Evaluating Music Improvisation Algorithms with a Modular Trading Fours System

Thomas Sjerps

Evaluating Music Improvisation Algorithms with a Modular Trading Fours System

by

Thomas Sjerps

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday June 19, 2025 at 11:00.

Student number:	5058287	
Project duration:	November 11, 2024 – June 19, 2025	
Thesis committee:	Dr. ir. A. R. Bidarra,	TU Delft, thesis advisor
	Dr C. C. S. Liem MMus,	TU Delft, daily supervisor
	Dr C. Raman,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

Six years of studying in Delft have flown by, and I want to start out by thanking all of the friends that I made during my study period here. Even though my study time involved a year of sitting secluded at home, I am still very happy and proud of all of the exciting people I've met, and all of the things that I have been able to do after 2020 outside of my studies. A special word of thanks goes out to the *Delftse Studenten Jazz Vereniging Groover*, the jazz association that has brought me nothing but joy (and back pain) since joining in 2021.

I would like to thank my parents for their continued support. A final thanks goes out to my thesis supervisors, Dr. ir. Rafa Bidarra and Dr Cynthia Liem MMus, who have helped me greatly in developing and researching this amazing topic (something I am still thankful for) and who have encouraged me to push my academic boundaries for the past seven months.

Thomas Sjerps
Delft, June 2025

Abstract

In musical (jazz) improvisation, musicians that are just starting out can often feel uncomfortable when being put on the spot by their fellow players. However, when a musician is on their own when practising or leisurely playing, this prevents them from listening to fellow musicians. When a musician wants to experience some notion of co-play when they are on their own, computers and musical generative techniques may be a source of help. We study the extent to which music improvisation algorithms can facilitate such interactions by proposing an experimental framework to evaluate and compare these different algorithms. We achieve this by developing MILES (*'Mixed-Initiative musicaL interactivE System'*), a generic music improvisation system that allows a musician to improvise with various musical improvisation models and facilitates comparative evaluation of these models. *MILES* makes use of the 'trading fours' paradigm, where two or more musicians exchange four measures of solo material. We conduct experiments with novice and advanced musicians in expert-pupil and peer-to-peer settings that compare differing algorithms, as well as different variations of similar algorithms. These comparisons are based on self-assessed opinions and third-party grading and ordering of recordings, based on improvisational reciprocity and enjoyment. Symbolic music recording analysis further quantifies the interactivity between the musician and the algorithms. With this experimental setup, we are able to track familiarity and enjoyment of using music improvisation algorithms, and compare different iterations of similar music improvisation algorithms.

Contents

1

Introduction

Improvising music is fun, and also very beneficial for other skills, like problem solving [1]. In principle, most jazz improvisation is very accessible: quite often, songs are played that are considered 'standards' due to their popularity, most improvisation structures centre around the 'form' of these standards and during open jams beginners are often welcome join in. In practice however, many beginning improvisers feel a mental barrier when performing. A form of pressure can emerge when musicians are put on the spot to make up and play novel musical material. It is reasonable to assume that alleviating this pressure increases comfort and enjoyment of performance.

Musicians already have existing options rehearse improvisation on their own: creating transcriptions of existing solos, soloing over backing tracks and practising more technical facets of playing (scales, rudiments, etc.) are tried and tested ways of learning jazz improvisation. However, these techniques lack the interplay and reciprocity of improvising with multiple people. Computational real-time systems might bridge this gap, by providing musical material to the musician in real-time.

The terminology used in articles describing existing music improvisation systems can vary a lot: the words 'system', 'model' and 'algorithm' vary in definition from system to system. In this research we use the following definitions (by taking an existing music improvisation system called *Mimi* as an example):

- A **music improvisation system** is the full unit with which music improvisation can take place. This includes the type of symbolic input (Does the user use a MIDI keyboard? Do they use a trumpet with live transcription? Etc.), the modality of the interaction (Do the human and computer alternate their play, or do they play over each other? Etc.) and the musical output (Does the system connect to a digital audio workstation, or does it generate the audio itself?). An example of a music improvisation system is *Mimi* itself.
- A **music improvisation algorithm** is responsible for taking the symbolic data, potentially processing it, and transforming it into some new symbolic data. Algorithms potentially have access to previously played material of the musician, external pre-existing solo material and their own generated material. The music improvisation algorithm of *Mimi* is responsible for dividing its musical input into chunks.
- A **music improvisation model** is a generic component which can be used to ingest ('train') and generate ('infer') any type of data. *Mimi* uses a 'factor oracle' model which re-emits its input musical chunks in a different order.

A diagram further illustrating our terminology can be found in Figure 1.1, and *Mimi* will be further discussed in chapter 2.

To our knowledge, existing music improvisation systems have been used only for musical demonstrations done by their (often musically capable) creators and other music experts, not for improvisational research with novices. In this research, we take inspiration from these systems by adapting their algorithms and evaluation strategies into a new experimental setup and improvisation system. We will make use of the 'trading fours' paradigm, where musicians alternatingly 'trade' musical material by playing four measures of musical material at a time: in addition to being an existing jazz

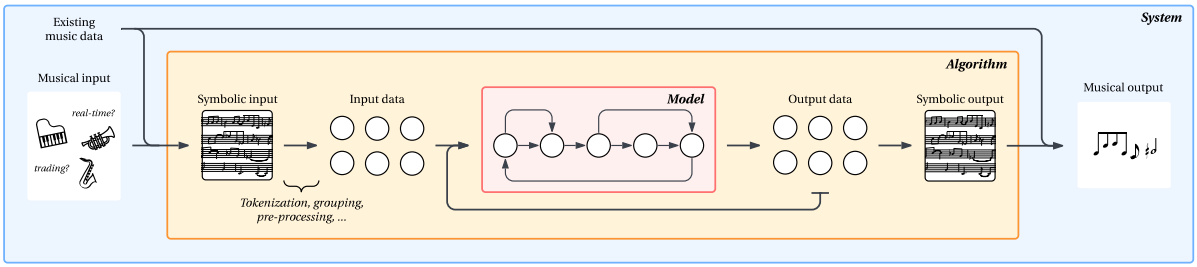


Figure 1.1: Diagram giving an overview of the terminology used in this thesis.

improvisation structure, the aspect of exchanging small snippets of musical material creates a computationally convenient environment for real-time generative algorithms. With the experimental design, we aim to answer the following research question:

To what extent can music improvisation algorithms facilitate a trading fours interaction with a novice musician on their own?

