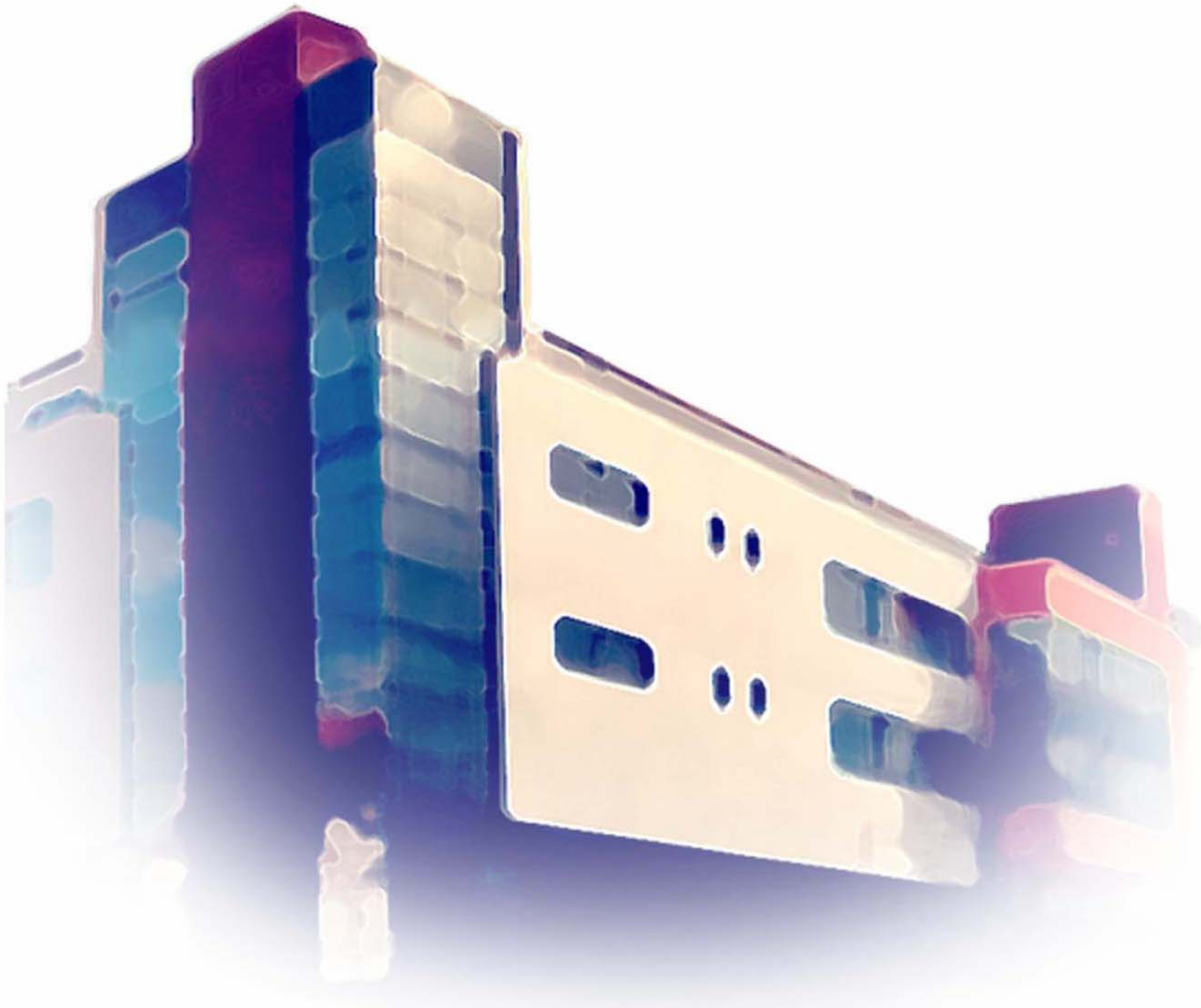


Manuel Ebert

*A Phenomenological Approach to
Artificial Jazz Improvisation*



PICS

Publications of the Institute of Cognitive Science

Volume 1-2010

ISSN: 1610-5389

Series title: PICS
Publications of the Institute of Cognitive Science

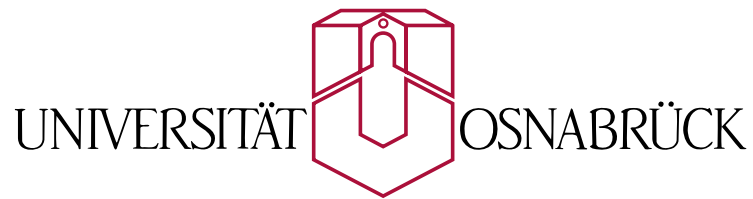
Volume: 1-2010

Place of publication: Osnabrück, Germany

Date: September 2010

Editors: Kai-Uwe Kühnberger
Peter König
Sven Walter

Cover design: Thorsten Hinrichs



Manuel Ebert

**A phenomenological approach to
artificial jazz improvisation**

A thesis submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science
Department of Cognitive Science
University of Osnabrück

First Supervisor: Priv. Doz. Dr. Helmar Gust
Second Supervisor: Prof. Dr. Bernd Enders
Submitted: May 23, 2009

Abstract

This thesis proposes a novel approach towards the design of artificially creative systems based on a phenomenological account of human creativity and the analysis of existing approaches towards the same task through the exemplary study of artificial jazz improvisation. It will be argued that in symbolic approaches towards creativity, similarity measures between symbols play a vital role in the modelling of human phenomena. An implementation based on the concepts and mechanisms found through the analysis of existing systems will be given and evaluated with respect to the phenomenology of human jazz performances.

Acknowledgements

The author would like to thank his supervisors, Helmar Gust and Bernd Enders, for their support and patience in answering numerous methodological questions; Tilman Weyde of the City University London for the continuous correspondence and Bob Keller at the Harvey Mudd College for the transcriptions of improvisations he provided with *Impro-Visor*, without which this thesis would have taken a lot longer.

Contents

1	Introduction	9
1.1	Introduction into the music theory of jazz music	10
1.2	A phenomenological approach to musical improvisation	11
1.2.1	Qualities and properties of human musicians	11
1.2.2	Existing approaches towards artificial jazz improvisation	12
1.3	Analysis of existing approaches	17
2	Proposal of a new approach	19
2.1	On the role of similarity measures in creativity	19
2.2	Overview of the suggested system	20
2.3	Creation phase	20
2.3.1	Pattern acquisition and processing	20
2.3.2	Rhythmic similarity measures	28
2.3.3	Melodic similarity measures	31
2.4	Training phase	32
2.4.1	Finding walks through pattern space	33
2.4.2	Applying note-level rules	34
2.4.3	Learning from human feedback	35
3	Discussion and Evaluation	37
3.1	Creativity in general	37
3.2	Learning behaviour	37
3.3	Playing behaviour	39
3.4	Other limitations	39
4	Conclusion and Outlook	41
	Bibliography	41

List of Figures

1.1	Eight measures from “Take the A train” in a lead sheet and an actual performance	10
1.2	Examples of different genomes in GenJam	13
1.3	The recurrent neural network used in CHIME	15
1.4	A parse tree produced with Impro-Visor	16
2.1	Pseudo-code for creating new players	21
2.2	Number of frequent rhythmic fragments required for different coverage thresholds on the data set	24
2.3	Number of frequent melodic fragments required for different coverage thresholds on the data set using different representations	27
2.4	Distribution of note intervals on the data set	27
2.5	Similarity of rhythms	30
2.6	A depiction of the geometric distance metric	31
2.7	Similarities between melodic fragments	32
2.8	Extrapolation of transition matrices	33
2.9	Pseudo-code of the training algorithm	34
2.10	The user interface	35
3.1	A performance of “Anthropology”	38

List of Tables

2.1	List of all tunes in the data set	22
2.2	Distribution of chord extensions on the data set	23
2.3	Frequent rhythmic fragments on the data set	25
2.4	Frequent melodic fragments on the data set	26

*“Life is like music; it must be composed by ear,
feeling, and instinct, not by rule.”*

Samuel Butler

CHAPTER 1

Introduction

Many people are quite quick to judge computers and artificial intelligence when it comes to how “human” modern AIs are, and many claim that AI will indeed never achieve human qualities as computers cannot feel or love or be creative; too often these deficiencies are then attributed to the absence of a “soul” or any other alleged entity vague enough to resist a scientific approach. With no intent to engage in a philosophical debate about the nature of creativity here, I believe that there are computational models for what we call creativity (or at least for those phenomena we attribute to creativity), and what’s more, that music is a particularly good domain to study creativity with the use of computers; mainly for two reasons:

1. We have a fairly complete structured formalisation of music through traditional notation and modern protocols and specifications such as MIDI and MusicXML. Similar formalisations exist for language (although they usually can’t account for all irregularities), but painting and other visual arts are much harder to represent in a way that allows for extraction of elementary features and components.
2. Elements of music usually have much less symbolic designation of things in the “real world”, which again does not hold for many visual arts or language. Although many musical elements or phrases may be connected to specific moods, the “meaning” of musical elements and thus the prior knowledge required to produce music is assumed to be less complex than that of e.g. paintings and poems.

Unfortunately, studies in artificial creativity up to now are little systematic as there are only few and very vague commonly accepted paradigms and ground facts on which analysis and synthesis of creative systems is possible. This thesis will therefore follow a phenomenological approach by first discussing a number of qualities, properties and phenomena that human musicians show when improvising music and learning to improvise music and thereafter examine the mechanisms of existing systems that bring about those properties in contrast to those that do not.

Considering creativity as a problem solving task in the widest sense, this thesis proposes that one of the essential features of creativity is to be able to judge the similarity of solutions within the solution space, based on the similarity of their constituents, e.g. humans can address the similarity of two paintings by comparing the colour palette, style of strokes, motives, arrangement of objects, lighting and so on. Moreover I will claim that without this ability, creativity as we know it would not exist and any creative products had to be products of chance and randomness.

The mechanisms discovered through analysis of existing systems along with a similarity relation on the basic constituents of music, melody and rhythm, will then be used to propose a new approach towards artificial improvisation which enables a system to learn by both listening and playing and furthermore develop

a unique style based on different training samples and user feedback, and be moreover able to improvise over previously unheard tunes. By implementing and training the system I will then aim to demonstrate that the proposed framework both in theory and in practise exhibits the qualities and phenomena described earlier on.

The thesis will conclude with a review of the existing approaches and comparison to the proposed one and will finally suggest extensions that might improve the computational and aesthetic performance of the system.

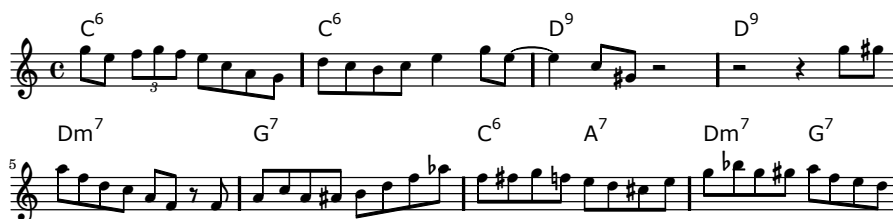
1.1 Introduction into the music theory of jazz music

Jazz as a genre now spans over more than a century of constant change so that modern flavours such as *smooth jazz*, *nu jazz* and *jazz pop* have little or nothing to do with the roots of jazz such as 1890s *ragtime* or *New Orleans* and *Dixieland* in the first two decades of the past century. This thesis will therefore focus on the principles of jazz music of the late 40s and 50s, namely *bebop*, *hard bop* and *cool jazz*. Despite their different acoustic qualities, they share the same “framework”: collective improvisations based on so-called lead sheets.

For those unfamiliar with music theory or jazz music, some of the understood phenomena about such music will now briefly be explained from a computational perspective. To start with, the very basic element of music are notes, which can be seen as triplets of pitch, volume and duration. Melodies are temporal sequences of notes (and rests) and chords are sets of simultaneously played notes. Lead sheets usually provide the melody of the tune and the chord structure. The task of bebop improvisation is to play notes in accordance to the chord structure of the lead sheet.



(a) The original lead sheet (The Real Book vol. 1)



(b) Transcription of an improvisation by Harold Land

Figure 1.1: The first eight measures from “Take the A train” illustrate the difference between what is written in a lead sheet and what is actually being played.

Most typical recordings in those styles then follow, according to Walker (1997), the same structure: first the soloist (typically a saxophonist or the pianist if there are no other solo instruments) plays the original tune with only minor variations. The next repetition of the chorus will feature the musician playing an improvisation over the chord structure of the tune that may have very little to do with the actual melody. Other musicians will then take their turns in playing solos, or special structures such as “trading fours” (four measures of an instrumental solo alternate with four measures of a drum solo, or more rarely, somebody else’s solo) will be played. Figure 1.1 depicts an excerpt from a lead sheet notation of a jazz standard and a transcription of an improvisation

over this standard.

