes are symbolic labels related to discrete variables in compositions, including pitch, dynamics, and playing style; sound features represent psychoacoustic characteristics such as brightness and roughness that can be used to quantify perceptual dissimilarity. To minimize the perceived dissimilarity, the authors used randomly weighted Chebychev aggregation functions to model dissimilarity as a set of mono-objective problems. Finally, a genetic algorithm is employed to find the optimal combination given constraints on sound attributes as well as perceptual dissimilarity.

More recently, Collins [2012] applied machine learning to automatic composition of electroacoustic music, taking into account the quality of the final mix. A piece is composed by combining and modifying existing audio segments. The system first analyses the features of the audio segments, such as percussive onset patterns and absolute peak amplitude, to produce suitable intermediate material for mixing. The segments are then modified by applying effects including delays, filters, and time stretching. The composition is iteratively refined using a density envelope on structural parameters such as perceived loudness, sensory dissonance, and increased tension to control the level of activities. The best mix is selected as the one most similar to an exemplar piece using dynamic time warping. A similar approach is employed by Sturm [2006], who uses adaptive concatenative sound synthesis to generate and transform digital sound. Short segments are used to synthesize variations of sound much like a collage, based on a measure of similarity, the L_1-norm of the difference, of audio features.

In the next three sections, aspects newly associated with music generation are discussed, such as interaction, narrative and difficulty. The section to follow most immediately will explore interactive improvisation systems which are capable of performing together with a human player. Style replication in real-time jazz and other improvisation systems is also addressed in the upcoming section.

2.5. Interaction

This section considers systems in which two-way communication between the computer and human player(s) exist; both the player and the system listen to what is being played, anticipate, and improvise new music in real time. Previous examples

addressed music generation in the absence of live interaction with a player, based on the generated music's similarity to either a target style or piece. Here, we focus on similarity in a social context and turn to interactive systems, in which the generation algorithm "improvises" in real-time with a player. In this scenario, similarity to the style of the player and self-similarity to what is previously played within the piece become an important goal, thus shifting the focus to the requirements of interaction.

While there are computer-assisted composition systems that allow the user/composer to interact with the system and iteratively improve generated solutions in a non-performance setting (e.g. [Farbood et al. 2007], most of these systems have been discussed in their respective sections above. In this section the focus lies on real-time *performance* systems.

Early Work. One of the earliest automatic improvisation systems was created by George Lewis in the 1980s. One of his compositions, called Voyager, is composed in automatic response to a musician playing, as well as to the program's "own internal processes". In this early work, the performer is not able to control the system during performance [Lewis 2000].

Structured improvisation. One of the first interactive jazz solo generators, GenJam [Biles 1998], generates melody lines over a given chord progression. It listens to a human player's last four bars, maps it to a chromosome representation and evolves what it "hears" with a GA into what it will play in real time. Fitness evaluation is performed by a human listener who continually gives feedback, rating the output as "good" or "bad".

Thom [2000] created Band-out-of-the-Box (BoB), an agent built for interactive jazz/blues improvisation of four-bar solos, with the goal of developing a system that is realistic and fun to play with. A probabilistic approach is used, based on variable tree encoding with multiple features—pitch class, interval, and melodic direction. The model is trained on warm-up sequences prior to the performance; the features extracted in the warm-up are first clustered based on histograms; the resulting statistics are then used during real-time generation to determine the current musical environment.

Interactive jazz generation is explored further in research by Franklin [2001], who uses a set of rudimentary rules for jazz and a neural network in combination with reinforcement learning to trade fours between a musician and the system. The system, called CHIME, has a stochastic element that allows for out-of-chord changes, which the author suggests can be done more pointedly and purposefully in future research. The author also points out that the hard coded rules do not encompass the developing of a statement or the creation of a shape.

Free improvisation. A second type of improvisation system generates music more freely with a performer, in real-time, without a fixed, predefined structure. Pachet [2003]'s Continuator uses a Lempel-Ziv parsing algorithm [Assayag et al. 1999a]—adapted to properly handle rhythm, beat, harmony and imprecision—to learn the characteristics of any style. It is able to concurrently learn and generate a stream of music that is similar to a style such as jazz or a player's own style; it can generate music, either as a standalone system, as continuations of a performer's input, or as an interactive improvisation backup. By aggregating clusters of notes and treating them as units, the Continuator is able to handle polyphonic music.

Using a different data structure, the factor oracle, improvisation systems belonging to the OMax family Assayag et al. [2006] can also concurrently encode and generate music in a player's style, and handle polyphonic music. The factor oracle is a finite state automaton, originally designed to efficiently search for substrings (factors) in a

text [Allauzen et al. 1999]. More recent extensions further allow the OMax system to handle audio signals instead of symbolic MIDI; the resulting audio-based system is called Ofon. OMax's approach has also been applied to speech to simulate rap, and to video frames [Bloch et al. 2008].

Assayag et al. [2010] and Dubnov and Assayag [2012] further experimented with audio-based factor oracles to improvise music resulting in systems such as the OMax-Ofon system developed by Assayag et al. [2010]. The system developed by Dubnov and Assayag [2012] produces variations from an audio recording using a graph of repeated factors found in the recording; the system's main challenge consisting of the marking and allocating of regions in the original audio that are deemed most promising for the oracle to focus on in order to achieve the desired result. To do this, Dubnov and Assayag introduce an analysis method based on Information Rate (a concept previously linked to musical anticipation [Dubnov 2006]); in contrast to the previous factor oracle systems, the audio analysis is done in advance, and a performance is in a sense pre-planned.

The earlier OMax systems are agnostic to rhythmic and longer-range structures. Nika et al. [2015]'s system ImproteK, also based on the factor oracle, generates both rhythms and harmonies. It was recently built into an architecture that aims at combining reactivity and anticipation in the music generation processes steered by a "scenario", which can be a four-bar jazz chord sequence. The system composes off-line given a scenario; during the performance, this can be re-written to fit the performance.