In order to answer this question, we define further sub-questions:

1. *How can familiarity and enjoyment in improvisation be measured for novices over time?*
2. *How can iterative comparison of music improvisation algorithms be carried out?*
3. *What kinds of algorithms and evaluation techniques are best suited for an improvisation interaction between improvisation algorithms and novice musicians on their own?*

The research will follow three phases. Firstly, the setting and evaluation of the experiment are defined (see chapter 3). Next, we outline the algorithms that are used in these experiments (see chapter 4) and we describe the implementation of the system that supports improvisation with and evaluation of these algorithms (called *MILES* (*'Mixed-Initiative musicaL interactivE System'*)). Finally, two experiments are carried out. The first of these experiments aims to compare various different models alongside each other over time using self-assessment, expert feedback and symbolic music analysis (see chapter 6). The other experiment compares different iterations of the same model using self-assessment, peer feedback and symbolic music analysis (see chapter 7).

2

Related Work

Since the 1950s, people have used computers to generate music. In these years, one of the first symbolic music generation algorithms (based on Markov chains) [2] and 'computer music programming languages' like *MUSIC I* [3] were introduced. Increased computing power allowed for more complex algorithms to be used, like genetic algorithms [4] [5] and grammar-based algorithms [6]. A further increase of computing power allowed for the advancement of machine learning techniques, which were increasingly incorporated into music generation systems. Systems like *JazzGAN* [7], *BebopNet* [8] and *Multitrack Music Transformer* [9] respectively, use algorithms with GAN, LSTM and Transformer models for generating symbolic music.

The increase in computing power not only advanced music generation algorithms, but also saw the advent of real-time music improvisation systems. These systems adapted models from these 'static' music generation techniques to work with real-time input and output. In this chapter, we first describe these existing systems through the lens of mixed-initiative systems. Next, we detail existing music visualisation techniques and existing music corpora which these systems may make use of. Finally, we describe the way that existing ('static' and improvisational) music systems are evaluated.

2.1. Mixed-initiative systems

In procedural content generation, a mixed-initiative system is a system in which a work of design is iterated upon between a human and a computer system. By interleaving these iterations, the user can take inspiration from suggestions made by the computer, and the computer can use content and preferences of the user to generate novel and potentially interesting design candidates.

An example of these mixed-initiative systems is *Sentient Sketchbook* [10], a procedural content generation system that aids a human author with creating game levels. While the user is sketching a potential tile map of such a level, a genetic algorithm provides suggestions based on the current design. By allowing the user to pick and improve on these suggestions, the human and system together make iterations throughout the creative process towards a finished game level. Another system, described by Gain et al. [11], allows a user to control the placement of procedurally generated terrain by sketching an outline of mountains and canyons.

Mixed-initiative systems may also allow for creation of musical material. *ProceduraLiszt* [12] allows a user to compose a piece of music by setting constraints for a hierarchical Wave Function Collapse algorithm, and to iterate on tweaking those constraints to converge to a desired composition.

2.2. Music improvisation systems

(Musical) content generation does not necessarily consist of alternatingly changing a single static work. In the context of real-time improvisational (jazz) music, a different perspective is required: musicians playing together in real time (taking turns, playing over each other, etc.) inherently creates a musical artifact on the spot, and as such, there is no room for iterative improvement and refinement towards a static final outcome. There exist music improvisation systems that allow for a user to play back and forth with an artificial agent. This section describes a selection of these systems.

Music improvisation systems can use many different algorithms to generate symbolic music. Thom takes the idea of 'believable agents' (defined by Loyall as "personality-rich autonomous agents" [13]) and creates a personalized trading fours 'believable improvisation music companion' by using a variable-length tree encoding scheme with extensions for temporal information [14]. Many of the first improvisational symbolic music generation systems relied on variations of Markov chains. For instance, Pachet's *The Continuator* [15] uses an algorithm that interactively and iteratively trains a variable-order Markov model with played musical phrases. In contrast to a fixed-order Markov model, this model uses higher-order continuations for more musically structured generations, while gradually lower-order continuations serve as fallbacks to prevent generation from stalling when no higher-order continuations are found. Many playing modes are outlined, where musicians can play with musical 'databases' from a single musician, a cumulative database that is built over time, and databases from each other.

Biles' *GenJam* [16] trades fours with a genetic approach by storing a 'population' of musical phrases per song, with real-time musician-assessed 'fitness scores' (indication of 'good' or 'bad'). During the training phase, these phrases are genetically bred and mutated with various symbolic music operators, which become the new population of phrases. During a performance, these phrases are retrieved and played without alteration. *Interactive GenJam* [17] takes the genetic mutations described in the original paper to generate trading fours by simply mutating the four measures played by the human performer, thereby getting rid of the need for the musician to first assess a large population of musical phrases over many generations. Another model that is used for real-time music generation is called a factor oracle. Originally developed for fuzzy pattern matching, this model represents an automaton with links representing each snippet of material from one node to the next, with possible links based on suffix structure [18]. The factor oracle builds a structure around a string p of length m. Construction of the model makes use of the fact that the 'supply' of every node i of the factor oracle of p (the index where the longest repeated suffix of p up until i ends) does not change when a new node is added. Thus, a value v can be added to the factor oracle of p, by appending a new node and linking it to the node of p_m, going to its supply node, adding a transition from it to $m+1$ with value v, going to the supply of that node, adding a transition of it to $m+1$ with m, et cetera. Creation of additional links stops when a transition for v is already present, or the start is reached. The result of adding a value of 'a' to a factor oracle of the string 'abbb' can be found in Figure 2.1.