It is noteworthy that sometimes accompanying musicians (e.g. the bass player) were just given the chord progression of a tune and could play along. This works because the chords and the key define what sounds consonant or harmonic and what will be dissonant. Let's have a short look at this by looking at the seventh measure of the example. The key of the example is *C major*, the chords in this measure are C^6 and then A^7 (the tonic with added sixth and then major tonic-parallel with seventh¹). So for the first two beats, the chord tones are C, E, G and A. Furthermore D, F and B will sound consonant as they are in the scale of a C major tonic, however the latter two will usually not be seen on a long on-beat note as they are only a semitone away from the chord tones E and C and will therefore be likely to sound dissonant to the accompaniment. For the second half of the measure the chord tones are A, Cis, E and G; with B, D and Fis also belonging to the scale. All tones on full beats are chord or scale tones; the Fis on the second eighth serves as an approach tone to the following G. By submitting to those rules the musicians can ensure that they will produce a consonant sound although none of them knows what the others are going to play next.

However, Ramalho and Ganascia (1994) noted that musicians will generally not be able to justify all decisions they make that lead towards their actual improvisation. The problem of filling the immense gap between the mere grid of chords and a well-sounding, interesting improvisations is what makes the study of jazz performance interesting for cognitive scientists.

1.2 A phenomenological approach to musical improvisation

1.2.1 Qualities and properties of human musicians

Usually the way things are designed heavily depends on the way the success of the design is going to be evaluated – an evaluation measures those criteria that matter to the designer, hence the designer tries to build things in a way that maximises the evaluation results. So instead of asking how to design a system that can improvise bebop we could rather ask how such systems can be evaluated and adapt our design strategies accordingly. In the simplest case, one could judge how musical or 'jazzy' output of a given system is. This, however, neglects the slightly different goals of different approaches and more importantly relies on rather subjective measures. We therefore propose to assess such a system by which phenomena it could principally produce and what behaviour it usually shows. If we had a comprehensive list of behaviours and qualities that we loosely connect to people being creative, we could analyse existing system that bring about those behaviours and phenomena and abstract the mechanisms that are responsible for that very thing. If a system produces all the phenomena and behaviours specified then it will be either indistinguishable from real creativity – regardless of any explicit definition of creativity – or the list is not complete.

On the following pages, I will state some behaviours and phenomena that can be observed in human jazz learners and performers and discuss their applicability to artificial learners and performers.

Creativity in general

Johnson-Laird (1988) notes that improvisation, being a creative task, is characterised by non-determinism and the absence of well-defined goals. However

¹One can rightfully argue that the C^6 - A^7 actually serves as a dominant-parallel and major dominant to the next Dm^7 chord, but for our purposes we can ignore subtleties of functional interpretation (if jazz with its many minor sevenths doesn't render functional interpretation useless, that is - but that's another debate).

already twenty years ago there were, according to Taylor (1988), more than 60 different models and definitions of creativity of varying explanatory depth and plausibility, and there is little evidence for any of them being the definite answer to the question of the nature of creativity, nor does it seem like there is any hope that this question will be settled any time soon. In fact, many people believe that there can never be any scientific explanation of creativity². This might well be a reason that the approaches to artificial improvisation covered in the next sections (rightfully?) follow a strategy of forthright ignorance by simply disregarding the terminological, psychological and modelling ballast that comes with the fact that they are implementing a creative process. Having said that, there are some observations on creativity that hold regardless of the model drawn on to explain these observations. Poincaré (1913) remarked that “to create consists of making new combinations of associative elements which are useful” – the matter of association in creativity will be elaborated in section 2.1; particular attention should be paid to what exactly these “elements” consist of.

Learning behaviour

Most contemporary European jazz musicians received, at least for some time, a classical music education and have developed technical skills on one or more instruments before starting to play jazz³. Although they will be familiar with the scales that different chords provide, classically trained musicians will generally not be able to come up with an innovative jazz improvisation if given a lead sheet, more likely will they, more or less, reproduce the melody with slight rhythmic variations of which they (intuitively) think that they might sound jazzy. Baker (1978) points out that Jazz musicians don’t learn by studying in conservatories but rather by *listening* and *practising*, and both depend on each other: an experienced jazz musician will find different things in an improvisation he listens to and incorporate that into his own style and repertoire than a novice. However at no point does a jazz learner sound “chaotic” or random – quite the opposite: if unsure what to play he will stick to more conventional licks.

Playing behaviour

Amongst the obvious phenomena of jazz music is that musicians can play and improvise over almost any unknown tune if given a lead sheet. However after having played a tune a few times musicians will sometimes stick to a certain improvisation which they like most and will play that (with minor variations) in performances. During jam sessions or performances, musicians can respond to musical material played by other players to maintain a coherent style in collaborative improvisations, and even within a single improvisation there are often motifs that are heavily repeated and referenced. Sloboda (1985) draws our attention to the seemingly subtle phenomena that when playing, musicians don’t just decide “which note to play next” but rather decide on whole chunks of notes at once. Nonetheless, Widmer (2001, 2005) showed that – at least in classical music – there are note-level rules that seem to be more or less innate to pianists that determine micro-timing, accentuation and other note-level features; see section 4.

1.2.2 Existing approaches towards artificial jazz improvisation

In this section several existing systems that generate Jazz performances (or are closely related to that task) will be introduced. The main focus is on different

²Which in most cases is a metaphysical stance based on the neglect and misunderstanding of what science and creativity are about rather than a conclusion based on the comprehension and awareness of these very things.

³This doesn’t necessarily hold for performers of the ‘rest of the world’; particularly many black American performers in the 50s and 60s were raised with jazz.

ways to represent and encode music, but the algorithms used to produce actual music from what is represented in the systems will also be touched on. This overview is far from being comprehensive and the reader is encouraged to look into the work of Dahlstedt and Nordahl (2001) on evolutionary “living” melodies and the statistical approaches of Allan and Williams (2005) and Thom (2000) (the latter with a focus on solo trading) which will not be covered in this section. However I do think that this selection of systems gives a good insight into the kind of work and different approaches towards the aforementioned task and will serve as examples to describe common patterns and issues.

Genome-based representations

Similar to Dahlstedt and Nordahl (2001) and Papadopoulos and Wiggins (1998), Biles (1994) represents melodies in artificial genomes. In one population each individual genome encodes one measure of music by specifying one of 16 possible events for each eighth beat of the measure: 1 to 14 denote the onset of one of fourteen different tones (which are relative to the scale the current chord provides, i.e. a 3 would be an E if the chord was C^6 but a C^is if the chord was A^7), 0 denotes a rest and 15 a “hold” – which extends the duration of the previous note or rest. Thus every genome of a measure individual is 32 bits long. In a second population the genomes encode a sequence of four individuals of the measure population (cf. figure 1.2); as this population is limited to 64 individuals, every phrase genome will be 24 bits long. Once two suitable populations are bred, GenJam builds an improvisation using a chord progression file (which also includes MIDI sequences for bass, drums and piano) and selecting individuals from the phrase population which then each supply melodic material for four measures. The relative tones encoded by the measures are translated into absolute tones by built-in scales for the aligned chord in the chord progression file and sent to a MIDI device. The listener can give feedback to the system by typing **g** for good or **b** for bad (even several times), which modifies the fitness values of the current phrase and measure individuals.

#23	-12	57	57	11	38
-----	-----	----	----	----	----

(a) Phrase Population

#11	6	9	7	0	5	7	8	7	5
-----	---	---	---	---	---	---	---	---	---

# 38	-4	7	8	7	7	15	15	15	0
------	----	---	---	---	---	----	----	----	---

# 57	22	9	7	0	5	7	15	15	0
------	----	---	---	---	---	---	----	----	---

(b) Measure population

Figure 1.2: Examples of genomes of individuals in (a) the phrase- and (b) measure population and their fitness in GenJam (adapted from Biles (1994)).

If the system is in *learning mode*, the phrase individuals are selected at random while in *demo mode* phrases are selected by a tournament process which not only takes the fitness of the phrases into account but also the fitness of constituent measures. In a third mode, called *breeding mode*, half of the population is replaced by mutated offsprings after each solo. The selection process works by randomly grouping the individuals into families of four and then keeping the two family members with the highest fitness as parents and replacing the other two by offsprings of those parents. The children are created by first applying a random single-point crossover to the parents’ genomes (i.e. each child’s genome will be identical to its fathers genome up to some point and from there on iden-

tical to its mother’s genome). One of the two children of a family will be kept intact, the other one mutated by one of the following musically “meaningful” mutation operators:

- (a) Reversing the notes (blocks of 4 bits)
- (b) Rotating the notes by a random number of notes
- (c) Inversion
- (d) Sorting notes in ascending order (rests and holds are not affected)
- (e) Sorting notes in descending order (rests and holds are not affected)
- (f) Transposing notes by a random number $\in [1, 14]$

Phrases are mutated by applying one of the following operators:

- (a) Reversing the measures (blocks of 5 bits)
- (b) Rotating the measures by a random number of measures
- (c) Genetic repair: replace measure with worst fitness by random measure index
- (d) Super phrase: replace all measures by winners of four three-measure tournaments
- (e) Lick thinner: replace a random measure by the one that occurs most often in the phrase population
- (f) Orphan Phrase: replace all measures by winners of four three-measure tournaments. where the winner is the measure that occurs least frequently on the phrase measure (maintains genetic diversity)

Two major differences to plain genetic algorithms (i.e. Mitchell (1996)) are obvious: firstly, GenJam uses the whole population to create a performance and not just the fittest. Secondly, the mutation operators are not “dumb” [sic] with respect to the structures they alter. Other remarkable features of GenJam include that the chord progression file can define a swing rhythm by which two quavers will sound as a triplet of a crotchet and a quaver. Furthermore, a collaborative mode exists in which GenJam will listen for MIDI input, convert this input into measure and phrase individual, apply the above mentioned genetic operators and play the results – which resembles “trading fours”.

Neural networks

Franklin (2001) utilises a recurrent neural network that can memorise three different melodies for her CHIME framework (“*computer human interacting musical entity*”). Recurrent networks differ from plain vanilla networks in that the output is used as part of the input; the learning data are then usually sequences. This has the effect that the previously seen data sample can be used as the context of the current sample. The network in use is depicted in figure 1.3. The exact network topology is as follows:

- Three *plan input* units. They indicate the index of one of the three melodies that are learned using *1-of-N* encoding (1-0-0 for 1, 0-1-0 for 2 etc.)
- Twelve *chord input* units; each unit corresponds to one chromatic note and is set to 1 if that note is part of the input chord.

- 26 *context input* units that have recurrent feedback connections from the output units. When propagating the error, a method called *teacher forcing* is used to update the self-connecting weights with the target output rather than the actual output.
- 50 *hidden units* (this number was experimentally derived)
- 26 *output units*, consisting of 24 units representing the chromatic notes of two octaves, one representing a rest and a new note indicator. Among the first 25 outputs, the one with the greatest value is selected as the next note that is being played; if the new note indicator is above a certain threshold a new note onset is played, otherwise the last note is being held for another time step.
- The learning rate was set to $\varepsilon = 0.075$

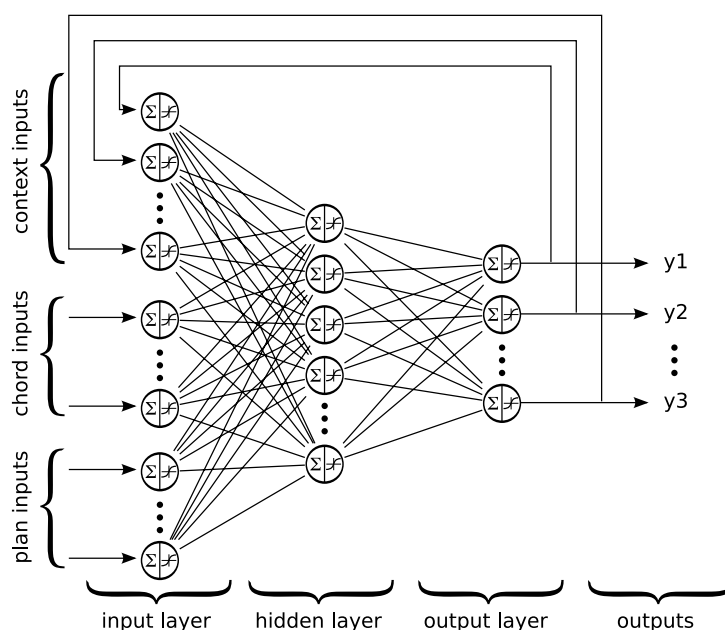


Figure 1.3: The recurrent neural network used by Franklin (2001) in phase 1.