Performer feedback. Mimi [François et al. 2007], like the OMax family of interactive improvisation systems, is based on the factor oracle. A novelty of this system is that it allows the user to visually see the recent past and future generated music, so as to be aware of the musical context and to plan a response. The performer is also the operator of Mimi, controlling her learning rate, switching on and off the learning, and clearing her memory [Schankler et al. 2014]. A variation on Mimi, Mimi4x [François et al. 2013], allows the user to control four interacting Mimi instances and to structure an improvisation by deciding when and which of the four Mimi-generated streams start and stop, their re-combination rates, and the playback dynamic levels.

Other interactions. Besides the above-mentioned systems, in which music is generated in partnership with a human musician, some systems use other types of interactions. A system created by Marsden [2004] generates melodies based on the movements of a dancer, combined with elaborations based on Schenkerian analysis. The system is given a "background", which consists of one note per bar, key and meter information. Depending on the speed of the dancer's movements, more target notes are generated. Gait is another determining influence, for example, a crouching gait is associated with high regularity. The output of the system follows harmonic and intervallic patterns found in real music, yet lacks subdivision into meaningful phrases. The association, reflecting similarity, between the movements of a dancer and changes in melody is credible, although it does not convey the feeling that the dancer is controlling the music, mostly because of a slight lag in the system. The interactive music improvisation system (SICIB) developed by Morales-Manzanares et al. [2001] has a similar setup, as it detect motion from sensors attached to dancers, and uses a rule-based approach to translate this to music. Motion characteristics such as curvature and torsion of movements, and velocity and acceleration are taken into account. The generation is performed by the Escamol system which uses grammar-rules [Morales-Manzanares 1992] and real-time synthesis by Aura [Dannenberg and Brandt 1996]

Another interactive system, the robot drummer called Haile, developed by Weinberg and Driscoll [2006] is discussed in more detail in Section 2.3, which calls out the rhyth-

mic aspects of music generation systems. The next section will explore music with a narrative, which includes game music and video background music.

emotion

2.6. Narrative

Narrative music is music that tells a story. The narration consists of a set of representational, organizational, and discursive cues that deliver story information to the audience. In this section we focus on different types of narratives, such as tension profiles, the blending of fragments for game music, leitmotifs, and film music.

Narrative cues create structure within music, including variation in emotions evoked throughout a piece (synchronized with video or game play), tension profiles, leitmotifs, repeated patterns and others. The enforcing of narrative structure can lead to similarity within a piece (e.g. repeated patterns and motifs) or, on a higher-level, between emotions evoked by the music and simultaneous media such as video or games.

In recent years, there has been increased interest in creating background music for games and video. The goal in these types of composition problems is that the music should match the content or emotional content of a scene/narrative. The idea of program music, music having an extra-musical narrative or purpose, is an old one. For example, "the adventures of Don Quixote" composed by Richard Strauss and Hector Berlioz's "Symphonie fantastique" both derive inspiration from extra-musical sources. Inherent in music with a narrative is the existence of long term structure, which was already touched upon in Section 2.1; the discussion continues in the following paragraphs.

Tension. An important tool for evoking emotion is the use of musical tension. Audio features, such as roughness, have been shown to correlate with perceived tension/relaxation patterns in music [Vassilakis 2005]. Farbood [2012] conducted an extensive experiment to build a perceptual tension model that takes into account the dynamic, temporal aspects of listening. Farbood models tension in terms of multiple musical parameters, inclusive of both audio and score-based features. Farbood et al. [2007] also created Hyperscore, a graphical, computer-assisted composition system in which users can intuitively edit and visualize musical structures. Both low-level and high-level musical features (such as tone color, melodic shape, dynamics, harmonic tension) are mapped to graphical elements that users can control and which allows them to create compositions. This allows users to, for instance, draw a tension line for the new composition.

Browne and Fox [2009] used simulated annealing to arrange pre-written motifs according to a pre-specified musical tension profiles. The tension profiles were computed using an ANN model, and Kullback–Leibler divergence was employed to measure the distance between the desired and the observed tension profiles.

More recently, Herremans and Chew [2016a] used a tonal tension model based on the spiral array [Herremans and Chew 2016b] to calculate tension of a polyphonic piece. The algorithm, called MorpheuS, constrains the detected patterns and generates music that fits as best possible a given tension profile. This tension profile can be provided by the user or calculated based on a template piece. The generation process is guided by a variable neighborhood search algorithm.

The breaking of rules that govern Western tonal music elicits tension. In Rutherford and Wiggins [2002]'s scary music study, more scary music is generated by breaking the Western tonal music rules. The results were verified by human listeners who noted the scariness dimensions of the generated music.

Blending. Game music is most frequently generated by cross-fading between audio files each time the player shifts from one game state to another [Collins 2008]. An

exception is the music for Depression Quest[3] which generates music dynamically as one moves through the different scenarios in the game. Brian Eno, a composer known for creating generative systems for ambient music, collaborated with Maxis/EA games to create a soundtrack generation system for the game "Spore", in which the music changes based on a gamer's style of play [Johnson 2006]. Details of how these systems work are not publicly available. In traditional games that use cross-fading, however, it is not uncommon for the two fragments to clash rhythmically or harmonically. The clash can be ameliorated by techniques such as crossfading quickly, which can be distracting or jarring.

Müller and Driedger [2012] devised an automatic DJ system for crossfading that ensures smooth blending, yet still requires the audio fragments to be harmonically and rhythmically similar. Smooth blending can be improved by restricting the range of allowed rhythms and harmonies; however, this would also restrict the musical variations and expressive capacity of the music. In order to solve this problem, Prechtl et al. [2014a] created a real-time system that generates the music from scratch instead of using existing fragments. The music generation process uses a stochastic model and takes into account emotion parameters such as alarm or danger.