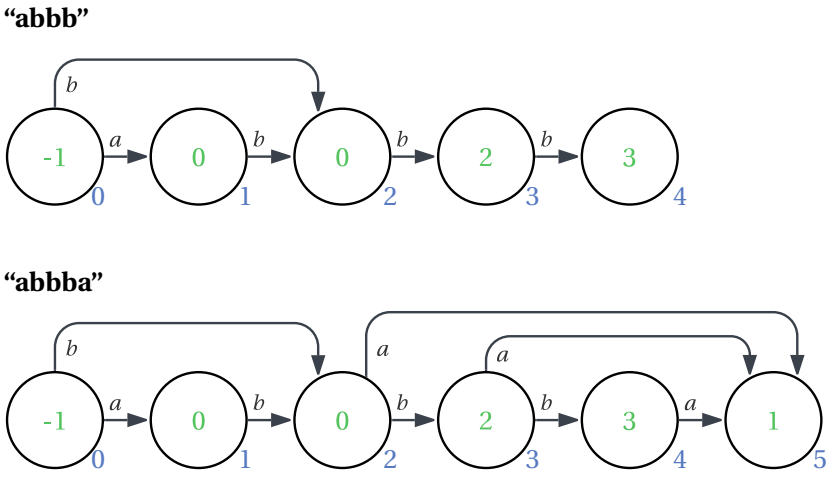


Figure 2.1: Addition of a value to a factor oracle. Supplies are shown coloured green inside of nodes, indices are shown at the bottom-right of nodes.

By repeatedly adding nodes, a fuzzy factor searching automaton can thus be constructed. Any material that can be encoded as a string can be fed into a factor oracle, and by traversing the model's automaton, this material can be generated respecting a notion of structure of the original material. Thus, by feeding symbolic music to a factor oracle and traversing it, convincing novel musical material can be generated [19].

Due to its simplicity and computationally efficient on-line real-time training and generation, many

music improvisation systems have been developed with the factor oracle model. One such system, *OMax* [20], can use symbolic music data for improvisation as well as audio data. The *OMax* algorithm (called *Open Music*) has been used in many contexts, like live performance (where the human and system continually play over one another) and style replication of a specific artist [21]. Another improvisational system where the human performer and system play over one another is called *Mimi* [22]. *Mimi* uses the factor oracle model in the same way as *OMax*, but does not immediately play the musical data it generates. Instead, the system schedules notes to be played a certain time in the future, allowing the user to anticipate and 'play towards' the generated material. *Mimi4x* [23] primes four *Mimi* models with initial musical content and allows them to improvise among themselves. The *ImproteK* [24] music improvisation system defines music improvisation to be over a "scenario" (structured constraints, like a lead sheet) with a specific "memory" (external or already played work). It uses these definitions to extend the factor oracle techniques of *OMax* (and its derivatives) with anticipatory behaviour.

The music improvisation system *AM-I-BLUES* [25] is a combination of two systems made for novices. The first, *MyJazzBand* [26], allows novice musicians to play approximate musical events, which are algorithmically transformed into stylistically 'correct' musical output. The second system, *AMIGO* [27], consists of a keyboard that shows outputs of a Markov-chain based improvisation model by lighting up LEDs above the keys. Together, the *AM-I-BLUES* system gives multiple LED 'suggestions', and allows for on-the-fly production of harmonic progressions.

2.3. Music visualisation

There exist music visualisation techniques, which turn (symbolic) music data (often in MIDI format) and turn it into pictures or videos. Music visualisation techniques can show the feeling, structure or another viewpoint of a piece of music at a certain point in time, or over time. This chapter gives a rough overview of some existing music visualisation techniques.

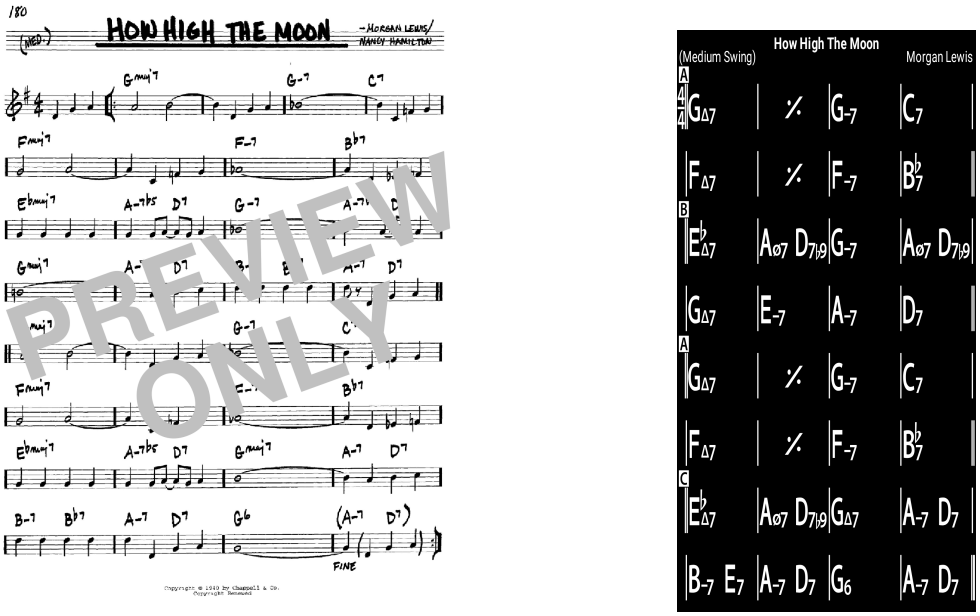
It is possible to show musical structure and rhythm of music with computer visualisation techniques. Such techniques include a self-similarity matrix of (the audio, MIDI data, etc.) of a piece of music [28] or an 'arc diagram' [29], where repeated fragments of material are shown graphically. Pitch and harmony of music can also be shown with computer visualisation techniques. The *spiral array model* [30] is based on a spiral-shaped 'Tonnetz' [31] (tonal net) mapping pitches into a tonal grid. Another visualisation technique based on pitch and harmony, called *ImproViz*, shows a melodic line graph and self-defined 'harmonic palette' of a specific jazz solo.