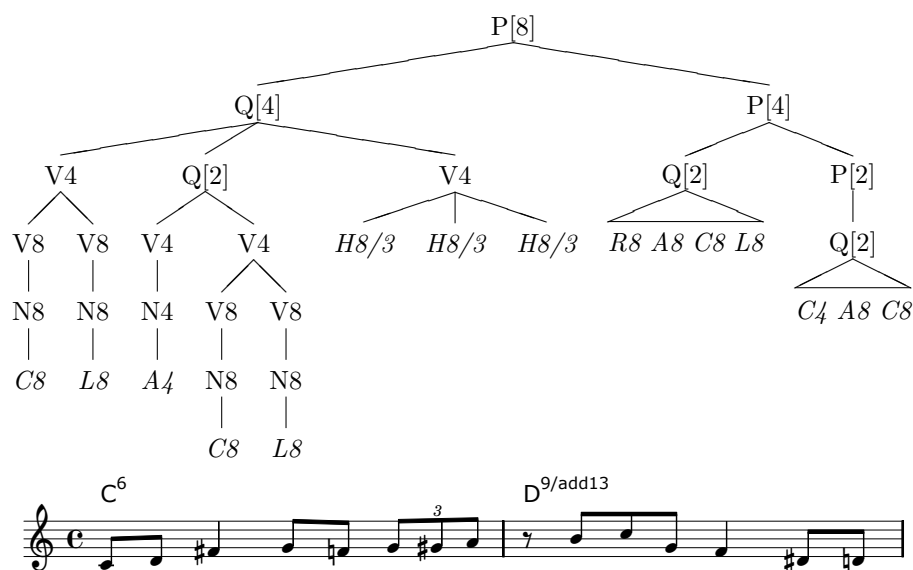
The network was then used to memorise three songs; each song contained 192 semiquavers and was presented to the network 15,000 times until the songs as replayed by the network were “easily recognised”. After training, the network can be used to play trading fours with a human player: the recurrent connections are removed and all plan inputs set to 1. Each time the human has played four measures, the recorded notes of the human solo are used as inputs for the network, and together with the weights learned from the three songs the network produces a melody of its own – though the network can only produce as much novel music as it receives input stored from the last four measures the human played.

Franklin extended the network by transforming it into an actor-critic reinforcement learner with human input, which will not be covered here due to space limitations. Also, we would like to draw the reader’s attention to the work of Eck and Schmidhuber (2002a,b) which uses *long short-term memory (LSTM) networks* to allow for the inclusion of a longer temporal context into the input.

Rule-based approaches

I will cover two different rule-based approaches in this section; one using *context-free grammars* (CFGs) for note-by-note construction and the other one using more abstract sets of *possible actions*.

Keller and Morrison (2007) successfully used probabilistic CFGs to create jazz melodies for their tool *Impro-Visor* (cf. Keller et al. (2006)). A probabilistic CFG is a four-tuple $(\Sigma, \mathcal{N}, \mathcal{R}, S)$ where Σ is a set of terminal symbols (actual constituents of what is going to be produced), \mathcal{N} is a set of non-terminal symbols, \mathcal{R} is a set of rules that relate every non-terminal symbol $N \in \mathcal{N}$ to a number of symbols in $\Sigma \cup \mathcal{N}$, each with an assigned weight (or probability) w and $S \in \mathcal{N}$ is the start symbol.



of each of the above mentioned symbols that depend on the chord that is being played. Also certain parameters for non-terminal symbols are used which strictly speaking are not context-free anymore, but ensure that the sequence has a fixed length, that approach tones are always succeeded by chord tones etc.

The melodies created by their grammar are mostly consistent with what you expect from jazz melodies, and Keller (a part-time jazz teacher) judges the better results to be of quality comparable to college-level jazz students. An example is given in figure 1.4.

Ramalho and Ganascia (1994) in contrast define more abstract rules and constituents in their *PACT* framework to make improvisation a classical problem solving task. First, they introduce two basic components of their system: *potential actions* (PACTs) and *musical memory*. A potential action may be thought of as the intentions musicians have during a performance. The authors distinguish between PACTs with a low level of abstraction that directly influence rhythm, pitch and amplitudes of notes (such as “play loudly”) and highly abstracted PACTs like “play the same lick, but transposed one semitone higher”. Furthermore they distinguish *procedural* PACTs (such as “play an arpeggio”) and *property-setting* PACTs, e.g. “play bluesy”. PACTs may combine to a new PACT, which entails two important properties of PACTs:

1. *Playability*: the reciprocal of the amount of information that’s missing to actually play the PACT
2. *Combinability*: whether PACTs can combine to produce more playable PACTs and to exclude contradicting PACTs (such as “play loudly” and “play quietly”).

The problem solving task is then characterised by starting with an empty set of bars and possibly a set of abstract PACTs and the goal is to fill the bars with notes through combining PACTs while obeying the constraint of combinability (“*the goal is to play!*”).

As there is no guarantee that a set of PACTs contains all information that is required to produce a full melody (in other words, that it is playable) the authors introduce *musical memory*. While the PACTs described above were manually implemented by formalising expert knowledge of jazz improvisation, the musical memory represents what a musician learns by listening to jazz music. Hence the musical memory contains (typically low-level) PACTs that have been automatically obtained (though with human guidance) from transcriptions of real improvisations such as rhythm patterns, melody fragments etc.

A reasoner is then used to produce unique playable PACTs based on a chord grid and the *reasoner’s mood*. The mood is controlled by a *perception module* that allows interaction of the external environment and the musician, however the authors are quite unspecific as to how exactly this module works. Similar to a planning system, the reasoner uses three operators: *combine* to combine two compatible PACTs into a new one, *delete* to resolve incompatibilities between PACTs and *add* to include PACTs from the musical memory that provide the information missing to produce a unique playable PACT.

1.3 Analysis of existing approaches

To summarise section 1.2.1, (most) human jazz music is characterised by the following features:

- (a) Non-determinism
- (b) Absence of well-defined goals

- (c) Combination of associative elements
- (d) Learning process starts with small variations of known tunes
- (e) Learning by listening *and* playing
- (f) Some known rules (e.g. chord scales)
- (g) Musicians play by chunks of notes rather than note-by-note
- (h) Possibly note-level rules for expression, articulation, timing etc.
- (i) Musicians can respond to musical input
- (j) Use of repeating motifs
- (k) Musicians can improvise over previously unknown songs but will also retain good improvisations for known songs.

By comparing the approaches described above (and the musical output produced by their implementations) it can be seen that despite the diversity of representations and algorithms used, the same mechanisms and underlying assumptions are used to bring about these properties in different systems:

- (c) Although this phenomenon offers a rather wide range for interpretation depending of what one considers to be an associative element, the most conspicuous elements are those on the level of the “chunks” of notes mentioned in (g). GenJam provides a good example of how to engender that very effect by representing chunks as individuals of the measure population and the association of chunks by the phrase population (whether a measure is a good approximation to a chunk will be discussed in section 3.4)
- (d, e) Systems that can learn by listening (supervised learning) before receiving human feedback learn a lot faster (in real time, not necessarily in iterations) and will have a solid basis before presenting material to the human mentor. This can be nicely seen by comparing GenJam with CHIME (with the reinforcement extension).
- (f) Most systems map the generated pitches to the scale provided by the current chord to ensure some harmony. Unfortunately there seems to be no study on how it affects the learning behaviour if the chord scales have to be learned as well; and it is hard to isolate this factor from systems that do (ie. CHIME). Nevertheless, we can assume that this is an useful way of incorporating prior knowledge into the system.
- (g) Approaches that represent knowledge using ‘patches’ of music like GenJam (where these patches are the individuals of a population) or the PACT framework tend to produce music with more local coherence than approaches that will produce music one note after another (i.e. CHIME or stochastic approaches using Markov models such as Allan and Williams (2005)).
- (h) Note-level rules for timing might be conceivable for approaches that work on note-level such as CHIME or HMMs, however Widmer’s work might be used as an add-on layer to patch- or pattern-based approaches as well.
- (i) In order to respond to human musicians, musical input has to be parsed and transformed to the representation the learner uses for its own musical material. Therefore it must be possible to reverse the output (actual music) to the inner representation of musical properties; at least to the level from where musical operators can be applied. It is not immediately clear how this can work in grammar-based approaches (although creating

a parse-tree and modifying it might work, however the cognitive motivation for this approach is questionable) and even more difficult when using stochastic models on note-level basis.

- (j) The intentional use of repeating motifs is, by definition, impossible for context-free grammars and is left to chance in all other systems with the notable exception of the PACT framework that might indeed have context-sensitive structures.
- (k) All the approaches described above lack the ability to generate unique and recognisable solos for different tunes without having been trained for each new tune separately; or at least none of them describes how tune-specific knowledge could be incorporated into their representation and algorithms.

CHAPTER 2

Proposal of a new approach

Based on the findings obtained through the deconstruction of previously presented systems, this section will expound the systematic assembly of the exposed mechanisms into a system that displays the traits and qualities of human jazz performances and performers inspected in the previous chapter.

2.1 On the role of similarity measures in creativity

It can hardly be argued that humans do not have an intuitive sense of similarity, no matter whether it is innate or culturally dependent, even if they cannot verbalise it. This sense of similarity seems to be an intrinsic property of creativity, as without it any product of creativity would be unrelated to any previously existing things, and although such a product may be the result of an analytic process this process will therefore lack the phenomenology of creativity. However, there are also formal arguments for the use of similarity in creative processes (and in fact for a lot of problems related to search and planning in “good old-fashioned” artificial intelligence): assuming that memories and ideas are not explicitly stored in our brain as discrete entities but rather as implicit properties of the structure of the brain, e.g. synaptic connections, studies on artificial neural networks suggest that the structure bears information on the correlation of afferent and efferent signals, i.e. the input and the output of networks (which in the case of the brain correspond to external stimuli and memories that are invoked by those stimuli). Examples of such networks are Hopfield networks (which are explicitly trained with the correlation between training patterns) and Multi-Layer perceptrons. This process of correlating input with output by means of previously learned correlation may be thought of as association and is also an important mechanism in dealing with incomplete knowledge (which, for human beings, may be the case almost every time and on every level of understanding). In systems where information is explicitly stored, such as it is the case in many classical applications of artificial intelligence however, there is no or only marginal knowledge of how these bits of information are correlated. Similarity measures are one way to allow associative processes and processes working on incomplete knowledge in those systems. One might even claim that similarity is merely another word for the way information is stored in our associative memory, which also explains why we can easily break down the similarity of two objects to the similarity of their components and the components’ components respectively.

In the following sections, a novel approach to artificial improvisation based on the concept associative components and similarity as means of association between components will be described.

2.2 Overview of the suggested system

The framework proposed can be split into two phases; the creation phase and the training phase. In the creation phase, transcriptions of solos are read and fragmented and a player object is instantiated with the information gained from those transcriptions. In the training phase, an instantiated player may use the previously gained knowledge to perform new solos and gain reward or punishment from the user which is then incorporated into its prior knowledge.

The aim of the creation phase is to estimate all probability distributions such that the conditional probability of any melodic and rhythmic fragment being played next given which fragments are being played right now can be computed:

$$\Pr(r_j, m_q | r_i, m_p) \approx \Pr(r_j | r_i) \cdot \Pr(m_q | m_p) \cdot \Pr(r_j, m_q) \quad (2.2.1)$$

for all rhythmic fragments $r_i, r_j \in R$, the set of frequent rhythmic fragments learned by the player, and accordingly $m_p, m_q \in M$, the set of frequent melodic fragments learned by the player. Equation 2.2.1 is the so-called performance task, along with the algorithm that combines a rhythmic with a melodic fragments. However, we would like to include specific knowledge for different tunes $\Theta \in T$ such that the probabilities described in equation 2.2.1 change according to the tune over which the solo is being performed. Formally, this is done by altering equation 2.2.1:

$$\begin{aligned} \Pr(r_j, m_q | r_i, m_p, \Theta) \approx & \Pr(r_j | r_i) \cdot \Pr(m_q | m_p) \cdot \Pr(r_j, m_q) \\ & \cdot \Pr(r_j | r_i, \Theta) \cdot \Pr(m_q | m_p, \Theta) \cdot \Pr(r_j, m_q, \Theta) \end{aligned} \quad (2.2.2)$$

We will call this the layered performance task, as we can think of tunes as additional layers of probabilities. The conditional probability distribution can be thought of as being a Markov model of order one, and that mere thought provides us with a great variety of tools for dealing with Markovian decision processes, e.g. from computational linguistics. However, given the nature of the data from which the probability distributions are estimated, they can also be thought of as resembling an associative process that lets musicians think of the next fragment to play based on what they are playing right now. To avoid precision issues in floating point operation when dealing with many fragments (and thus very small probabilities), it might be necessary to operate on the sum of logarithms of probabilities instead of the product; however in tests with up to forty rhythmic and melodic fragments no visible errors occurred.