Leitmotifs. The system developed by Brown [2012] focuses on "Leitmotifs", short and distinctive musical fragments associated with game characters and elements, a strategy commonly employed in Western opera. Each of these motifs are embedded in different musical forms; each musical form is associated with different degrees of harmonic tension and formal regularity, thus conveying different amounts of "markedness". In combination, the leitmotifs and forms correspond to different states of the story of a game. See Collins [2009] for a more complete overview of procedural music in games.

Film music. Music with a narrative is frequently used as background music to films. The effect that music has on perceived emotion in film has been studied by Parke et al. [2007]. When mapping perceived emotion to a three-dimensional space of stress, activity, and dominance, the geometrical center of mass of the three perceived emotions (in this space) when experiencing film and music combined is found to be in between that of the participants who listened to music alone and watched film alone. In the study, the film clips were selected for their ambiguous meanings. Prechtl et al. [2014b] argue for the need for thorough empirical evaluation when generating music purported to communicate particular emotions.

Nakamura et al. [1994] created a prototype system that automatically generates sound effects and background music for short video animations. Music—harmony, melody and rhythm—is generated for each scene, taking into consideration the mood, the intensity of the mood, and the musical key used in the preceding scene. The characteristics and intensity of the movements on screen determine the sound effects. Another example application for video background music is MAgentA, created by Casella and Paiva [2001], the goal for which is to generate "film-like" music using the mood of the environment within which it is embedded.

The final functional aspect in music generation systems, that of the instrument playing difficulty, is discussed in the next section.

2.7. Difficulty

The difficulty of a piece of music refers to the level of skill required for a musician to play the piece. When automatically composing musical pieces, the manipulation of features such as melody, harmony, rhythm, and timbre often rise to the fore, and

[3]https://isaacschankler.bandcamp.com/album/depression-quest-ost

ergonomic goals such as ease of playing are often ignored. One could argue that if a model is trained on existing pieces using the appropriate feature set, a new piece that is sampled from this model should be equally playable. It would be interesting to explicitly measure difficulty to verify this causality. Thus, generating a piece of similar playability to pieces in a corpus, or of a predefined difficulty level for an instrument could be the main goal of a music generation system.

Music generation according to difficulty level. Tuohy and Potter [2005] developed a genetic algorithm that generates playable guitar music by minimizing hand and finger movements. More recently, McVicar et al. [2014] automatically generated lead and rhythm guitar music in tablature notation, based on a given chord and key sequence.

Sébastien et al. [2012] implemented a system that measures the difficulty of a piano piece based on seven different characteristics including harmony, fingering, polyphony, and irregularity of rhythm. Such a system could easily be improved with systems for automatically computing piano fingerings [Lin and Liu 2006; De Prisco et al. 2012; Balliauw et al. 2017] and string instrument fingerings [Sayegh 1989; Radisavljevic and Driessen 2004; Radicioni et al. 2004]. The combination of fingering and difficulty evaluation systems with music generation systems provides an opportunity to evaluate pieces in a non-traditional, yet essential way.

3. FUTURE CHALLENGES

Over the last few decades, research in music generation has achieved tremendous progress in generating well-defined aspects of music such as melody, harmony, rhythm, and timbre. State-of-the-art statistical models, advanced optimization techniques, larger digital databases on which to train models, and increase in computing power have all led to the field producing better systems. Why then are we not using music generation systems in our day to day lives? The above survey shows that an important overarching challenge remains: that of creating music with long-term structure.

Long-term structure, which often takes the form of recurring themes, motivs, and patterns, is an essential part of any music listening experience [Lerdahl and Jackendoff 1983]. Recent music generation systems have tackled this challenge by constraining certain types of long-term structure, such as recurrent patterns [Herremans and Chew 2016a], form [Tanaka et al. 2016; Herremans et al. 2015b], cadence [Cunha et al. 2016], and pitch contour [Pachet and Roy 2001]. Secondly, developments in the field of deep learning [Eck and Schmidhuber 2002; Boulanger-Lewandowski et al. 2012] show that neural networks can incorporate memory structures when learning sequential data. The ability of techniques such as RNN and LSTM to capture long-term structure should be further investigated.

In order to make computer generated music systems part of our daily lives, there is a crucial need for more "intelligent" systems in which newly composed music matches higher-level concepts. This intelligence can be expressed in the functional domain of the "narrative". While there are recent attempts at generating music with tension [Farbood et al. 2007; Herremans and Chew 2016a], that matches a computer game [Prechtl et al. 2014a], that embody leitmotivs [Brown 2012] or that accompany film [Nakamura et al. 1994], and others, there is still ample room for better understanding the connection between music and emotion, so as to integrate this crucial relationship in music generation systems. This could lead to real-life practical applications such as real-time music generation for games, and background music for film and video.

While machine learning techniques can be extremely useful in tasks such as the above-mentioned modeling of emotion in music, they usually require large amounts of data. Therefore, the field has seen an ongoing need for more data. There lies a real potential for future work to move towards intelligent systems that do not require

copious amounts of data, that are capable of innate reasoning, and thus better mirror the workings of the human mind. This would also solve the continuous challenge of finding a balance between regenerating existing music and novel fragments without plagiarizing, as touched upon in Section 1.3.

One of the characteristics of computer generated music that is often neglected is playing difficulty. While one application would be to tailor novel music to a certain level of musician skill, there is also potential for using detected/calculated playing difficulty as an evaluation measure for generated music.

A challenge related to making music generation systems usable by the general public is, not only the quality of the generated musical content, but also the quality of the rendering. While this is not an aspect that we explicitly surveyed in this paper, it nevertheless is important in creating a real-life applications. In recent years, the field of automatic music production has gained increasing traction. Research topics in this field include human-like rendering of midi files with expressive timing [Bresin and Friberg 2000; Grachten et al. 2014] and automatic mixing [Deruty 2016]. Furthering the development of systems for realistic rendering of generated music, which is often in MIDI, will stimulate the attractiveness and usability of music that is generated automatically.