In a typical jazz improvisation setting, musicians have a so-called 'lead sheet' in front of them, which contains the structure and chords of the standard they are currently playing. This lead sheet could be a physical book (i.e. a *'Real Book'*), or it could be an app on their phone or tablet, like *iReal Pro* [32]. *iReal Pro* only shows the chords of a standard (due to copyright and licensing constraints), whereas a real book also shows the melody. This difference in layout can be seen in Figure 2.2. In *iReal Pro*, users can also choose to listen to a backing track which is automatically generated based on the given chords of the song. The visualisation that is used with this feature highlights the currently-playing measure, which can be seen in Figure 2.3.

Some existing music improvisation systems already make use of visualisation for conveying their internal state. In one of the many performances of *GenJam* [16] that can be seen online [33], a piano roll visualisation can be seen that shows all notes scrolling from right to left, including the notes that the system generates itself. As well as that, the system shows the currently-playing bass note, and some circles and stripes to indicate MIDI percussion events. *Mimi* [22] also uses a piano roll to visualise all notes that are being played. The fact that *Mimi* schedules notes to be played at a later time influences the way that this piano roll is laid out. The delay from generation to playback is shown by scrolling all notes from right to left on the screen, and showing the current time as a vertical line at the centre of the screen. By emitting all human-played notes from this centre line, but all *Mimi*-generated notes from the far right side of the screen, the human performer can anticipate the notes that will be played as they reach the vertical line.

2.4. Existing (jazz) musical material

Most music generation techniques depend on a corpus of existing music to train from. Audio datasets, like those used by *Deezer* for creating their stem separation tool *Spleeter* [34], are used for analysing



(a) Hal Leonard Real Book.

(b) iReal Pro.

Figure 2.2: Lead sheet visualisations of the *Hal Leonard Real Book* and *iReal Pro* (both of jazz standard *How High the Moon*).

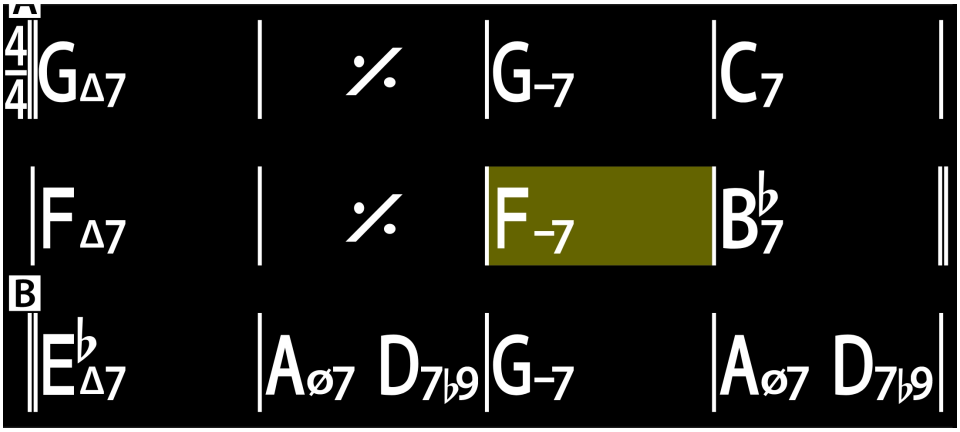


Figure 2.3: *iReal Pro* leadsheet visualisation during playback.

generating audio data. This research only considers generating and analysing symbolic music. When generating symbolic music, the corpus needs to comprise transcriptions of music in symbolic format. A selection of these symbolic corpora is outlined here.

Symbolic corpora exist for many different genres. Folk music has been preserved in various corpora: the *Lyra* dataset contains 1570 pieces of Greek traditional and folk music, and the Meertens Institute has a collection of many Dutch songs originating from the Middle Ages up until the 20th century [35]. These songs are stored in the *Nederlandse Liederenbank*[1] and are documented in various ways, ranging from audio recordings to transcriptions in various file formats. There also exist many classical corpora. The *Symbolic Orchestral Database* links piano scores to their corresponding orchestrations [36]. The *Lakh MIDI Dataset* [37] consists of roughly 100.000 MIDI files of all genres and styles. Via the *Million Song Dataset* [38], a subset of these MIDI files are annotated with song metadata, like genre tags.

There also exist symbolic jazz music datasets. The *Jazzomat Research Project* [39] consisted of creating and analysing the *Weimar Jazz Database*. This database contains 456 annotated monophonic jazz solos (including underlying chords, note volume, music style, feel and other metadata). The *Dig*

[1] www.liederenbank.nl/

That Lick project [40] extends the Weimar Jazz Database with a number of automatically transcribed solos for use in their analysis. Jazz sheet music publisher Hal Leonard sells so-called *Omnibooks* [41] with transcribed solos of jazz masters (Charlie Parker, John Coltrane, Sonny Rollins, Stan Getz, etc.), in order for (novice) jazz musicians to study and practice these existing solos.

2.5. Evaluation techniques

When a new music generation model or system is presented, it is often accompanied by an evaluation to measure its efficacy or similarity with respect to some other musical source. In this section, we detail evaluations made by existing non-improvisational music generation systems, as well as existing music improvisation systems.

2.5.1. Non-improvisational music generation analysis

Music generation techniques can be evaluated in several different ways, ranging from subjective surveys to more objective and computational evaluation. This section outlines such existing subjective and computational analysis techniques.

As music is a subjective field, letting subjective analysis be part of evaluation is an evident choice, by asking listeners to rate recordings or discern them from other recordings. The *Jazz Transformer* and *Multitrack Music Transformer* use survey methods where multiple systems are rated on overall quality, structure and richness. The former uses this for comparing their differently trained models, while the latter compares their system to different systems. *FIGARO* is evaluated by asking participants to listen to two samples of different generation techniques at a time, and picking the best of the two. In these studies, the participants have a variety of skill levels, but these are not considered in their analysis. *MINGUS*' [42] evaluation survey asks participants to rate its output together with *BebopNet* output and canonical *Weimar Jazz Database* solos. Frieler and Zaddach [43] prepare different stimuli of their system, ranging from deadpan MIDI generation, to recordings of students, jazz legends and one of the authors. Both evaluations take the skill level of participants into account, and do indeed show a difference in assessment between skill levels.