The creation algorithm is sketched in figure 2.1, further explanations are given in the subsequent sections. Everything is implemented in *Python 2.5*¹ (see Van Rossum and Drake (2003) for details) with the help of the *PyGame* module² for audio output.

2.3 Creation phase

2.3.1 Pattern acquisition and processing

The data set in use is based on the transcriptions of improvisations that ship with Impro-Visor 3.39 (see Keller et al. (2006)). A detailed listing of tunes used is given in table 2.1. All transcriptions used are in 4/4 rhythm and meant to be played in a swingy style (two quavers becoming a triplet of a crotchet and a quaver). Table 2.2 shows the distribution of chords; as the frequencies of all chord extensions that do not alter the scale given by the chord root (e.g. Xm^7 in contrast to Xm^{7b5} , which introduces the flat fifth into the scale) already sums up to 0.927 we can assume that in most cases the distinction between major

¹Available at <http://www.python.org/>

²Available at <http://www.pygame.org/>

1. **require** transcription files
2. notes \leftarrow **read** transcription files
3. $N(t) \leftarrow$ **calculate** conditional note functions **from** notes
4. $F^* \leftarrow$ **fragmentise** notes
5. $R^* \leftarrow$ **get** rhythmic fragments from F^*
6. $M^* \leftarrow$ **get** melodic fragments from F^*
7. $R \leftarrow$ **select** frequent rhythmic fragments **from** R^*
8. $M \leftarrow$ **select** frequent melodic fragments **from** M^*
9. $\mathcal{R} \leftarrow$ **count** rhythmic transitions **of** R **in** R^*
10. $\mathcal{M} \leftarrow$ **count** melodic transitions **of** M **in** M^*
11. $\mathcal{A} \leftarrow$ **count** alignments **of** R, M **in** R^*, M^*
12. $S^{(R)} \leftarrow$ **calculate** rhythmic similarities **in** R
13. $S^{(M)} \leftarrow$ **calculate** melodic similarities **in** M
14. $\tilde{\mathcal{R}} \leftarrow$ **extrapolate** \mathcal{R} **using** $S^{(R)}$
15. $\tilde{\mathcal{M}} \leftarrow$ **extrapolate** \mathcal{M} **using** $S^{(M)}$
16. $\tilde{\mathcal{A}} \leftarrow$ **extrapolate** \mathcal{A} **using** $S^{(M)}, S^{(R)}$
17. $\tilde{\mathcal{R}}, \tilde{\mathcal{M}}, \tilde{\mathcal{A}} \leftarrow$ **thin** $\tilde{\mathcal{R}}, \tilde{\mathcal{M}}, \tilde{\mathcal{A}}$
18. **create new player with** $R, M, \tilde{\mathcal{R}}, \tilde{\mathcal{M}}, \tilde{\mathcal{A}}$

Figure 2.1: Simplified pseudo-code for creating new players.

and minor chords is enough to determine whether a given note is out of scale or not.

The transcriptions are formatted by pairs of keys and arguments, surrounded by round brackets, e.g. (**composer** Charlie Parker). The keywords are **title**, **composer**, **key** to identify the key of the solo, e.g. Bb, **choruses** followed by a number that indicates how many repetitions of the chorus are transcribed, **chords** and **notes**. The argument to **chords** is the chord sequence for this solo. Measures are separated by the pipe character, and each measure can contain one, two or four chords from the list of chords give in table 2.2 (along with a root for the chord) or the slash character that repeats the previous chord. The **notes** keyword expects an argument that lists all notes of the solo separated by white spaces. Each note has to match the following expression:

$$([a-g] [b\#]? | r) (\backslash+* | \backslash-*) (\backslash D (\backslash+D) *) ?$$

with $\backslash D$ set to

$$(1|2|4|8|16|32) (\backslash. | /3) ?$$

The first part, ($[a-g] [b\#]? | r$), identifies the note with optional accidental or a rest, each + or - in ($[\backslash+]* | [\backslash-]*$) increases or decreases the octave of the note by one (that is, adds 12 or -12 to the pitch), and the last part identifies the duration of the note, which has to start with a number, followed by an optional . to identify a dotted note or an /3 to identify a triplet, optionally followed by

an unlimited number of duration definitions joined by the + character which will be added to the notes duration. If no duration is specified, the duration will be set to one eighth. In the process of reading transcription files, the duration will be converted to an integer such that a breve has the value 96 (i.e. the shortest possible duration is 32/3 which has value 1) and the pitch will be converted to an integer such that A440 will have a value of 48.

Name	Artist	Measures
Anthropology	Charlie Parker	60
Bye Bye Blackbird	Ray Henderson	31
Byrd Like	Freddie Hubbard	107
Cheryl	Charlie Parker	58
Dewey Square	Charlie Parker	30
Groovin' High	Dizzy Gillespie	17
Laird Baird	Charlie Parker	36
Moment's Notice	John Coltrane	41
Moose The Mooche	Charlie Parker	30
Now's The Time ³	Charlie Parker	58
Now's The Time ³	Charlie Parker	90
On Green Dolphin Street	Unknown	57
Ornithology	Charlie Parker	31
Scrapple From the Apple	Charlie Parker	30
Yardbird Suite	Charlie Parker	31

Table 2.1: *Tunes in the set of transcriptions used by Impro-Visor, adding up to a total length of 707 measures.*

Transcription files can be read, parsed and transformed to a pattern inventory using the Python script

```
$ ./run.py create input_file1 [input_file2 ...]
```

With the `-r` argument, a profile can be specified (as defined in `profiles.conf`); if no profile is given the user will be queried with a list of available profiles. A profile is a set of parameters that override the standard parameter set which is specified as the `default-profile` in `profiles.conf`. With the optional argument `-n` a name can be set for the player that is about to be created, furthermore `-d` turns on the debug mode. This will run the algorithm that is sketched in figure 2.1 and save the player into the `players` folder, along with a `.stats` file which includes everything that was learned during creation and to which changes that are made during the training phase will be appended.

Then in step 3 of the creation algorithm, the conditional note function distribution $N : \mathbb{N} \rightarrow \mathbb{R}^4$ is calculated from the notes read in the previous step such that $N(t)$ is a four-tuple⁴ of probabilities for a note at onset t of a fragment being a **chord** tone, **colour** tone, **approach** tone or **arbitrary** tone (**x**) (cf. Keller (2007)):

$$N(t) = (\Pr(\mathbf{h}|t), \Pr(\mathbf{c}|t), \Pr(\mathbf{a}|t), \Pr(\mathbf{x}|t))$$

Thereafter, the data set will be fragmentised in step 4, and this sequence of musical fragments will be converted into a sequence of corresponding rhythmic and melodic fragments afterwards. Fragmentation simply means splitting the sequence of notes on the data set into parts such that the notes and rests in every part sum up to the same constant value. It is usually said that jazz

³The first version was played in F major and contains six choruses, the second version was played in G major over eight choruses.

⁴Although *rest* is another note function, it will not be covered by this probability distribution for reasons that will become evident in section 2.4.2.

Chord	Frequency	Σ
X^7	0.476	0.476
X^{m7}	0.241	0.717
X^{M7}	0.092	0.810
X	0.050	0.860
X^6	0.039	0.899
X^{7alt}	0.034	0.934
X^{o7}	0.016	0.950
X^m	0.012	0.963
X^{m7b5}	0.011	0.974
X^{7b9}	0.007	0.985
X^9	0.007	0.983
X^{m6}	0.003	0.996
X^{M6}	0.002	0.999
$X^{7\#9}$	0.001	1.000

Table 2.2: Distribution of chord extensions on the Impro-Visor data set, summed over all possible roots. The X is a placeholder for different chord bases.

musicians think about three seconds in advance when deciding what to play next; with a typical tempo of 160 beats per minute in bebop, that adds up to two measures. However for data analytical reasons all experiments described here were performed using fragments of half a measure’s length and a full measure’s length – we will see in the following sections why this is beneficial. Fragments of variable length (i.e. the data set being split by a criterion different to fragment length) are conceivable, too, and this point will be discussed later on.

Rhythmic representation and segmentation

As said in section 1.1, a melody is defined as a sequence of triples of pitch, volume and duration, and each fragment that is extracted from the data set is a melody in this sense such that all durations of this sequence sum up to a given length (with rests included as notes with the special pitch of -1). Simply put, a rhythmic fragment is a sequence of notes and rests without pitches. As the data set in use does not give any volume information of the notes a rhythmic fragment can be represented as a set of pairs of onset on duration (where 96 is the duration of a full measure).

Although a data set of some 800 tokens seems utterly undersized to represent all aspects of bebop improvisation, we can show that certain characteristics used by the system proposed are invariant to the size of the data set: the number of distinct tokens that are minimally required (i.e. the number of most frequent fragments) to cover a certain fraction of the data set only depends on the length of the token and the fraction to be covered, given the data set is sufficiently large. This can be shown by starting with a subset of the data set of length one and incrementally increasing the subset. For every step, it was calculated how many distinct fragments were required for covering the different fractions of that data set of this size. The results can be seen in figure 2.2: apparently the minimal number of fragments that cover specific fractions of the data set converges to a fixed point. These numbers will be used as references in initialising the system’s inventory: smaller numbers (i.e. less fragments) will yield more robust results, however larger numbers may yield aesthetically more interesting results.

Table 2.3 shows the twenty most frequent rhythmic fragments R for different fragment sizes on the full data set R^* which are selected in step 7 in algorithm 2.1.

Step 9 in algorithm 2.1 then estimates the transition probabilities from and

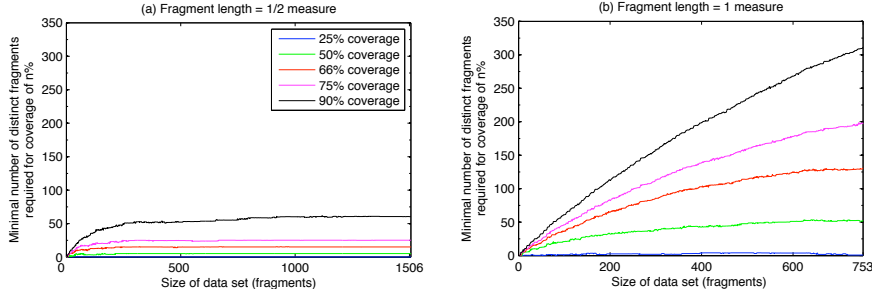


Figure 2.2: Number of most frequent rhythmic fragments required for covering different fractions of the (rhythmic) data set for different data set sizes and (a) fragments of half a measure and (b) fragments of a full measure’s length. This figure depicts the average over ten randomisations of the data set (ie. random permutations of the ordering of fragments), however the results for random permutations of tunes or no randomisation at all are almost the same.

to each of the fragments selected in step 7 such that for all $r_i, r_j \in R$:

$$\Pr(r_j|r_i) = \frac{C_{R^*}(r_i, r_j)}{C_{R^*}(r_i)}$$

where $C_{R^*}(r_i, r_j)$ is the number of occurrences of the sequence (r_i, r_j) in the rhythmic data set R^* . These transition probabilities are stored in a transition matrix \mathcal{R} such that $\mathcal{R}_{i,j} = \Pr(r_j, r_i)$.

According to some similarity measure σ (which will be described in section 2.3.2), a similarity matrix $S^{(R)}$ will be computed in step 12 of the creation algorithm such that $S_{i,j}^{(R)} = \sigma(r_i, r_j)$. This similarity matrix is used to extrapolate unseen rhythmic transitions in step 14 using the nearest neighbour heuristic and the *rhythmic extrapolation factor* $\kappa_r \in [0, 1]$ that controls the weight of extrapolated values compared to actually seen transitions. This is done by setting any unseen transitions to the probability of the transition with the most similar conditional multiplied by the similarity of the two conditionals and κ_r :

$$\tilde{\mathcal{R}}_{i,j} = \begin{cases} \kappa_r \cdot \mathcal{R}_{k,j} \cdot S_{k,i}^{(R)} & \text{such that } k = \arg \max_k S_{k,i}^{(R)} \text{ if } \mathcal{R}_{i,j} = 0 \\ \mathcal{R}_{i,j} & \text{else} \end{cases} \quad (2.3.1)$$

Other ways of extrapolating missing values are certainly possible and should be investigated. After performing equation 2.3.1 in step 14 each row has to be normalised again so as to add up to one:

$$\tilde{\mathcal{R}}_{i,j} \leftarrow \frac{\tilde{\mathcal{R}}_{i,j}}{\sum_{k=1}^{|R|} \tilde{\mathcal{R}}_{i,k}}$$

Finally, the extrapolated rhythmic transitions will be thinned in step 17 by a rhythmic thinning factor δ_r such that $0 \leq \delta_r \leq 1$, which means that a fraction δ_r of each transitions probability will be evenly distributed over all transition probabilities with the same conditional.

$$\tilde{\mathcal{R}}_{i,j} \leftarrow (1 - \delta_r)\tilde{\mathcal{R}}_{i,j} + \frac{1}{|R|} \sum_{k=1}^{|R|} \delta_r \tilde{\mathcal{R}}_{i,k}$$

As each row in $\tilde{\mathcal{R}}$ adds up to one this simplifies to

$$\tilde{\mathcal{R}}_{i,j} \leftarrow (1 - \delta_r)\tilde{\mathcal{R}}_{i,j} + \frac{\delta_r}{|R|} \quad (2.3.2)$$

Thinning is an important mechanism for the performance task 2.2.2 as it regulates the “importance” of e.g. rhythmic transitions over melodic transitions – the more homogeneous one of the factors becomes, the less influence it will have on the final conditional probabilities from which a next pair of rhythmic and melodic fragments will be selected, up to the extreme of $\delta_r = 1$ at which a matrix will be entirely homogeneous after thinning and thus have no impact on the decision process. Thinning factors for rhythmic and melodic transition matrices as well as the alignment matrix can be set in `profiles.conf`.