The success of music generation systems is not only measured through the practical adoption of the systems. Over the course of the years, researchers have adopted multiple methods for evaluating the output of systems, as outlined in Section 1.3. It remains difficult, however, to objectively compare different systems as they usually take different input parameters, generate different aspects of music, are trained on different styles, or do not have audio examples available for the reader. Furthermore, in listening experiments, the Mere-exposure effect [Zajonc 1968] will make listeners prefer existing pieces over new ones, as familiarity causes a higher enjoyment. To address this need for the proper comparison of systems to assess the state-of-the art, the authors of this paper have set up a publicly accessible repository of computer generated music systems[4]. Apart from the goal of stimulating the visibility of music generation systems and their outputs, this online repository will facilitate the comparison of systems by collecting detailed information such as the nature of the system's input/output, and potential manual corrections performed.

4. CONCLUSIONS

This article has presented a taxonomy for the key concepts that form the functional goals of music generation systems. We then provided a survey of the state-of-the-art in music generation systems with respect to this functional taxonomy. By focusing on what current systems can and cannot do, rather than the algorithmic techniques, we obtain a clearer view of the frontiers of automatic music generation, thus setting the stage for new breakthroughs. This approach has allowed us to identify uncharted areas and challenges for the field of automatic music composition.

In line with the current trend of companies such as Google (through the Magenta project) and Jukedeck, music generation systems will become ever more prominent in our day to day lives. The functional overview of systems described in this paper shows the areas with opportunities for further advancement to make automatic music generation a viable tool for applications ranging from artistic innovation to the creation of adaptive, copyright-free music for games and videos. Current challenges of the field include generating music with long-term structure; capturing higher-level content such as emotion and tension; creating models that possesses innate reasoning so as to reduce the amount of training data needed; and the promotion of transparent and

[4]http://dorienherremans.com/cogemur

objective evaluation methods. In order to facilitate the latter and stimulate visibility and evaluation of current music generation systems, the authors have set up an online repository for computer generated music results[4].

References

K. Agres, J.E. DeLong, and M. Spivey. 2009. The sparsity of simple recurrent networks in musical structure learning. In *Proceedings of the 31th Annual Conference of the Cognitive Science Society*. 3099–3104.

K. Agres, J. Forth, and G.A. Wiggins. 2017. Evaluation of Musical Creativity and Musical Metacreation Systems. *Computers and Entertainment* 14, 3, Article 3 (Jan. 2017), 33 pages.

K. Agres, D. Herremans, L. Bigo, and D. Conklin. 2016. Harmonic Structure Predicts the Enjoyment of Uplifting Trance Music. *Frontiers in Psychology* 7 (2016), 1999.

G. Aguilera, J. Luis Galán, R. Madrid, A.M. Martínez, Y. Padilla, and P. Rodríguez. 2010. Automated generation of contrapuntal musical compositions using probabilistic logic in Derive. *Mathematics and Computers in Simulation* 80, 6 (2010), 1200–1211.

M. Allan and C.K.I. Williams. 2005. Harmonising chorales by probabilistic inference. *Advances in neural information processing systems* 17 (2005), 25–32.

C. Allauzen, M. Crochemore, and M. Raffinot. 1999. Factor oracle: A new structure for pattern matching. In *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer, 295–310.

A. Alpern. 1995. Techniques for algorithmic composition of music. *On the web: http://hamp. hampshire. edu/˜adaF92/algocomp/algocomp* 95 (1995).

T. Anders. 2007. *Composing music by composing rules: Design and usage of a generic music constraint system*. Ph.D. Dissertation. Queen's University Belfast.

T. Anders and E.R. Miranda. 2009. A computational model that generalises Schoenberg's guidelines for favourable chord progressions. In *Proceedings of the Sound and Music Computing Conference*. 48–52.

C. Ariza. 2002. Prokaryotic Groove: Rhythmic Cycles as Real-Value Encoded Genetic Algorithms.. In *Proceedings of the International Computer Music Conference (ICMC)*. 561–568.

G. Assayag, G. Bloch, and M. Chemillier. 2006. OMax-ofon. *Sound and Music Computing (SMC)* 2006 (2006).

G. Assayag, G. Bloch, A. Cont, and S. Dubnov. 2010. Interaction with machine improvisation. In *The Structure of Style*. Springer, 219–245.

G. Assayag, S. Dubnov, and O. Delerue. 1999a. Guessing the Composer's Mind. In *Proceedings of the International Computer Music Conference (ICMC*. Bejing, China, 1–1.

G. Assayag, C. Rueda, M. Laurson, C. Agon, and O. Delerue. 1999b. Computer-assisted composition at IRCAM: from PatchWork to OpenMusic. *Computer Music Journal* 23, 3 (1999), 59–72.

M. Balliauw, D. Herremans, D. Palhazi Cuervo, and K. Sörensen. 2017. A variable neighbourhood search algorithm to generate piano fingerings for polyphonic sheet music. *International Transactions Of Operations Research, Special Issue on Variable Neighbourhood Search* 24 (2017), 509–535. Issue 3.

L.E. Baum and T. Petrie. 1966. Statistical inference for probabilistic functions of finite state Markov chains. *The annals of mathematical statistics* 37, 6 (1966), 1554–1563.

B. Bemman and D. Meredith. 2016. Generating Milton Babbitt's all-partition arrays. *Journal of New Music Research* 45, 2 (2016), 184–204.