Many static systems opt for a computational analysis for their evaluation. Jazz generation system *BebopNet* [8] coins the term 'harmonic coherence' in their analysis, which consists of counting the fraction of a measure in which notes match the chord they are played over. Moreover, to our knowledge *BebopNet* is the only technique to carry out a plagiarism analysis, by finding matches of the original dataset in generated material (up to transposition). Another jazz generation system, *MINGUS* [42], uses the *MGeval* objective evaluation toolbox [44] for its objective analysis to compare itself to *BebopNet*. McKay [45] lists many global features on musical content, as a stage in his proposed pipeline for automatic genre classification. Some of these can be applied to monophonic symbolic melody data, like average note duration and variability of time between attacks. There exist more toolkits and works that list many different global features: in his work on quantifying rhythmic complexity, Abrams [46] defines a set of 15 global features, and Müllensiefen's *FANTASTIC* toolkit [47] contains only monophonic symbolic melody features. In another work, Müllensiefen et al. [48] define many musical similarity metrics, with which two pieces of musical material can be compared. One of these techniques, called the edit distance, is equal to the minimal number of 'edits' (basic transformations) that need to be carried out to transform one string into another. This edit distance can be applied to different transformations of musical material (i.e. edit distance of all pitches in a fragment, edit-distance of all intervals in a fragment, et cetera).

2.5.2. Music improvisation system analysis

Evaluation of music improvisation systems brings its own challenges, when compared to music generation systems. For instance, Pachet and Roy[49] point out that in a common jazz improvisational setting, musicians do not 'really' interact with each other when this is evaluated audio-based similarity metrics. Aside from questions about the efficacy of these kinds of computational analyses, the fact that a human performs part of the interaction opens up the potential for many different kinds of self-assessment and third-party assessment evaluation techniques.

Most music improvisation systems opt for an evaluation of surveying the performers themselves. The *AM-I-BLUES* [25] improvisation system uses heuristic evaluation [50], which relies on the fact that when comments from multiple test participants is aggregated (even with a very low amount of

participants), the results can find most usability problems. *GenJam* [16] is described by its author Al Biles as "competent with some nice moments". Moreover, variations of the *Mimi* [22] and *OMax* [21] systems have been positively described by the experts who have performed with them. Third-party participants can be asked to rate recordings from music improvisation systems after they are performed. For instance, *The Continuator* [15] was evaluated by asking (unspecified) 'listeners' if they could tell the system output apart from human-made music.

3

Experimental Design

In this research, we aim to answer to what extent can music improvisation algorithms facilitate a trading fours interaction with a novice musician on their own. Novice musicians have the most to gain from familiarity, and can give more contrasting data than experts who are already familiar with improvisation and can more easily become comfortable with new improvisational systems. Thus, we see a use in evaluating (and comparing) music improvisation algorithms in the context of familiarity, progression and improvisational reciprocity by carrying out a set of experiments.

We start this chapter by describing the improvisational interaction we seek to carry out. Next, we take a look at what subjective evaluation methods we can use from existing techniques (see chapter 2.5.2), and then take a look at additional evaluation techniques we can consider based on symbolic analysis. Finally, we fully detail the experiments we set out to perform.

3.1. Trading fours

We intend to keep the experiment relevant to actual improvisational interaction by keeping the user study as close to a real improvisational interaction as possible. As well as that, we aim to carry out comparative analysis by creating a level playing field, where every algorithm is presented with exactly the same interaction characteristics.

Jazz music usually consists of repetition of the same chord structure, known as the 'form'. In a typical jazz performance, the first and last instance of the 'form' are usually performed with the original melody, with all other playthroughs of the 'form' being designated to soli. In such soli, accompanying instruments keep playing while the soloist plays improvised musical material. Soli can take an entire playthrough of a form, though many other configurations exist.

One such configuration, 'trading fours', consists of many soloists collaborating by alternatingly playing four measures of solo material. In some variants of trading fours, more variation is created by letting percussion players (drums, etc.) solo over every other 'four'. Trading does not necessarily have to take up four measures: 'eights', 'two's', or even single measures can be taken when trading. By interleaving short snippets of solo material, trading fours gives musicians a more immediate chance to collaborate with, listen to and take inspiration from one another.

Like already discussed in chapter 1, we choose to use trading fours as the main interaction of improvisation in our experiments because of this immediate collaboration and the consequent computationally convenient improvisation environment for real-time generative algorithms.

While many instruments can be used in jazz improvisation, we focus on evaluating trading fours with jazz piano players. This is done due to the popularity of the instrument, the ease of use of working with (MIDI) keyboards in interactive software and our own personal familiarity with piano playing.

3.2. Evaluation

In line with other music system analyses, we aim to survey participants on the (musical) qualities of the improvisation interaction, as well as provide symbolic analysis. This section describes our

goals and aims based on self-assessment and third-party assessment criteria, symbolic analysis and tracking of progression.

3.2.1. Enjoyment and familiarity

One of our goals is to to measure enjoyment of playing with the chosen models of the system. To achieve this, we opt to survey to what degree the novices felt inspired, and enjoyed playing with the system. In order to give them the opportunity to give additional feedback, we will also provide an option for free-form input for further elaboration of their opinions.

3.2.2. Reciprocity

In a real-life trading fours setup with human players, it is imaginable that while trading, the players get inspired by one another and thus will refer to earlier material while improvising. Thus, we wish to study the feeling of improvisational reciprocity of the chosen models of the system. Therefore, we will survey whether we can observe musical interaction between pupils and algorithms. Response in improvisation with any partner is a two-way street: each party has some efficacy of responding to the other. Thus, we survey the pupils on how the algorithm responded to them, as well as on how they could respond to the algorithm. Moreover, we aim to create different improvisation interactions that functionally differ in their reciprocity (for instance, an algorithm that does not respond to human input at all).

3.2.3. Third-party assessment

Music generation systems are often evaluated by (experienced) musicians listening to and comparing recordings of various techniques, including potentially human performances. We wish to include this notion of assessment with groups of different skill levels.

For the evaluation of our approach, we choose not to follow past strategies in which users are asked to distinguish machine output from user input or canonical (solo) material (as e.g. is done in Pachet's *the Continuator* [15]. To our feeling, such a Turing test style evaluation places an improvisation algorithm in a frame of competition with (and possible replacement of) the human musician. Instead, we rather want to analyse whether pupils feel empowered to play with the algorithms, and whether this aligns with the assessment of external listeners. We aim to achieve this by surveying these listeners with the same questions as were given to the performers.