Melodic representation and segmentation

Section 2.3.1 described the representation of and calculations done with rhythmic fragments. Although steps 7, 9, 14 and 17 of the creation algorithm 2.1 can be directly applied to melodic fragments as well (which is done in steps 8, 10, 15 and 17, respectively), it is not immediately apparent how to represent melodic fragments.

As opposed to rhythmic fractions, where we have seen that a fairly small number of distinct fragments accounts for most of the rhythms seen on the data set, considerable abstraction from the surface form of melodies is necessary to produce similar effects for melodic fragments. Simply encoding the absolute pitch of each note (whilst ignoring onset and duration as these are encoded in

Rank	Rhythm	f	$\sum_r f$	Rank	Rhythm	f	$\sum_r f$
(1)		0.292	0.292	(1)		0.141	0.141
(2)		0.162	0.454	(2)		0.089	0.230
(3)		0.031	0.485	(3)		0.015	0.244
(4)		0.029	0.513	(4)		0.015	0.259
(5)		0.028	0.542	(5)		0.009	0.268
(6)		0.025	0.566	(6)		0.009	0.278
(7)		0.023	0.590	(7)		0.009	0.287
(8)		0.020	0.610	(8)		0.009	0.296
(9)		0.019	0.629	(9)		0.009	0.305
(10)		0.019	0.648	(10)		0.008	0.313
(11)		0.015	0.664	(11)		0.008	0.321
(12)		0.015	0.678	(12)		0.008	0.329
(13)		0.013	0.692	(13)		0.008	0.337
(14)		0.013	0.704	(14)		0.008	0.345
(15)		0.013	0.717	(15)		0.008	0.353
(16)		0.010	0.727	(16)		0.008	0.361
(17)		0.009	0.737	(17)		0.007	0.368
(18)		0.009	0.746	(18)		0.007	0.375
(19)		0.009	0.755	(19)		0.007	0.381
(20)		0.009	0.765	(20)		0.007	0.388

Table 2.3: The twenty most common rhythmic fragments for (a) a fragment size of half a measure and (b) a full measure, their frequency on the data set (occurrences over size of the data set) and the coverage (accumulated frequencies over all fragments of equal or lower rank). In the former case, 172 distinct fragments can be found on the data set, and it is evident that only about ten percent of the fragments account for two thirds of the whole data set. In the latter case, there are 394 distinct fragments.


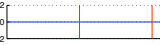
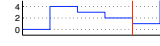
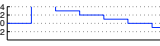








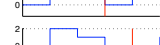
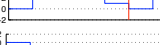

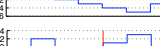
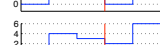

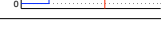

Rank	Melody	f	$\sum_r f$	Rank	Melody	f	$\sum_r f$
(1)		0.121	0.121	(1)		0.070	0.070
(2)		0.044	0.165	(2)		0.013	0.083
(3)		0.016	0.182	(3)		0.010	0.093
(4)		0.013	0.195	(4)		0.010	0.104
(5)		0.012	0.207	(5)		0.010	0.115
(6)		0.009	0.216	(6)		0.008	0.123
(7)		0.009	0.226	(7)		0.008	0.132
(8)		0.009	0.235	(8)		0.008	0.140
(9)		0.008	0.244	(9)		0.008	0.148
(10)		0.007	0.251	(10)		0.006	0.155

Table 2.4: The ten most frequent melodic fragments for a pattern length of (a) half a measure and (b) a full measure. The blue line shows the accumulated intervals, the red lines indicate where the melodic fragment ends, from which point on it will be repeated. For instance, the fragment underlying plot (a) (5) has the interval sequence (+4, -1, -1, -2). Melodies with less than three different pitches have been ignored, except for the fragment representing a rest.

rhythmic fragments, and furthermore volume, as no information about note velocity is given in the data set), we would only seldom find two identical fragments on the data set. Hence abstractions in the form of prior knowledge, contextual dependencies and heuristics have to be brought in to reduce the amount of information each melodic fragment carries. Firstly, the melody shall be invariant to the harmony with which it occurred, which is done by simply subtracting the pitch of the chord base modulo twelve from the pitch of each note, e.g. an A of pitch 48 played on a F major chord (which has a base pitch of eight) would yield 40, whilst the same note played over a D major would yield 37. Other techniques include storing the interval to the previous note instead of the relative pitch to the chord along with the function of the note (i.e. chord tone or scale tone) as done by Keller and Morrison (2007). Figure 2.3 illustrates the results of performing the same experiment as performed for figure 2.2 using different techniques. The results were used to compare different options for the representation of melodies: in the first three rows, different binnings of intervals to the previous notes were used; in the lower three rows the function of each note was additionally stored. However the quality of a representation not only depends on efficient encoding of melodies, but also on the quality of the reconstruction (which to some extent is a very subjective measure). There is not always an obvious way to resolve ambiguities, and even if ambiguities can be resolved, to little information content of melodic fragments means that major parts of the information required to reconstruct melodies resides in contextual and algorithmic factors rather than in the melodic fragments that are extracted from transcriptions of real solos.

Ultimately the representation in use stores melodic fragments as sequences of pairs of binned intervals and note functions. The size of the bins was statistically determined by analysis of the distribution of note intervals as depicted in figure 2.4. By separating the intervals from functions (which will be statistically assigned to notes), it was possible to cover a vast part of the data set with a limited number of fragments whilst achieving a high quality of the reconstruction of melodies. A visualisation of the ten most frequent fragments on the data

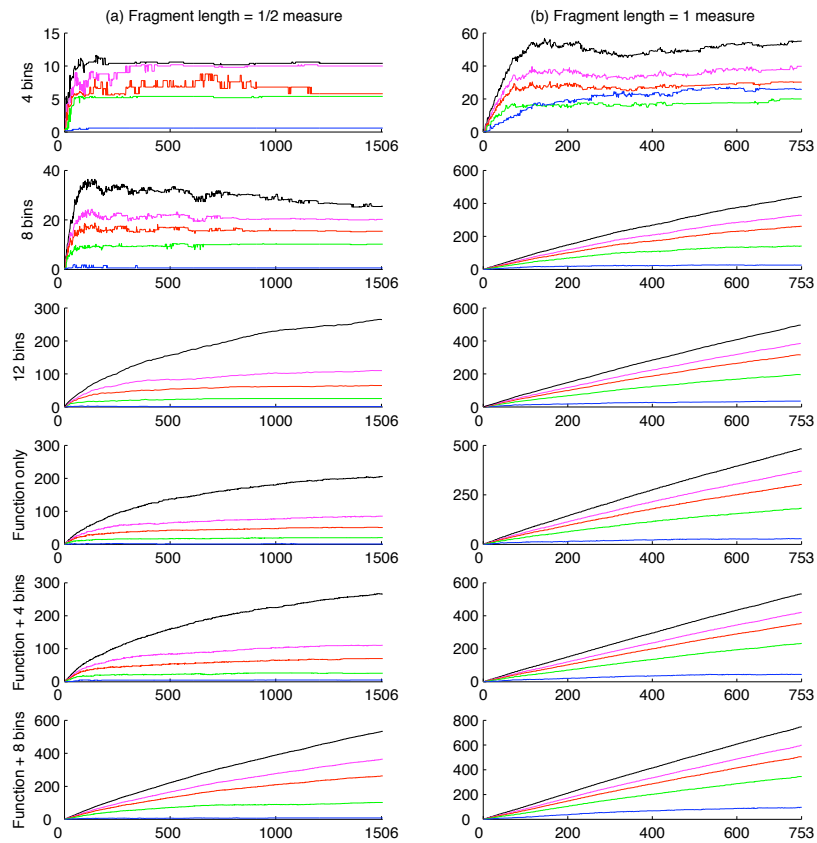


Figure 2.3: Analogously to figure 2.2, this figure depicts the minimal number of distinct fragments that is required to cover a given ratio of the data set for variable sizes of the data set; averaged over ten randomisations of fragments on the data set.

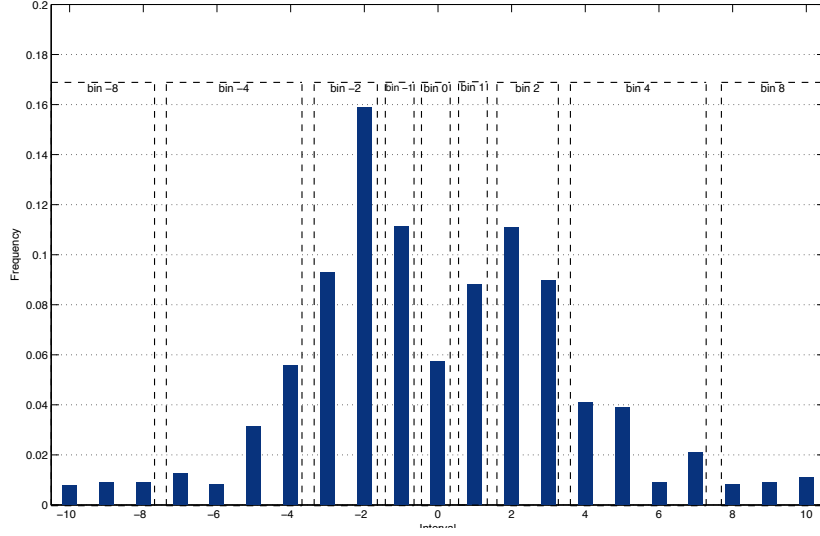


Figure 2.4: Distribution of note intervals on the data set and placement into bins. The outer bins include all other intervals not depicted in this plot.

set is shown in table 2.4.