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with

gradient descent is difficult. *Neural Networks, IEEE Transactions on* 5, 2 (1994), 157–166.
L. Bigo and D. Conklin. 2015. A viewpoint approach to symbolic music transformation. In *International Symposium on Computer Music Multidisciplinary Research*. Springer, 213–227.
J.A. Biles. 1998. Interactive GenJam: Integrating real-time performance with a genetic algorithm. In *Proceedings of the 1998 international computer music conference*. 232–235.
J.A. Biles. 2001. Autonomous GenJam: eliminating the fitness bottleneck by eliminating fitness. In *Proceedings of the GECCO-2001 Workshop on Non-routine Design with Evolutionary Systems*. Morgan Kaufmann, San Francisco, California, USA.
G. Bloch, S. Dubnov, and G. Assayag. 2008. Introducing video features and spectral descriptors in the OMax improvisation system. In *International Computer Music Conference*, Vol. 8.
N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. 2012. Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, John Langford and Joelle Pineau (Eds.). ACM, New York, NY, USA, 1159–1166.
R. Bresin and A. Friberg. 2000. Emotional coloring of computer-controlled music performances. *Computer Music Journal* 24, 4 (2000), 44–63.
F.P. Brooks, A.L. Hopkins, P.G. Neumann, and W.V. Wright. 1957. An experiment in musical composition. *Electronic Computers, IRE Transactions on* 3 (1957), 175–182.
D. Brown. 2012. Mezzo: An adaptive, real-time composition program for game soundtracks. In *Proceedings of the AIIDE Workshop on Musical Metacreativity*.
T. M. Browne and C. Fox. 2009. Global Expectation-Violation as fitness function in evolutionary composition. In *Applications of Evolutionary Computing*. Springer, 538–546.
G. Carpentier, G. Assayag, and E. Saint-James. 2010. Solving the musical orchestration problem using multiobjective constrained optimization with a genetic local search approach. *Journal of Heuristics* 16, 5 (2010), 1–34.
P. Casella and A. Paiva. 2001. Magenta: An architecture for real time automatic composition of background music. In *Intelligent Virtual Agents*. Springer, 224–232.
M. Chemillier. 2001. Improvising jazz chord sequences by means of formal grammars. In *Journées d'informatique musicale*. 121–126.
E. Chew. 2014. *Mathematical and Computational Modeling of Tonality: Theory and Applications*. Vol. 204. Springer, New York.
C.-H. Chuan and E. Chew. 2011. Generating and evaluating musical harmonizations that emulate style. *Computer Music Journal* 35, 4 (2011), 64–82.
K. Collins. 2008. *Game sound: an introduction to the history, theory, and practice of video game music and sound design*. Mit Press.
K. Collins. 2009. An introduction to procedural music in video games. *Contemporary Music Review* 28, 1 (2009), 5–15.
N. Collins. 2012. Automatic composition of electroacoustic art music utilizing machine listening. *Computer Music Journal* 36, 3 (2012), 8–23.
D. Conklin. 2003. Music generation from statistical models. In *Proceedings of the AISB Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*. Aberystwyth, Wales, 30–35.
D. Conklin and I. Witten. 1995. Multiple viewpoint systems for music prediction. *Journal of New Music Research* 24, 1 (1995), 51–73.
D. Cope. 1996. *Experiments in musical intelligence*. Vol. 12. AR editions Madison, WI.
D. Cope. 2000. *New directions in music*. Waveland Press.

D. Cope. 2004. A musical learning algorithm. *Computer Music Journal* 28, 3 (2004), 12–27.

N. Cunha, A. Subramanian, and D. Herremans. 2016. Uma abordagem baseada em programação linear inteira para a geração de solos de guitarra. *XLVIII Simpósio Brasileiro de Pesquisa Operacional (SBPO)* (09/2016 2016).

R.B. Dannenberg and E. Brandt. 1996. A flexible real-time software synthesis system. In *Proceedings of the International Computer Music Conference, International Computer Music Association*. 270–273.

S. Davismoon and J. Eccles. 2010. Combining musical constraints with Markov transition probabilities to improve the generation of creative musical structures. In *Applications of Evolutionary Computation*. Springer, 361–370.

R. De Prisco, A. Eletto, A. Torre, and R. Zaccagnino. 2010. A Neural Network for Bass Functional Harmonization. *Applications of Evolutionary Computation* 6025 (2010), 351–360.

R. De Prisco, G. Zaccagnino, and R. Zaccagnino. 2012. A differential evolution algorithm assisted by ANFIS for music fingering. In *Swarm and Evolutionary Computation*. Springer, 48–56.

E. Deruty. 2016. Goal-Oriented Mixing. In *Proceedings of the 2nd AES Workshop on Intelligent Music Production*, Vol. 13. London.

M. Dostál. 2013. Evolutionary music composition. In *Handbook of Optimization*. Springer, 935–964.

S. Dubnov. 2006. Spectral anticipations. *Computer Music Journal* 30, 2 (2006), 63–83.

S. Dubnov and G. Assayag. 2012. Music design with audio oracle using information rate. In *Musical Metacreation: Papers from the 2012 AIIDE Workshop*.

M.O. Duff. 1989. Backpropagation and Bach's 5th cello suite (Sarabande). In *Proceedings of the International Joint Conference on Neural Networks*. 575.

K. Ebcioğlu. 1988. An expert system for harmonizing four-part chorales. *Computer Music Journal* 12, 3 (1988), 43–51.

D. Eck and J. Schmidhuber. 2002. A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale* (2002).

A. Eigenfeldt and P. Pasquier. 2009. A realtime generative music system using autonomous melody, harmony, and rhythm agents. In *12th Generative Art Conference GA*.

M.M. Farbood. 2012. A parametric, temporal model of musical tension. *Music Perception* 29, 4 (2012), 387–428.

M. Farbood, H. Kaufman, and K. Jennings. 2007. Composing with hyperscore: An intuitive interface for visualizing musical structure. In *Proceedings of the International Computer Music Conference (ICMC)*, Vol. 59.

M. Farbood and B. Schoner. 2001. Analysis and synthesis of Palestrina-style counterpoint using Markov chains. In *Proceedings of the International Computer Music Conference*. 471–474.

J.D. Fernández and F. Vico. 2013. AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research* (2013), 513–582.

A.R.J. François, E. Chew, and D. Thurmond. 2007. Visual feedback in performer-machine interaction for musical improvisation. In *Proceedings of the 7^{th} international conference on New interfaces for musical expression*. ACM, 277–280.