Firstly, we aim to introduce a notion of expert feedback into our evaluation. As experts potentially have a role in coaching novices, they potentially have a more developed ear for musical competence in students and may pass more musically informed judgements. This expert feedback will later be included in the first of our experiments (see chapter 3.3.4). Secondly, we aim to also include peer evaluation as a potential evaluation technique. By making sure that all novices that use the system have the same level of familiarity with improvisation, we aim to study the difference between like-minded and expert assessment. This peer feedback will later be included in our second experiment (see chapter 3.3.5).

3.2.4. Symbolic analysis

Although Pachet and Roy [49] show no reciprocity of improvisation of jazz musicians by analysing audio, we believe that additional research in using computational music analysis for measuring reciprocity of improvisation with a music improvisation system is needed. Applying these techniques to our evaluation of improvisational music generation models could prove to be a useful extra viewpoint for analysis.

In this research, we will use global features of symbolic music data, inspired by existing 'toolkits' of these analyses. In our global features, we look for properties of contour, pitch and rhythm, as we deem these properties most important to (monophonic) jazz solo analysis. We aim to measuring a potential trend in symbolic metrics within the human material throughout a recording, and similarity metrics between human and system musical material, to signify a change in style of human output when exposed to a musical improvisation system.

These features do not directly analyse the reciprocity between the fours played by the human and the algorithm. For explicitly quantifying this reciprocity, we also look for a similarity metric between human and algorithmic 'fours'.

3.2.5. Tracking progression

We aim to track familiarity and enjoyment over time in our research, in order to get a deeper understanding of how novices can use music improvisation systems on their own. This can be achieved by carrying out multiple sessions where the novice plays with the various algorithms, with some timespan in between.

By applying our subjective evaluation and symbolic analysis multiple times, we aim to measure a trend in enjoyment, familiarity and reciprocity. Aside from applying the same techniques over time, techniques can be used where third parties can directly compare material from different sessions. This allows for a more explicit evaluation of sessions over time.

We make a distinction between general improvisational familiarity and specific familiarity of our system. Thus, we intend to introduce a small familiarity phase at the beginning of each session we carry out. This allows us to measure the effectiveness of the system with the algorithms over time.

3.3. Experiments

With the intentions presented in the previous sections, we detail the two experiments we will carry out in this research. We first detail the kinds of algorithms we wish to evaluate, the assessment form we use and the symbolic analyses we carry out. Next, we describe the generic setup of the two experiments we conduct.

3.3.1. Algorithms

In order to present all algorithms with a level playing field, we aim to implement our own music improvisation system where several algorithms can be swapped out.

We choose to evaluate algorithms that we implement, inspired by existing music improvisation systems. In order to potentially improve the musical output of the algorithms, we choose to feed a selection of canonical solos from the *Weimar Jazz Database* to these algorithms. By having these algorithms use the canonical solo material and the human-played material differently, we capture our intent to include many differently-responsive algorithms that we detail in chapter 3.2.2. We intend to at least implement a few algorithms which do not base their generation off of human input, a Markov-based algorithm as used in *The Continuator* [15], a factor oracle-based algorithm as used in *OMax* [21] and *Mimi* [22], and a genetic algorithm as used in *Interactive GenJam* [17].

3.3.2. Assessment form

We have compiled a set of questions regarding enjoyment, inspiration and reciprocity in an assessment form, a variation of which will also be used for third-party assessment. The questions in this form are as follows for self-assessment and third-party assessment:

- I feel that the algorithm could respond well to me / Algorithm responds well to pupil
- I feel that I could respond well to the algorithm / Pupil responds well to algorithm
- I feel that the algorithm gives me inspiration / Pupil got inspired by algorithm
- I enjoyed playing this song with the algorithm / Pupil enjoyed playing with algorithm

The final forms can be found in Appendix A.

3.3.3. Symbolic analysis

In the symbolic analysis we describe in chapter 3.2.4, we aim to make use of global features for melodic contour, pitch and rhythm, and an explicit similarity metric. In this subsection we describe these methods and how we use them.

We aim to use the global features on every 'four' of all recordings, and aggregate them throughout the duration of the performances. With this information, we can see the evolution of the values of a metric over time between human 'fours' and algorithmic 'fours', and throughout time. The global features we use can analyse symbolic music recordings, which contain n notes N over m measures and have the following properties:

- `OutputName` (type: enum): the instrument or player that played this note;

- `Time` (type: double): the time (in measures) of the onset of the note;
- `Length` (tpye: double): the time (in measures) of the length of the note;
- `Note` (type: int): the MIDI note number of the note;
- `Velocity` (type: int): the velocity (0-127) of the note.

The list of global features that were implemented, but ended up not being used, can be found in Appendix C.

For melodic contour, we decided to use 'extrema ratio', which the *Weimar Jazz Database* [39] defines to be the "share of notes with direction reversal (i.e, minima and maxima of pitch contour)". Computation starts by constructing a list of 'melodic arcs', recording the (BPM-dependent) time durations and pitch heights of each. A visual overview of melodic arcs can be seen in Figure 3.1. An algorithm to compute melodic arcs from symbolic input, as well as further metrics using them, can be found in Appendix C.

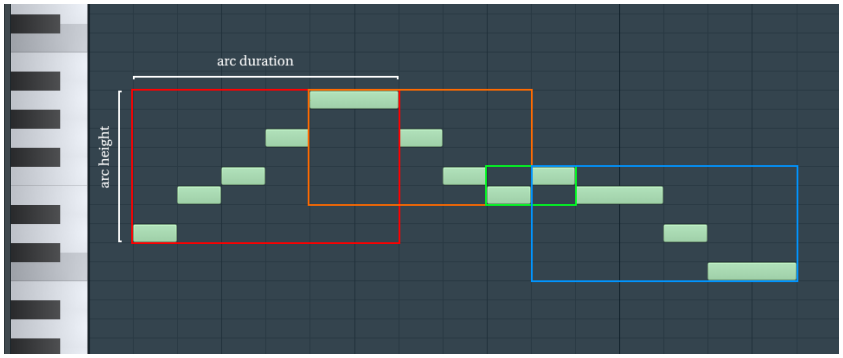


Figure 3.1: A piano roll illustrating the notion of melodic arcs.