Analogously to equation 2.3.1, melodic similarities are extrapolated by calculating

$$\tilde{\mathcal{M}}_{i,j} = \begin{cases} \gamma_m \cdot \mathcal{M}_{k,j} \cdot S_{k,i}^{(M)} & \text{such that } k = \arg \max_k S_{k,i}^{(M)} \text{ if } \mathcal{M}_{i,j} = 0 \\ \mathcal{M}_{i,j} & \text{else} \end{cases}$$

and afterwards normalised such that

$$\tilde{\mathcal{M}}_{i,j} \leftarrow \frac{\tilde{\mathcal{M}}_{i,j}}{\sum_{k=1}^{|M|} \tilde{\mathcal{M}}_{i,k}}$$

Along with the rhythmic transitions and alignments the extrapolated melodic transition matrix will be thinned in step 17 according to

$$\tilde{\mathcal{M}}_{i,j} \leftarrow (1 - \delta_m) \tilde{\mathcal{M}}_{i,j} + \frac{\delta_m}{|M|} \quad (2.3.3)$$

Calculating alignments

After a set of rhythmic and melodic fragments has been selected and transition probabilities within these sets have been computed and extrapolated, the joint probabilities of a specific rhythmic fragment in R occurring together with a specific melodic fragment in M have to be computed in step 11 of algorithm 2.1. This is done by performing calculation 2.3.4 on each pair of elements $(r_i, m_p) \in R \times M$.

$$\Pr(r_i, m_p) = \frac{C_{R^*}(r_i, m_p)}{\sum_{r \in R} \sum_{m \in M} C_{R^*}(r, m)} \quad (2.3.4)$$

Afterwards, these probabilities are written into matrix \mathcal{A} such that $\mathcal{A}_{i,j} = \Pr r_i, m_j$, and unseen alignments are extrapolated using the nearest neighbour estimation in similarity space:

$$\tilde{\mathcal{A}}_{i,j} = \begin{cases} \mathcal{A}_{k,l} \cdot \gamma_a \cdot S_{k,i}^{(R)} \cdot S_{l,j}^{(M)} & \text{such that } (k,l) = \arg \max_{k,l} (S_{k,i}^{(R)} S_{l,j}^{(M)}) \text{ if } \mathcal{A}_{i,j} = 0 \\ \mathcal{A}_{i,j} & \text{else} \end{cases}$$

where γ_a is the *alignment extrapolation factor* as defined in `profiles.conf`. The formula for thinning the alignment matrix is slightly different to those for the rhythmic and melodic transition matrices (eq. 2.3.2 and 2.3.3, respectively) as the probability mass of an individual joint probability has to be distributed over the entire matrix instead of just the same row:

$$\tilde{\mathcal{A}}_{i,j} \leftarrow (1 - \delta_a)\tilde{\mathcal{A}}_{i,j} + \frac{\delta_a}{|R| \cdot |M|} \quad (2.3.5)$$

2.3.2 Rhythmic similarity measures

Steps 12 and 13 of the algorithm sketched in figure 2.1 require a formalised notion of similarity between rhythmic and melodic fragments. The trouble is that rhythmic similarity as a rather vague concept as human judgement about the (dis-)similarity of two rhythms may depend on various different and probably unknown factors. Numerous similarity measures have been developed as a response to this problem over the past two decades, however most of them have been designed with rather specific purposes in mind, e.g. automatic retrieval of music from databases. When evaluating different metrics for this framework, two aspects should be considered: instead of comparing percussive rhythms (rhythms that are easily associated with a certain style or genre) we will use the note onsets and durations in single measures of real improvisations. Furthermore, our (informal) definition of rhythmic similarity shall be: given a sequence of fragments with underlying rhythmic fragments (r_i, r_j, r_k) that sounds rhythmically coherent, a rhythmic fragment r'_j will be rhythmically similar to r_j if this rhythmic coherence will be maintained by replacing rhythmic fragment r_j by fragment r'_j .

Existing approaches

A natural approach to formalising rhythmic similarity is using an edit distance, where the similarity is the reciprocal of some measure of work required to transform one rhythm into another given a set of operators. The possibly simplest of such edit distance metrics is the Hamming Distance (see Hamming (1986)). For this metric, rhythms are represented as bit-strings where 1 denotes a note onset and 0 denotes a hold or rest. The distance of two rhythms is then the number of different bits:

$$D_{Hamming}(A, B) = \sum_{k=1}^n |a_k - b_k|$$

where a_k denotes the value of the k -th character of A and n the length of strings A and B . This approach has a number of obvious shortcomings: firstly, small movements of note onsets have the same effect on the distance as long movements. Secondly we can assume that the presence or absence of notes on some beats is more important for judging the similarity of rhythms than on others; for example adding a note on the second beat is probably less significant than deleting a note on the first beat.

Bookstein et al. (2001) extend the Hamming Distance to what they call a *fuzzy* Hamming Distance by introducing a shift-operator to the two operators that are already implicitly used by the naïve Hamming distance (namely insertion and deletion) that allows to shift bits at lower cost than inserting and deleting them to account for neighbourhood structures within the strings. Bookstein et al. use a dynamic programming method to compute the fuzzy Hamming distance in polynomial time.

Toussaint (2003) suggests a simplified version of the fuzzy Hamming distance which they call the *swap* distance. The distance of two bit-strings is then defined as the number of swaps of adjacent bits that have to be performed in order to

transform one bit-string into another. If we represent a bit-string as a vector of indices at which a 1 occurs in the string, ie. $A = 0110$ as $\vec{v}_A = (2, 3)$, this distance can be computed in linear time:

$$D_{Swap}(A, B) = \sum_{k=1}^n \left| \vec{v}_A^{(k)} - \vec{v}_B^{(k)} \right|$$

Although we have now means to account for the neighbourhood of onsets, we still can't express the importance of different beats. Furthermore Toussaint does not specify what happens if one pattern contains more onsets than the other; setting n to the length of the shorter vector implies that insertions / deletions have the same cost as one swap.

A more general form of the swap distance is the *earth mover's distance* (EMD) as used by Typke et al. (2003) for melodic similarities (see section 2.3.3). This distance can be seen as the *minimal* amount of work required to move earth from multiple hills of various size to holes of various size. If the total size of the holes (called "demand points") equals the total size of hills ("supply points") this distance measure is also known as the *proportional transportation distance* (and in fact only then it is a true metric as this constraint is required to maintain the triangle inequality). Formally, the EMD is defined over the weighted set of demand points \mathcal{D} , the weighted sets of supply points \mathcal{S} and a ground distance measure δ where D_i denotes the weight of the i -th element of \mathcal{D} and $|\mathcal{D}|$ the number of elements in \mathcal{D} , by a set of operations $\mathcal{F} = [f_{ij}]$ that moves some amount of mass from \mathcal{D} to \mathcal{S} ,

$$D_{EMD}(\mathcal{D}, \mathcal{S}) = \min_{\mathcal{F} \subseteq \mathcal{F}} \frac{\sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{D}} f_{ij} \delta_{ij}}{\sum_{j \in \mathcal{D}} D_j}$$

subject to the following constraints:

1. $f_{ij} \geq 0, i \in \mathcal{S}, j \in \mathcal{D}$
2. $\sum_{i \in \mathcal{S}} f_{ij} = S_i, j \in \mathcal{D}$
3. $\sum_{j \in \mathcal{D}} f_{ij} \leq D_i, i \in \mathcal{S}$

See Rubner et al. (1998) for more details. The EMD can be adopted to measuring rhythmic similarity by defining the note onsets of one pattern as supply points, where the position of the note onset in the measure is translated to the position of the supply point in space and the importance of that beat determines the weight of the supply points. The same applies for the other pattern which becomes the demand point. Of course the "importance of different beats" is somewhat vague and hard to define. Typically, the duration of the notes is used as the weight. However, under the assumption that with optimal weights of different beats the EMD could indeed resemble human judgement about similarity (or dissimilarity) of rhythms, these weights could be learned automatically if a sufficiently large training set with human rated similarities was given. Typke et al. (2003) stated the metric as a linear programming problem for melodic similarities; yet the distance can be easily calculated in linear time by reducing the the problem to 1-dimensional space (which is the time).

Converting from a measure of distance to similarity requires a correspondence function $h : \mathbb{R}_0^+ \rightarrow [0, 1]$ such that h is monotonic decreasing and $h(0) = 1$. There are a number of options to select from in `profiles.conf`, however the default is

$$h(x) = \frac{1}{\sqrt{x+1}}$$

Evaluation of approaches

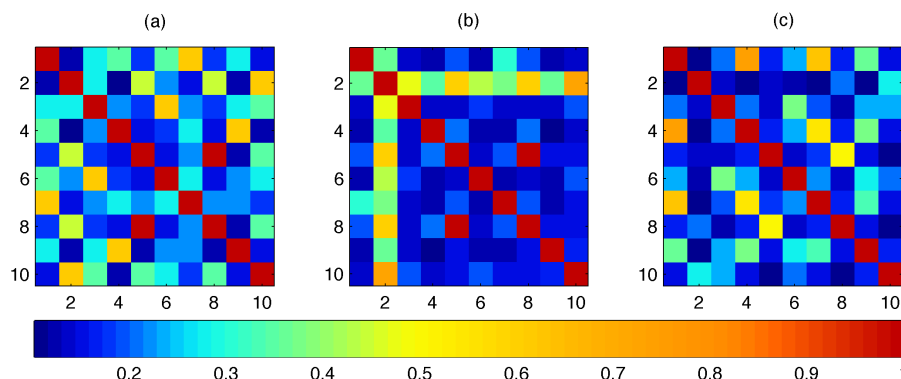


Figure 2.5: Similarity matrices of the first ten fragments shown in table 2.3 (b), using (a) the Hamming distance, (b) the swap distance and (c) the earth mover's distance with weights according to note durations. The colour of each square marks the similarity of the fragments whose ranks corresponds to the row and column of the square, according to the scale provided by the figure.

All of the suggested distance measures are implemented in `similarity.py` and can be used by changing the *rhythmic similarity* parameter in `profiles.conf`. A visualisation of the effect of different metrics is shown in figure 2.5. Determining which of the metrics is 'best' is not a trivial task, and at this point we cannot provide any hard evidence of one of the metrics being superior to any other. One way of producing such evidence would be to compare the judgements of these metrics with those of human beings, however lacking such a gold standard we are forced to compare the metrics by qualitative measures. One important aesthetic criterion is the preservation of rests at the beginning and the end of fragments, and we can observe that e.g. fragments (3), (6), (9) and (19) are assigned a high similarity towards each other using D_{Swap} and D_{EMD} , but have comparably lower similarity using $D_{Hamming}$. Clearly, more research is required to investigate the effects of different similarity metrics. Furthermore, the quality of the extrapolation using similarity measures has to be evaluated against methods that are 'dumb' with respect to the content of the data e.g. smoothing techniques for Markov processes such as additive smoothing or Good-Turing smoothing (see Chen and Goodman (1996) for a comprehensive comparison).

2.3.3 Melodic similarity measures

Existing approaches

The problem that most research on musical similarity measure was done with tasks such as retrieval from databases in mind is even more evident in melodic similarity measures compared to rhythmic similarity measures, even more so as only few classical distance metrics can be directly applied to melodies.

In the simplest case, we merely compute the absolute difference vector of our interval representation of melodies:

$$D_{Int}(\vec{m}_1, \vec{m}_2) = \sum_k |m_1^{(k)} - m_2^{(k)}|$$

However, this does not account for the pitches that will actually be perceived when the melody is performed. A straight forward way of measuring the melodic similarity based on actual pitches is the geometric distance as proposed by

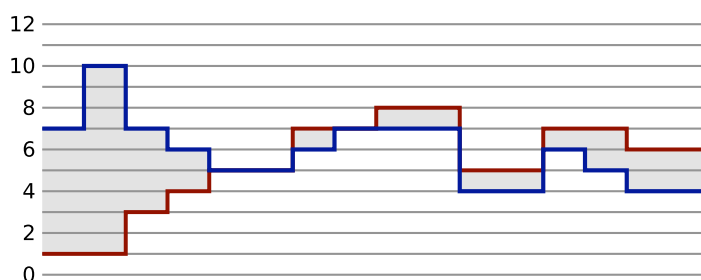


Figure 2.6: A depiction of the geometric distance as defined in equation 2.3.6 - the distance is the integral over the grey area; in this case 33. The minimal distance of 31 is achieved by transposing the red melody by -1 .

Ó Maidín (1998) which requires the melodies to be monotonic rectilinear functions as depicted in figure 2.6. The distance is then measured as the area between two polygonal melody chains; with the nice property that it also works in the continuous case:

$$D_{Geo}(m_1, m_2) = \int_0^T |m_1(t) - m_2(t)| dt \quad (2.3.6)$$

where m_1, m_2 are melodies as functions of pitch over time (transposed by the Chord with which they were originally found) and T is the duration of a pattern. In the discrete case we take the sum instead of the integral.

As the melodies will be transposed by the base note of the chord with which they will be played, it seems feasible to make the distance invariant to transposition. Francu and Nevill-Manning (2000) extend this approach to by defining the distance as the minimal distance over all transpositions of one melody:

$$D_{MinGeo}(m_1, m_2) = \min_{\Delta} \int_0^T |m_1(t) - m_2(t) + \Delta| dt$$

Evaluation of approaches

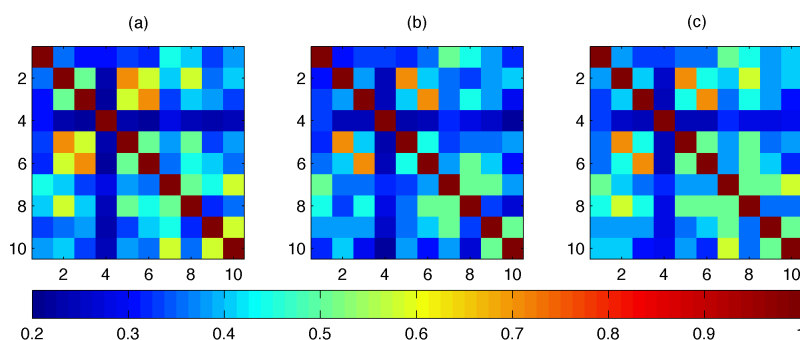


Figure 2.7: The similarities calculated between the melodic fragments of table 2.4 (a) using (a) the interval distance, (b) the geometric distance and (c) the minimal geometric distance.