A.R.J. François, I. Schankler, and E. Chew. 2013. Mimi4x: An interactive audio-visual installation for high-level structural improvisation. *International Journal of Arts and Technology* 6, 2 (2013), 138–151.

J.A. Franklin. 2001. Multi-phase learning for jazz improvisation and interaction. In *Proceedings of the Eighth Biennial Symposium for Arts & Technology*.

J.A. Franklin. 2006. Recurrent neural networks for music computation. *INFORMS Journal on Computing* 18, 3 (2006), 321–338.
J.J. Fux and A. Mann. 1971. *The study of counterpoint from Johann Joseph Fux's Gradus Ad Parnassum - 1725*. Norton, New York.
M. Geis and M. Middendorf. 2007. An ant colony optimizer for melody creation with baroque harmony. In *IEEE Congress on Evolutionary Computation*. 461–468.
S. Gherman. 2008. Harmony and its Functionality: A Gloss on the Substantial Similarity Test in Music Copyrights. *Fordham Intell. Prop. Media & Ent. LJ* 19 (2008), 483.
J. Gillick, K. Tang, and R.M. Keller. 2010. Machine learning of jazz grammars. *Computer Music Journal* 34, 3 (2010), 56–66.
M. Grachten, C.E. Cancino Chacón, and G. Widmer. 2014. Analysis and prediction of expressive dynamics using Bayesian linear models. In *Proceedings of the 1st international workshop on computer and robotic Systems for Automatic Music Performance*. 545–552.
A. Hawryshkewich, P. Pasquier, and A. Eigenfeldt. 2011. Beatback: A real-time interactive percussion system for rhythmic practise and exploration. In *Proceedings of the NIME Conference*. 100–105.
S.A. Hedges. 1978. Dice music in the eighteenth century. *Music & Letters* 59, 2 (1978), 180–187.
D. Herremans and E. Chew. 2016a. MorpheuS: Automatic music generation with recurrent pattern constraints and tension profiles. *IEEE TENCON* (November 2016).
D. Herremans and E. Chew. 2016b. Tension ribbons: Quantifying and visualising tonal tension. In *Second International Conference on Technologies for Music Notation and Representation (TENOR)*. Cambridge, UK.
D. Herremans and C.-H. Chuan. 2017. Modeling Musical Context with Word2vec. *First International Workshop On Deep Learning and Music* 1 (05/2017 2017), 11–18.
D. Herremans and K. Sörensen. 2012. Composing first species counterpoint with a variable neighbourhood search algorithm. *Journal of Mathematics and the Arts* 6, 4 (2012), 169–189.
D Herremans and K Sörensen. 2013. Composing Fifth Species Counterpoint Music With A Variable Neighborhood Search Algorithm. *Expert Systems with Applications* 40, 16 (2013), 6427–6437.
D. Herremans and K. Sorensen. 2013. FuX, an android app that generates counterpoint. In *Computational Intelligence for Creativity and Affective Computing (CICAC), 2013 IEEE Symposium on*. 48–55.
D. Herremans, K. Sörensen, and D. Martens. 2015a. Classification and generation of composer-specific music using global feature models and variable neighborhood search. *Computer Music Journal* 39 (2015), 91.
D. Herremans, S. Weisser, K. Sörensen, and D. Conklin. 2015b. Generating structured music for bagana using quality metrics based on Markov models. *Expert Systems with Applications* 42, 21 (2015), 7424–7435.
H. Hild, J. Feulner, and W. Menzel. 1992. HARMONET: A neural net for harmonizing chorales in the style of JS Bach. In *Advances in Neural Information Processing Systems*. 267–274.
L. Hiller. 1989. *Lejaren Hiller: Computer Music Retrospective (1957-1985)*. Wergo Schallplatten. 79 pages.
A. Hiller Jr, L. and L.M. Isaacson. 1957. Musical composition with a high speed digital computer. In *Audio Engineering Society Convention 9*. Audio Engineering Society.
S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
J.H. Holland. 1992. *Adaptation in natural and artificial systems: an introductory anal-*

ysis with applications to biology, control, and artificial intelligence. MIT press.
A. Horner and D.E. Goldberg. 1991. Genetic algorithms and computer-assisted music composition. *Urbana* 51, 61801 (1991), 437–441.
C.Z.A. Huang and E. Chew. 2005. Palestrina Pal: a grammar checker for music compositions in the style of Palestrina. In *Proceedings of the 5th Conference on Understanding and Creating Music*.
T.A. Hummel. 2005. Simulation of human voice timbre by orchestration of acoustic music instruments. In *Proceedings of International Computer Music Conference (ICMC)*. 185.
D.B. Huron. 2006. *Sweet anticipation: Music and the psychology of expectation*. MIT press.
S. Johnson. 2006. The long zoom. *New York times magazine* (2006), 50–55.
R.M. Keller and D.R. Morrison. 2007. A grammatical approach to automatic improvisation. In *Proceedings, Fourth Sound and Music Conference, Lefkada, Greece, July*. 330–336.
S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. 1983. Optimization by simulated annealing. *science* 220, 4598 (1983), 671–680.
J.P. Kirnberger. 1757. Der Allezeit fertige Menuetten-und Polonoisenkomponist. *Berlin: Germany Winter* (1757).
C.L. Krumhansl. 2001. *Cognitive foundations of musical pitch*. Oxford University Press.
S. Lattner, M. Grachten, K. Agres, and C.E.C. Chacón. 2015. Probabilistic segmentation of musical sequences using restricted Boltzmann machines. In *International Conference on Mathematics and Computation in Music*. Springer, 323–334.
H.-R. Lee and J.S. R. Jang. 2004. i-Ring: A system for humming transcription and chord generation. In *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*, Vol. 2. IEEE, 1031–1034.
F. Lerdahl and R. Jackendoff. 1983. An overview of hierarchical structure in music. *Music Perception: An Interdisciplinary Journal* 1, 2 (1983), 229–252.
G.E. Lewis. 2000. Too many notes: Computers, complexity and culture in voyager. *Leonardo Music Journal* 10 (2000), 33–39.
J.P. Lewis. 1991. Creation by refinement and the problem of algorithmic music composition. *Music and connectionism* (1991), 212.
T.R. Lezon, J.R. Banavar, M. Cieplak, A. Maritan, and N.V. Fedoroff. 2006. Using the principle of entropy maximization to infer genetic interaction networks from gene expression patterns. *Proceedings of the National Academy of Sciences* 103, 50 (2006), 19033–19038.
Y.J. Liebesman. 2007. Using Innovative Technologies to Analyze for Similarity Between Musical Works in Copyright Infringement Disputes. *AIPLA QJ* 35 (2007), 331.
C.-C. Lin and D. S.-M. Liu. 2006. An intelligent virtual piano tutor. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*. ACM, 353–356.
A. Lovelace. 1843. 'Notes on L. Menabrea's 'Sketch of the Analytical Engine Invented by Charles Babbage, Esq.'". *Taylor's Scientific Memoirs* 3 (1843).
M. Marchini and H. Purwins. 2010. Unsupervised generation of percussion sound sequences from a sound example. In *Sound and Music Computing Conference*, Vol. 220.
A. Marsden. 2004. Novagen: a combination of Eyesweb and an elaboration-network representation for the generation of melodies under gestural control.. In *COST287-ConGAS Symposium on Gesture Interfaces for Multimedia Systems*.
J. McCormack. 1996. Grammar based music composition. *Complex systems* 96 (1996),