We define $A = GetArcList(measures, bpm, I)$ to be a list of durations (in seconds) and heights (in semitones) of each arc. The value of the **extrema ratio** is the amount of melodic arcs (minus one), divided by the number of notes:

$$extrema\ ratio = \frac{1}{n} \cdot (|A| - 1)$$

This metric is low when the melodic contour is very simple, and increases with melodic complexity, thereby making it a logical metric to use in analysis.

Inspired by the evaluation techniques of *BebopNet* [8] and features based on pitch statistics given by McKay [45], we use the **root, third and fifth note share** for our pitch class global feature. This feature measures the prevalence of the root, third and fifth note of the key of the song:

$$note\ share_{1,3,5} = \frac{1}{n} \cdot \sum_{i=1}^{n} (i \bmod 12 \in \{\texttt{Root}(Key), \texttt{Third}(Key), \texttt{Fifth}(Key)\})$$

By measuring the percentage amount of notes that adhere to these primary scale degrees of the key signature of the song, this global feature is a potentially useful candidate in measuring complexity and 'spiciness' of the solo material.

Abrams [46] describes a list of rhythmic complexity metrics. One of these metrics, called the 'Normalized Pairwise Variability Index' (**NPVI**), measures rhythmic contrast by comparing successive inter-onset intervals [51]. For analysing NPVI, we first define these inter-onset interval to be the difference in onset time between notes, and use these to compute the NPVI itself.

$$IOI = \{\texttt{Time}(N_{i+1}) - \texttt{Time}(N_i) \mid i \in \{0, 1, \ldots, n-2\}\}$$

$$NPVI = (\frac{100}{N-1}) \cdot \sum_{i=1}^{N-1} |\frac{IOI_i - IOI_{i+1}}{(IOI_i + IOI_{i+1}) \,/\, 2}|$$

For an additional explicit similarity metric we have chosen the **Levenshtein edit distance** for analysing melody similarity between the fours of the pupil and the algorithm. This edit distance will be calculated over the token representation of the fours, and is equal to the minimal number of tokens of the first 'four' that need to be added, substituted, deleted or transposed in order to get to the second 'four'. We use the external *Fastenshtein*[1] library for a performant Levenshtein edit distance implementation. In our analysis, we will compute the edit distances from every human-played four to its subsequent algorithmic four, and from every algorithmic four to its subsequent algorithmic four. With these algorithm-to-pupil and pupil-to-algorithm edit distances per performance, we aim to measure reciprocity between algorithms, songs, and sessions.

3.3.4. Experiment 1: initial comparison

The first experiment we describe will have the main goal of tracking enjoyment and familiarity of novice musicians ('pupils') with evaluation by self-assessment by the pupils themselves, assessment by musicians with a high musical skill level ('experts'), and symbolic analysis methods. This subsection describes the generic procedure for this experiment.

Firstly, a wide range of algorithms will be developed based on our descriptions in chapter 3.3.1. Next, a selection of jazz standards is made ahead of the experiment, with corresponding canonical solos from the Weimar Jazz database. These solos are fed to the algorithms for potential use in their generation. An initial pilot is conducted with the experts, to evaluate their experience in using our improvisation system, as well as to select three algorithms for later use by the pupils.

A group of pupils will take part in consecutive playing one-on-one sessions, where each session covers the standards with the selected algorithms, in randomized order. After all sessions, each pupil will have improvised with every algorithm on every standard. A supplementary session takes place with the same song-algorithm configuration and order as the first, to allow for expert assessment in progression.

The pupils are informed of the chosen jazz standards several days in advance of the first session, with lead sheet PDFs and a link to an online recording. At the beginning of each session, the pupil first will be (re)familiarized with the interaction with the improvisation software by practising with a standard that will not be used in our experiments. Through these design choices, we try to avoid that the interaction during the experimental session will be confounded with pupils still needing to learn the standard or the general system interaction, and thus maximize the odds that the sessions reflect the musical trading fours interaction between pupil and improvisation system.

Assessment is done in three parts, following our design in Chapters 3.2, 3.3.3 and 3.2.5. Firstly, after every performance, the pupils evaluate their performance and experience with the algorithm using our evaluation form. For an additional view into the performance, the experts that partook in the pilot phase also evaluate the performances of the pupils, using their version of the examination form. Next, we ask the experts to discern between pairs of recordings of the performances of the first session and the last session, by asking them which performance of each pair was recorded last. Finally, we employ the objective symbolic analysis methods we defined to gain additional insight into the reciprocity of interaction between the human and the algorithms.

3.3.5. Experiment 2: evaluating iterations

The second experiment we describe builds off of the first experiment, and has the goal of carrying out iterative comparison of music improvisation algorithms with novice musicians ('pupils'), using evaluation by self-assessment by the pupils themselves, peer assessment from the other pupils, and symbolic analysis methods. This subsection describes the generic procedure for this experiment.

The setup and logistics of this experiment are similar to that of the first, while omitting the pilot phase and multiple sessions and songs, and substituting expert evaluation for peer evaluation. This way, we seek to obtain useful information about the algorithms we set out to improve, while avoiding the time and effort needed for a larger experiment like the first.

[1] GitHub: DanHarltey/Fastenshtein

The goal of this experiment was to carry out iterative comparison of music improvisation algorithms. To achieve this, we improve the algorithms that were used in the first experiment based off of the feedback we collect during the first experiment. We choose to only select a single song from the *Weimar Jazz Database*, due to the narrower scope of this experiment.

Again, a group of pupils are asked to improvise with MILES, though this time only doing so for one session. In order to sidestep the issues in familiarisation when only playing a single session, we ask the pupils that participated in the first experiment to also participate for this experiment. The familiarisation of the pupil will also be controlled by shuffling the order of the algorithms per pupil and including a familiarisation phase in the improvisation session (again, with a different song than the one used for evaluation).