As opposed to the different rhythmic similarity measures, the applications of whom have been shown in figure 2.5, the application of different melodic similarity measures on the melodic fragments of table 2.4 bears only marginal differences, as evident by inspection of figure 2.7, and there is no evidence that the models described above are cognitively adequate with respect to the

similarity they assign to melodic fragments. The search for an adequate melodic similarity measures for this application partly suffers from the representation of melodic fragments as time-invariant intervals between pitches, which exclude metrics that require absolute note pitches, e.g. the work done by Hofmann-Engl (2001) using the notion of melodic chains. Considerably different results might be obtained by contourisation of melodies, which based on the assumption that the extrema of pitches are more important to human listeners than the pitches in between.

Undeniably much more research on the effectiveness and cognitive adequacy of different rhythmic and melodic similarity measures is required for this application. Nevertheless the metrics discussed offer a good starting point and can straightforwardly applied to extrapolate the transition matrices in steps 14 to 16 of algorithm 2.1. Exemplary results of this extrapolation and the subsequent thinning are depicted in figure 2.8.

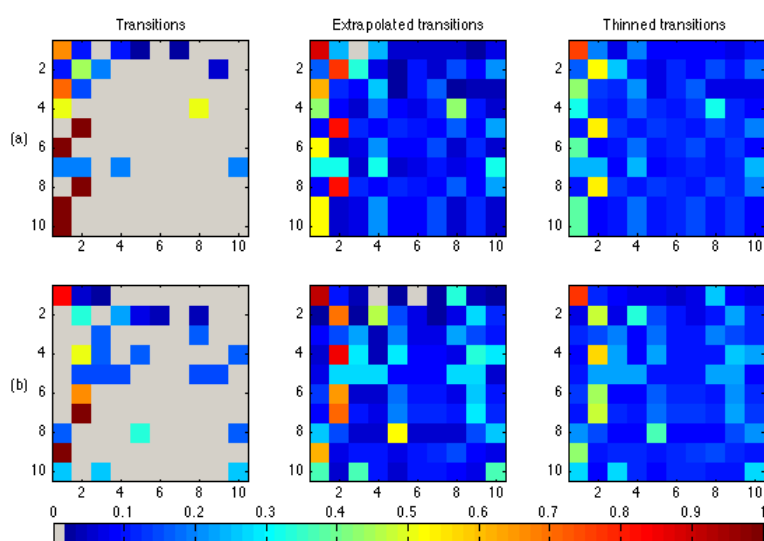


Figure 2.8: Effects of applying (a) the EMD-distance to the rhythmic transition matrix using the nearest-neighbour extrapolation and afterwards thinning the matrix by factor 0.4 on the first ten rhythmic fragments of table 2.3 (b) and (b) applying the geometric distance to the melodic transition matrix to generate the extrapolated and thinned melodic transition matrix over the fragments shown in table 2.4 (a). Grey values indicate missing transitions.

2.4 Training phase

In the training phase, a solo over a given lead sheet will be created and presented to the user, who can then give temporally aligned feedback to the system. The training procedure can be started by calling

```
./run.py train
```

with the optional arguments `-p player` to specify a player file and `-r profile` to specify a profile with training parameters from `profiles.conf`. If these arguments are missing, the user will be prompted to select a player and a training profile from a list of available items. A list of all tunes known to the player will be displayed, and after the user has selected a tune the training algorithm sketched

1. **require** tune with lead sheet L and tune layer $\hat{\mathcal{M}}, \hat{\mathcal{R}}, \hat{\mathcal{A}}$
2. $r(0), m(0) \leftarrow$ empty fragments
3. **for** $t \in \{1, \dots, |L|\}$:
4. $r(t), m(t) \leftarrow$ **select** successor fragments **using** $r(t-1), m(t-1)$
5. $S(t) \leftarrow (r(t), m(t))$
6. $S'(t) \leftarrow$ **join** $r(t), m(t)$
7. $S'(t) \leftarrow$ **apply** note-level rules **to** $S'(t)$
8. **for** $t \in 1, \dots, |L|$:
9. $F(t) \leftarrow$ **play** $S'(t)$ and receive feedback
10. **update** $\tilde{\mathcal{M}}, \tilde{\mathcal{R}}, \tilde{\mathcal{A}}$ **with** F
11. **update** $\hat{\mathcal{M}}, \hat{\mathcal{R}}, \hat{\mathcal{A}}$ **with** F

Figure 2.9: Pseudo-code of the training algorithm.

in figure 2.9 will be executed; further details of this algorithm can be found in the subsequent sections. The user can give positive or negative feedback⁵ to the performance by pressing the + and - keys on his keyboard, and interrupt the performance by pressing the q key (any reward already given will be distributed nevertheless). After the performance is ended, the user can opt to save the changes made to the player and a log of all changes will be appended to the player's `.stats`-file. A transcription of the performance will be saved into the `performances` folder, either in the internally used format or optionally in the format used by *Impro-Visor*.

2.4.1 Finding walks through pattern space

The lead sheet L required for step 1 of the training algorithm 2.1 includes a sequence of chords, and a so-called tune layer with rhythmic and melodic transition and alignment matrices, $\hat{\mathcal{R}}, \hat{\mathcal{M}}$ and $\hat{\mathcal{A}}$, respectively. When the player is newly created, these matrices will be entirely homogeneous, i.e. all values in $\hat{\mathcal{R}}$ and $\hat{\mathcal{M}}$ will be $1/|R|$ and $1/|M|$ respectively and all values in $\hat{\mathcal{A}}$ will be $1/(|R| \cdot |M|)$, where R and M are the player's sets of rhythmic and melodic fragments. The length of the tune $|L|$ is calculated depending on the length of the fragments used, and hence for one performance of the tune without repetitions, $|L|$ musical fragments have to be generated and joined to produce playable notes. This is done by iteratively solving the layered performance task specified by equation 2.2.2, starting with the fragments that represent the rhythmic and melodic rests (which, technically, are played before the first fragment of the actual solo). In algebraic terms, this means calculating

$$Q^{(i,p)} = (\tilde{\mathcal{R}}_i^T \tilde{\mathcal{M}}_p) \bullet \tilde{\mathcal{M}} \bullet (\hat{\mathcal{R}}_i^T \hat{\mathcal{M}}_p) \bullet \hat{\mathcal{M}}$$

where i denotes the index of the last rhythmic fragment, j denotes the index of the last melodic fragment and $A \bullet B$ denotes the Hadamard product of matrices A and B . The resulting matrix $Q^{(i,p)}$ can then be seen as a probability distribution where $Q_{j,q}^{(i,p)}$ is the probability that fragments r_j and m_q will be played next given that r_i and m_p have been played before. Let $\Psi : (\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{R}^+) \rightarrow \mathbb{N} \times \mathbb{N}$ be an

⁵Propitiously, the computer lacks any sign of ego and thus takes criticism rather well.

operator that will select a pair of indices from such a matrix according to this probability distribution. Hence step 5 is performed by computing $S : \mathbb{N} \rightarrow \mathbb{N} \times \mathbb{N}$:

$$S(t) = \Psi(Q^{S(t-1)}).$$

2.4.2 Applying note-level rules

Furthermore, $S'(t)$ is computed in step 6 of algorithm 2.9 by joining the rhythmic fragments with the indices of fragments provided by $S(t)$. To do so, a note is created for every beat in the rhythmic fragment, and the i -th non-rest note will be assigned the pitch of the $(i - 1)$ -th note of the fragment plus the i -th interval of the melodic fragments, where the $(0 - 1)$ -th note corresponds to the last note⁶ of the fragment joined in $S'(t - 1)$. If there are more notes in the rhythmic fragment than intervals in the melodic fragment, the melodic fragment will be repeated from start. Surely there are other ways of ‘stretching’ melodic fragments, however this method proved to yield interesting and coherent musical results. A special case occurs if the pitch assigned to a note exceeds the range of the instrument defined in `profiles.conf`; in this case the interval is inverted.

In the next step, the pitches of the raw notes will be adjusted according to the conditional note functions learned in step 3 of the creation algorithm; first a function (*chord tone*, *colour tone*, *approach tone* or *arbitrary tone*) will be assigned to any non-rest note depending on the probability distribution of functions $N(t)$ calculated in step 3 of the creation algorithm, where t is the onset of the note. If a chord or colour tone is selected, the note pitch will be shifted in the direction of the interval from the last note (i.e. in the direction of the melody) until the note pitch actually resembles this function. If the note becomes an approach tone and the next note is a chord tone, the pitch will be set to be the pitch of the next note plus one if the interval from the previous note is negative (i.e. the melody is descending) or minus one, else.

Further note-level enhancements are conceivable, i.e. adjustments in micro-timing and accentuation as done by Widmer (2001).

2.4.3 Learning from human feedback

Once $S'(t)$ is computed for all $t \in \{1, \dots, |L|\}$, the notes associated with each of the fragments will be converted to MIDI. If the *swing* parameter in `profiles.conf` is set to true, two consecutive quavers will be played ‘swingly’, i.e. will be converted to a triplet of a crotchet and a quaver. If the corresponding options are activated, the solo will also automatically be accompanied by percussion and a piano, which makes it easier for the listener to follow and judge the improvisation. Again, different styles for this accompaniment can be defined in `profiles.conf`. Other parameters taken into account when converting from notes to MIDI is the *tempo* in beats per minute (BPM) and the MIDI patches of the solo and the piano instrument.

Once the entire MIDI file for one solo is created, it will be played using the `pygame.mixer` module and the user can provide feedback by pressing the + and - keys, as shown in figure 2.10. This feedback is stored in a function $F(t)$ that maps the time the feedback was given (measured in fragments) to the number of times the + key was pressed minus the number of times the - key was pressed. The *offset* parameter δ_t controls whether the feedback will be aligned to the fragment which was played at the time the feedback was given ($\delta_t = 0$) or some previous fragment ($\delta_t > 0$). This allows the listener some time to decide whether a phrase played was actually good or not; the value of δ_t should be adjusted to the listeners preference and to the length of fragments. For each

⁶Currently, the last pitch of the initial fragment $S'(0)$ is defined to be a random number within the range of the instrument

fragment played, the feedback reinforces or punishes the transitions from the previous rhythmic and melodic fragments and the alignment between them in the player's master layer and the tune layer, and the same applies to a number of fragments played immediately before, albeit with a discounted feedback.

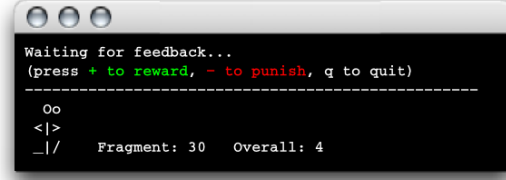


Figure 2.10: The user interface that queries the user for feedback while a freshly generated solo is being performed.

The *discount steps* parameter $\tau \in \mathbb{N}_0^+$ defines how long the reward temporally extends, i.e. over how many previous fragments the reward will be distributed and the *discount factor* parameter $\gamma \in]0, 1]$ describes how much of the reward will be ‘lost’ over time. The latter comes in three fashions, for rhythmic and melodic transitions and for alignments, which will be denoted by a subscript r , m and a respectively. The parameters ρ_p and $\rho_T \in \mathbb{R}^+$ allow for different learning rates for the player's layer and the tune layer. Finally the *reward factors* ε_r , ε_m and $\varepsilon_a \in \mathbb{R}^+$ are used to control the individual learning rates of the three aforementioned probability distributions. The update rule for the transition matrices is then $\forall t, s$ such that $0 \leq t \leq |L|$ and $0 \leq s \leq \tau$:

$$\begin{aligned}
 \tilde{\mathcal{R}}_{r_i, r_j} &\leftarrow \tilde{\mathcal{R}}_{r_i, r_j} \cdot \exp(\rho_p \varepsilon_r \gamma_r^s F(t - \delta_t)) \\
 \tilde{\mathcal{M}}_{m_i, m_j} &\leftarrow \tilde{\mathcal{M}}_{m_i, m_j} \cdot \exp(\rho_p \varepsilon_m \gamma_m^s F(t - \delta_t)) \\
 \tilde{\mathcal{A}}_{r_j, m_j} &\leftarrow \tilde{\mathcal{A}}_{r_j, m_j} \cdot \exp(\rho_p \varepsilon_a \gamma_a^s F(t - \delta_t))
 \end{aligned} \tag{2.4.1}$$

where $(r_j, m_j) = S(t - \delta_t - s)$ and $(r_i, m_i) = S(t - \delta_t - s - 1)$. Afterwards the modified matrices are thinned according to equation 2.3.2 and 2.3.5 respectively. The same operation is done for the tune layer in step 11 with $\hat{\mathcal{R}}$, $\hat{\mathcal{M}}$ and $\hat{\mathcal{A}}$ which ρ_T instead of ρ_p .