321–336.
W.S. McCulloch and W. Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5, 4 (1943), 115–133.
R.A. McIntyre. 1994. 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*. IEEE, 852–857.
M. McVicar, S. Fukayama, and M. Goto. 2014. AutoLeadGuitar: Automatic generation of guitar solo phrases in the tablature space. In *Signal Processing (ICSP), 2014 12th International Conference on*. IEEE, 599–604.
D. Meredith. 2013. COSIATEC and SIATECCompress: Pattern discovery by geometric compression. In *International Society for Music Information Retrieval Conference*.
N. Mladenović and P. Hansen. 1997. Variable neighborhood search. *Computers & operations research* 24, 11 (1997), 1097–1100.
J.A. Moorer. 1972. Music and computer composition. *Commun. ACM* 15, 2 (1972), 104–113.
R. Morales-Manzanares. 1992. Non-Deterministic Automatons Controlled by Rules for Composition. In *Proceedings of the International Computer Music Conference*. 400–400.
R. Morales-Manzanares, E.F. Morales, R. Dannenberg, and J. Berger. 2001. SICIB: An interactive music composition system using body movements. *Computer Music Journal* 25, 2 (2001), 25–36.
M.C. Mozer. 1991. Connectionist music composition based on melodic, stylistic and psychophysical constraints. *Music and connectionism* (1991), 195–211.
M. Müller and J. Driedger. 2012. Data-Driven Sound Track Generation.. In *Multimodal Music Processing*. 175–194.
J.-I. Nakamura, T. Kaku, K. Hyun, T. Noma, and S. Yoshida. 1994. Automatic background music generation based on actors' mood and motions. *The Journal of Visualization and Computer Animation* 5, 4 (1994), 247–264.
G. Nierhaus. 2009. *Algorithmic composition: paradigms of automated music generation*. Springer.
J. Nika, D. Bouche, J. Bresson, M. Chemillier, and G. Assayag. 2015. Guided improvisation as dynamic calls to an offline model. In *Sound and Music Computing (SMC)*.
H. Norden. 1969. *Fundamental Counterpoint*. Crescendo Publishing Co., Boston.
F. Pachet. 2003. The continuator: Musical interaction with style. *Journal of New Music Research* 32, 3 (2003), 333–341.
F. Pachet. 2016. A joyful ode to automatic orchestration. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 2 (2016), 18.
F. Pachet and G. Roy, P.and Barbieri. 2001. Finite-length Markov processes with constraints. *transition* 6, 1/3 (2001).
A. Papadopoulos, P. Roy, and F. Pachet. 2014. Avoiding Plagiarism in Markov Sequence Generation. In *Proceedings of AAAI*. Quebec.
G. Papadopoulos and G. Wiggins. 1999. AI methods for algorithmic composition: A survey, a critical view and future prospects. In *AISB Symposium on Musical Creativity*. Edinburgh, UK, 110–117.
R. Parke, E. Chew, and C. Kyriakakis. 2007. Quantitative and visual analysis of the impact of music on perceived emotion of film. *Computers in Entertainment (CIE)* 5, 3 (2007), 5.
M.T. Pearce, Ruiz M.H., S. Kapasi, G.A. Wiggins, and J. Bhattacharya. 2010. Unsupervised statistical learning underpins computational, behavioural, and neural manifestations of musical expectation. *NeuroImage* 50, 1 (2010), 302–313.
M. Pearce and G. Wiggins. 2001. Towards a framework for the evaluation of machine compositions. In *Proceedings of the AISB'01 Symposium on Artificial Intelligence*