CHAPTER 3

Discussion and Evaluation

In the following sections I will evaluate the performance of the implemented system according to the phenomenology described in section 1.2. It is not my intention to endeavour in an evaluation of the aesthetic qualities of the system¹, yet I will comment on certain characteristics that could be ameliorated in section 3.4. An exemplary output of the system after training is given in figure 3.1; a MIDI version of this performance should be found in the attachments to this thesis.

3.1 Creativity in general

To recapitulate, the more general phenomena of creativity as described in section 1.2.1 were non-determinism, the absence of well-defined goals and the combination of associative elements. At least for the former the question of as to how far the proposed system effectuates these phenomena is fairly simple to answer. As solving the layered performance task given in equation 2.2.2 includes a Markovian decision process (which we have interpreted as an associative process), the creative output of the system is intrinsically non-deterministic. Less clear is what a well-defined goal is in the case of a decision process such as ours; i.e. whether solving task 2.2.2 is a well-defined goal state by its own. Then again, “playing” as such would be a goal. In the sense that there is no pre-defined set of goal states to the performance task, it seems as if the systems satisfies this trait of creativity. The latter of the three aspects, the associativity and combination of elements, is exactly what this system is build on.

3.2 Learning behaviour

Features of the learning behaviour of human jazz players are that the learning process starts with small variations of known tunes, that learning is possible by both listening and playing and that some rules can be made explicit and taught by teachers whilst others can not. Currently the system can not literally *listen* to performances by others, however it can learn from others by analysing transcriptions of performances instead. It is noteworthy that a good deal of information, especially information about accentuation and micro-timing, is lost in the transcriptions and can therefore not be learned; nonetheless the learning algorithm tries to exploit existing performances as much as possible.

Learning by playing is realised through human feedback as described in section 2.4.3. Here another limitation of the system becomes apparent: it is assumed that the virtual player has already acquired technical expertise on his instrument and can play whatever he wants to play; in fact it doesn't even

¹Partly because, with the notable exception of Keller et al. (2006), none of the existing approaches discussed in section 1.2.2 supplies any implementation that can be used for creating comparable results.

Anthropology smallbird

The musical score for "Anthropology" is presented in a single system with eight staves. The key signature is two flats (B-flat major), and the time signature is 4/4. The score includes a variety of musical notations, including eighth notes, quarter notes, and triplets. Chord changes are indicated above the staff, and the piece concludes with a final chord of B-flat.

Figure 3.1: The first chorus of a performance of “Anthropology” by a player created with the *smallbird*-profile after nine ‘rehearsals’ of the tune.

matter *which* instrument he wants to play as different instruments are merely realised via different MIDI mappings. In reality though, a lot of what musicians play when performing depends on the instrument; some licks are preferred over others because there are more intuitive fingerings to them, and furthermore musicians can *imagine* playing something that they can’t actually play – be it because they lack the technical skill or because their imagination is not explicit enough to be put into notes. Thus part of the feedback used when learning by playing is how close one gets to what one *wants* to play.

Be that as it may, incorporating this kind of learning would require fundamentally new learning paradigms, and to our knowledge there is currently no approach that distinctively models what can be performed and what should be performed whatsoever. The lack of such a model also makes it difficult to start improvising by playing small variations rather than genuinely new material. On the level of learning that we are working with, the system nevertheless successfully adapts after being given feedback for its performance.

Explicit rules can be found in the form of chord scales; section 2.4.2 describes how note pitches are adjusted according to hard-coded chord scales. On the level of meta-learning, the decision of *what* is being learned can be seen as explicit

knowledge, i.e. that note functions are being learned is a design decision that includes the explicit information that note-functions are of importance.

3.3 Playing behaviour

The features of humans playing behaviour summarised in section 1.3 were

- (g) Musicians play by chunks of notes rather than note-by-note
- (h) Possibly note-level rules for expression, articulation, timing etc.
- (i) Musicians can respond to musical input
- (j) Use of repeating motifs
- (k) Musicians can improvise over previously unknown songs but will also retain good improvisations for known songs.

The chunks of notes are realised by means of fragments. The optimal length of a fragment is arguable of course, even more so whether it is at all feasible to have fragments of fixed sizes. Yet, given a suitable algorithm for “intelligent” segmentation of solos into rhythmic and melodic fragments of variable length, there are no theoretical constraints that limit the system described to fixed length fragments. Whilst no note-level rules are defined yet, possible extensions towards them will be discussed in section 4. The same applies to the interaction with human musicians, for which the system principally qualifies due to the possibility to convert arbitrary improvisations into the internal representation using rhythmic and melodic fragments. Details will be discussed in section 4.

The use of repeating motifs can be observed after training a player for a specific tune as certain pairs and sequences of rhythmic and melodic fragments will be likely to occur again in a solo. These motifs are, though limited by the number of distinct fragments in the virtual player’s repertoire, specific to different tunes that can indeed be trained individually. Yet motifs are repeated only on an associative basis; there is no explicit structure to the performances that would allow for top-down influence on when and how motifs are played and repeated.

3.4 Other limitations

We have seen that, as intended, the proposed framework copes well with learning by analysing existing performances, adaptiveness to human feedback, development of distinct solos for different, previously unseen tunes. Nevertheless, to be true to the phenomenology of human jazz players, some limitations that have not been addressed yet should not be passed over.

First of all, the output is limited to exact quantisation of rhythmic fragments and even more so, to a very limited number of distinct rhythms. Yet, this issue is less severe in comparison to other approaches such as GenJam or CHIME, which use quavers as the smallest quantisation and would suffer severe learning difficulties with more fine grained quantisations as the size of the solution space in these systems grows exponentially with the resolution of the quantisation.

Furthermore, the representation of melodies seems to limit the aesthetic quality of the output, and up to now there is no way to e.g. alter the learned conditional note functions after the creation phase. Setting all note functions to ‘h’ (for **chord** tone) makes the output more audible, but also considerably less interesting. Although all approaches discussed in section 1.2.2 neglect the chord progression in what is being played, the results suggest that some influence of the harmony on the performance task would be desirable, i.e. to comply with the

cadence structure of the tune (the end of a phrase is often marked by the base tone of the chord to which a cadence resolves; usually the tonic). Yet it is not immediately fathomable which form this influence would take in the theoretic framework proposed in this thesis.

CHAPTER 4

Conclusion and Outlook

In summary, the results obtained through the study of the implemented system corroborate that, although due to implementational and time constraints certain characteristics had to be neglected, through consistent phenomenological analysis and the deconstruction of existing, intelligible systems with regard to that very analysis it is indeed possible to assemble an artificially creative system that systematically exhibits the desired phenomenology.

One of the core advantages of the system is the extendibility and adaptiveness that partly results from being based on a very general and flexible probabilistic approach that allows for inclusion of new factors into the performance task and modification of a virtual player's repertoire and output on several levels. To conclude this thesis, I'd like to exemplarily suggest two extensions from which the performance and capability of the system would benefit most.

As mentioned in section 3.3, the system still lacks the capability to respond to human input. A simple realisation of "trading fours" could be implemented closely after Biles (1994) model by real-time conversion of human solos into an internal representation and the application of musically meaningful operators on this representation. In our case, this means that a human player's solo has to be fragmented into rhythmic and melodic fragments known to the virtual player – if a fragment played by the human doesn't have an exact match, the most similar fragment has to be computed – and then either modifying the performance task such that a high proportion of the probability mass is shifted towards the fragments and transitions played by the human musician, or by substituting each of these fragments by a different fragment according to their similarity.

A further conceivable improvement derives from the work on automated discovery of expressiveness in music by Widmer (2001, 2005). Using Inductive Logic Programming on a training set of performances of 13 complete Mozart sonatas played by professional pianists on a prepared grand piano, he was able to find simple rules that explain the difference between the piano score and what is actually being played, such as:

```
abstract_duration_context = equal-longer
& metric_strength ≤ 1
⇒ ritardanto
```

"Given two notes of equal duration followed by a longer note, lengthen the note that precedes the final, longer one, if this note is in a metrically weak position ('metrical strength' ≤ 1)." (Widmer (2005))

It seems likely that similar rules can be found for jazz performances by comparing what soloists play to the underlying rhythmic and melodic fragments of a virtual player's repertoire that show most similarity with the performance, and that applying the rules to the output of an artificial jazz player can immensely enhance the "jazziness" and humanness of the music created.

Bibliography

- Allan, M. and Williams, C. (2005). Harmonising Chorales by Probabilistic Inference. *Advances in Neural Information Processing Systems 17: Proceedings Of The 2004 Conference*.
- Baker, D. (1978). *Miles Davis: Trumpet*. Shattinger Intl. Music: C. Hansen, dist.
- Biles, J. (1994). GenJam: A genetic algorithm for generating jazz solos. *Proceedings of the 1994 International Computer Music Conference*, pages 131–137.
- Bookstein, A., Klein, S., and Raita, T. (2001). Fuzzy Hamming Distance: A New Dissimilarity Measure. *Combinatorial Pattern Matching: 12th Annual Symposium, CPM 2001, Jerusalem, Israel, July 1-4, 2001: Proceedings*.
- Chen, S. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics Morristown, NJ, USA.
- Dahlstedt, P. and Nordahl, M. (2001). “Living Melodies”: Coevolution of sonic communication. *Leonardo*, 34(3):243–248.
- Eck, D. and Schmidhuber, J. (2002a). Finding temporal structure in music: blues improvisation with lstm recurrent networks. *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 747–756.
- Eck, D. and Schmidhuber, J. (2002b). Learning the Long-Term Structure of the Blues. *Artificial Neural Networks-Icann 2002: International Conference, Madrid, Spain, August 28-30, 2002: Proceedings*.
- Francu, C. and Nevill-Manning, C. (2000). Distance metrics and indexing strategies for a digital library of popular music. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 2.
- Franklin, J. (2001). Multi-phase learning for jazz improvisation and interaction. *Proc. Eighth Biennial Symposium on Arts and Technology*.
- Hamming, R. (1986). *Coding and information theory*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA.
- Hofmann-Engl, L. (2001). Towards a cognitive model of melodic similarity. *Proceedings of the 2nd Annual International Symposium on Music Information Retrieval (ISMIR 2001)*, pages 143–151.
- Johnson-Laird, P. (1988). *The computer and the mind*. Harvard University Press Cambridge, MA, USA.
- Keller, B. (2007). How to Improvise Jazz Melodies.
<http://www.cs.hmc.edu/~keller/jazz/improvisor/HowToImproviseJazz.pdf>.
- Keller, B., Jones, S., Thom, B., and Wolin, A. (2006). An Interactive Tool for Learning Improvisation Through Composition. Technical report, Technical Report HMC-CS-2005-02, Harvey Mudd College, 2006.

- Keller, R. and Morrison, D. (2007). A grammatical approach to automatic improvisation. *Proceedings, Fourth Sound and Music Conference, Lefkada, Greece, July*.
- Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press.
- Ó Maidín, D. S. (1998). A geometrical algorithm for melodic difference. *Computing in Musicology*, 11:65–72.
- Papadopoulos, G. and Wiggins, G. (1998). A Genetic Algorithm for the Generation of Jazz Melodies. *SteP*, 98:7–9.
- Poincaré, H. (1913). *The Foundations of Science*. The Science Press, New York.
- Ramalho, G. and Ganascia, J. (1994). Simulating Creativity in Jazz Performance. *National Conference on Artificial Intelligence*, 1:108–113.
- Rubner, Y., Tomasi, C., and Guibas, L. (1998). A metric for distributions with applications to image databases. In *Computer Vision, 1998. Sixth International Conference on*, pages 59–66.
- Sloboda, J. (1985). *The musical mind*. Clarendon Press.
- Taylor, C. (1988). Various approaches to and definitions of creativity. *The nature of creativity: Contemporary psychological perspectives*, pages 99–121.
- Thom, B. (2000). *BoB: an interactive improvisational music companion*. ACM Press New York, NY, USA.
- Toussaint, G. (2003). Classification and phylogenetic analysis of African ternary rhythm timelines. *Proceedings of BRIDGES: Mathematical Connections in Art, Music and Science*, pages 25–36.
- Typke, R., Giannopoulos, P., Veltkamp, R., Wiering, F., and van Oostrum, R. (2003). Using transportation distances for measuring melodic similarity. *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, pages 107–114.
- Van Rossum, G. and Drake, F. (2003). *Python Language Reference Manual*. Network Theory.
- Walker, W. (1997). A computer participant in musical improvisation. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 123–130.
- Widmer, G. (2001). Discovering Strong Principles of Expressive Music Performance with the PLCG Rule Learning Strategy. *Proceedings of the 11th European Conference on Machine Learning (ECML 01)*.
- Widmer, G. (2005). Studying a creative act with computers: Music performance studies with automated discovery methods. *Musicae Scientiae*, 9(1):11–30.

“Most of the soloists at Birdland had to wait for Parker’s next record in order to find out what to play next. What will they do now?”

Charles Mingus