and Creativity in the Arts and Sciences. 22–32.
I. Peretz, D. Gaudreau, and A.-M. Bonnel. 1998. Exposure effects on music preference and recognition. *Memory & Cognition* 26, 5 (1998), 884–902.
S. Phon-Amnuaisuk and G. Wiggins. 1999. The four-part harmonisation problem: a comparison between genetic algorithms and a rule-based system. In *Proceedings of the AISB'99 Symposium on Musical Creativity*. 28–34.
R.C. Pinkerton. 1956. Information theory and melody. *Scientific American* 194, 2 (1956), 77–86.
J. Polito, J. Daida, and T. Bersano-Begey. 1997. Musica ex Machina: Composing 16th-Century Counterpoint with Genetic Programming and Symbiosis.. In *Evolutionary Programming VI (Lecture Notes in Computer Science)*, Vol. 1213. Springer, 113–124.
A. Prechtl, R. Laney, A. Willis, and R. Samuels. 2014a. Algorithmic music as intelligent game music. In *AISB50: The 50th Annual Convention of the AISB, 1-4 April 2014, London, UK*.
A. Prechtl, R. Laney, A. Willis, and R. Samuels. 2014b. Methodological approaches to the evaluation of game music systems. In *Proceedings of the 9th Audio Mostly: A Conference on Interaction With Sound*. ACM, 26.
J. Pritchett. 1994. The Completion of John Cage's Freeman Etudes. *Perspectives of new music* (1994), 264–270.
D. Psenicka. 2003. Sporch: An algorithm for orchestration based on spectral analyses of recorded sounds. In *Proceedings of International Computer Music Conference (ICMC)*. 184.
D. Radicioni, L. Anselma, and V. Lombardo. 2004. An algorithm to compute fingering for string instruments. In *Proceedings of the National Congress of the Associazione Italiana di Scienze Cognitive, Ivrea, Italy*.
A. Radisavljevic and P. Driessen. 2004. Path difference learning for guitar fingering problem. In *Proceedings of the International Computer Music Conference*, Vol. 28.
R. Raines. 2015. *Composition in the Digital World: Conversations with 21st Century American Composers*. Oxford University Press. 241 pages.
D. Ralley. 1995. Genetic algorithms as a tool for melodic development. *Urbana* 101 (1995), 61801.
C. Roig, L. J. Tardón, I. Barbancho, and A. M. Barbancho. 2014. Automatic melody composition based on a probabilistic model of music style and harmonic rules. *Knowledge-Based Systems* 71 (2014), 419–434.
D.E. Rumelhart, G.E. Hinton, and R.J. Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5, 3 (1988), 1.
D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1985. *Learning internal representations by error propagation*. Technical Report. DTIC Document.
J. Rutherford and G Wiggins. 2002. An experiment in the automatic creation of music which has specific emotional content. In *Proc. for the 7th International Conference on music Perception and Cognition*.
S. I. Sayegh. 1989. Fingering for string instruments with the optimum path paradigm. *Computer Music Journal* (1989), 76–84.
I. Schankler, E. Chew, and A. François. 2014. Improvising with Digital Auto-Scaffolding: How Mimi Changes and Enhances the Creative Process. In *Digital Da Vinci*. Springer, 99–125.
V. Sébastien, H. Ralambondrainy, O. Sébastien, and N. Conruyt. 2012. Score Analyzer: Automatically Determining Scores Difficulty Level for Instrumental e-Learning.. In *ISMIR*. 571–576.
R. Siddharthan. 1999. Music, mathematics and Bach. *Resonance* 4, 5 (1999), 61–70.
I. Simon, D. Morris, and S. Basu. 2008. MySong: automatic accompaniment generation for vocal melodies. In *Proceedings of the SIGCHI Conference on Human Factors in*

Computing Systems. ACM, 725–734.
P. Smolensky. 1986. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Information Processing in Dynamical Systems: Foundations of Harmony Theory. *MIT Press, Cambridge, MA, USA* 15 (1986), 18.
M.J. Steedman. 1984. A generative grammar for jazz chord sequences. *Music Perception* (1984), 52–77.
B.L. Sturm. 2006. Adaptive concatenative sound synthesis and its application to micromontage composition. *Computer Music Journal* 30, 4 (2006), 46–66.
T. Tanaka, B. Bemman, and D. Meredith. 2016. Constraint programming formulation of the problem of generating Milton Babbitt's all-partition arrays. In *Proceedings of the 22nd International Conference on Principles and Practice of Constraint Programming*. Toulouse, France.
B. Thom. 2000. BoB: an interactive improvisational music companion. In *Proceedings of the fourth international conference on Autonomous agents*. ACM, 309–316.
A. Tidemann and Y. Demiris. 2008. A Drum Machine That Learns to Groove. In *KI 2008: Advances in Artificial Intelligence*. Springer, 144–151.
P.M. Todd. 1989. A connectionist approach to algorithmic composition. *Computer Music Journal* (1989), 27–43.
P. Toiviainen. 1995. Modeling the target-note technique of bebop-style jazz improvisation: An artificial neural network approach. *Music Perception* (1995), 399–413.
N. Tokui and H. Iba. 2000. Music composition with interactive evolutionary computation. In *Proceedings of the Third International Conference on Generative Art*, Vol. 17:2. 215–226.
M.W. Towsey, A.R. Brown, S.K. Wright, and J. Diederich. 2001. Towards melodic extension using genetic algorithms. *Educational Technology & Society* 4, 2 (2001), 54–65.
C.P. Tsang and M. Aitken. 1999. Harmonizing music as a discipline of constraint logic programming. In *International Computer Music Conference*.
D.R. Tuohy and W.D. Potter. 2005. A genetic algorithm for the automatic generation of playable guitar tablature. In *Proceedings of the International Computer Music Conference*. sn, 499–502.
P.N. Vassilakis. 2005. An improvisation on the Middle-Eastern mijwiz; auditory roughness profiles and tension/release patterns. *The Journal of the Acoustical Society of America* 117, 4 (2005), 2476–2476.
RODNEY WASCHKA II. 2007. Composing with genetic algorithms: GenDash. In *Evolutionary Computer Music*. Springer, 117–136.
G. Weinberg and S. Driscoll. 2006. Toward robotic musicianship. *Computer Music Journal* 30, 4 (2006), 28–45.
I. Xenakis. 1992. *Formalized Music: Thought and mathematics in composition*. Number 6. Pendragon Pr.
J. Yamato, J. Ohya, and K. Ishii. 1992. Recognizing human action in time-sequential images using hidden markov model. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*. IEEE, 379–385.
L. Yi and J. Goldsmith. 2007. Automatic Generation of Four-part Harmony. In *BMA, CEUR Workshop*.
R.B. Zajonc. 1968. Attitudinal effects of mere exposure. *Journal of personality and social psychology* 9, 2p2 (1968), 1–27.

Received August 2016; revised 000; accepted 000