Assessment is done similarly to the first experiment, with pupils being given the same self-evaluation form. However, instead of third-party experts performing evaluation, the pupils themselves are asked to perform peer evaluation by answering the same questions after listening to each other's performances. This is handled with the same evaluation form as used for the experts in experiment 1. Finally, we perform a similar symbolic analysis procedure as the first experiment, using the described global features and similarity metric.

3.4. Ethics

For the user studies of the experiments, the standard-practice human research ethics committee procedure of the TU Delft was followed. Informed consent forms were shared with participants ahead of the research, and participants were asked to fill in these forms in person at the start of their first one-on-one session. All informed consent forms can be found in Appendix A. We use standard TU Delft practices for anonymizing, storing and destroying data, in compliance with local data protection regulation.

For our choice of the jazz standards that participants will play with, we made sure to curate standards that are in the public domain in the Netherlands, the country in which we performed our experiments. The chosen canonical soli technically are not in the public domain in their recorded versions, but as we make use of the MIDI transcriptions of the Weimar Jazz database, and as very short fragments of these soli are used in the trading fours setup, we consider our use to be fair use for studying purposes, in which the original copyrighted works are not reconstructed.

4

Algorithm Overview

This chapter contains an overview of all of the algorithms we use throughout our experiments.

Please note that although all algorithms are introduced here, this also includes the algorithms that are made for and used in experiment 2, based on feedback gathered during experiment 1. This feedback, and further design choices of second-iteration algorithms can thus be found at the start of chapter 7. As will be discussed in chapter 4.1.2, we make use of a tokenization strategy for some of our algorithms, which also has two iterations. These algorithms have been named with 'v1' and 'v2' accordingly.

In experimenting with our music improvisation software, we have developed more algorithms than the selection we describe here. These algorithms can be found in Appendix B. The full implementation of the algorithms can be found in our GitHub repository[1].

4.1. Common functionality

This section describes common functionality that is present in multiple algorithms. We detail the way we model reactivity between the algorithm and the pupil, and our tokenization strategy. More common functionality of our algorithms can be found in chapter B.1.

4.1.1. Reactivity

Most of the algorithms described in this appendix model reactivity in jazz improvisation in the same way:

- Firstly, initialization is carried out, where the algorithm gets the lead sheet of the current song, a canonical solo over this song, and three supplementary solo's by the same artists (though over different songs). In our research, these solos were sourced from the *Weimar Jazz Database* [39]. The algorithm has the possibility to ingest these solos and can already 'learn from' them before the human starts playing.

- Next, whenever the human improviser finishes a 'four', it is passed over to the algorithm. The algorithm can again learn from this material.

- Immediately afterwards, the algorithm is asked to generate a 'four', which is played to the user.

The last two steps of this routine are repeated until the song finishes playing. All algorithms derive from the `IAlgorithm` interface. This interface provides an easy-to-use abstraction of MIDI recording and scheduling functionality, by exposing a `Learn` and `Generate` function which take and produce musical material respectively. By providing this abstraction, additional algorithms can be easily implemented.

[1] GitHub: sjerpsthomas/miles

4.1.2. Tokens

Many algorithms depend on our tokenization strategy, which turns monophonic symbolic solo material into a compressed string of characters. These tokens contain information about the pitches, rests, and differing note speeds and velocities of the melodies of the (monophonic) solo they encode. Crucially, we aim for reconstruction of tokens to work, even if the tokens are random and prescribe a musically impossible melody. In this research, we present two iterations of this tokenization approach, which we roughly describe here. The full details of tokenization can be found in Appendix D.

The first iteration of our token pipeline included 7 tokens for chord-scale system (see chapter B.1.1) pitch tokens, a rest token, a passing tone token, four different note length indication tokens, two different note velocity indication tokens, and a measure indication token. During reconstruction, the length and velocity indication tokens respectively modify the length and velocity of subsequent notes until updated by another indication token. 'Tokenization' and 'reconstruction' of notes is roughly carried out with a procedure per type of token, that respectively infers these tokens based on solo input, and attempts to interpret the concrete symbolic information behind these tokens. All of these procedures are hand-written, and make heavy use of heuristics to make sure that the output is musically valid.

The second iteration of our token pipeline increases the amount of pitch tokens to a full 12 semitones, and drops the passing tone token. As well as that, the reconstruction procedure where speed indication tokens are removed and timing info is inferred was improved with an integer programming solution, which balances adherence to the token-prescribed speed with fitting all notes within the 'four' and having all onsets of notes be at a musically logical time.

4.2. 'Baseline' algorithms

In chapter 3, we described the need for algorithms that do not necessarily respond to human musical input. These techniques are described here.

4.2.1. Note Retrieval

The simplest algorithm of all, Note Retrieval, is an algorithm which strongly reflects skilled musical expertise, but without any reciprocity to what a pupil would do. It achieves this by ignoring any human-played notes during the learning phase, and retrieving the canonical Weimar Jazz Database solo wholesale at the currently-playing time during generation.

4.2.2. Token Random v1

This algorithm generates 4 to 8 random tokens per measure. Tokens are selected along a uniform distribution (and exclude measure tokens); the table for this distribution can be found in Table 4.1. After delimiting these measures with measure tokens, these tokens are reconstructed and the resulting notes are played.

Chance	Outcome
15%	`Rest`
25%	`Note` *(random from 1 to 7)*
20%	`PassingTone`
5%	`SuperFast`
7%	`Fast`
7%	`Slow`
7%	`SuperSlow`
7%	`Loud`
7%	`Quiet`

Table 4.1: The probability distribution used by Token Random v1 generation.

4.3. Markov-based algorithms

In chapter 3, we aimed to implement algorithms that take inspiration from many (early) music systems that use Markov chain models (work by Brooks et al. [2], Pachet's *The Continuator* [15], et cetera).

This section presents the algorithms we implemented that make use of these types of models.

During development, we noticed that output quality of these models increased when more canonical solo's were given to the model at the beginning of playback. Thus, in order to improve the generation capabilities of these algorithms, this extended canonical solo data includes three additional solos performed by the same artist as the solo over the currently-playing song (note: these additional solos are over different songs).

4.3.1. Token Markov v1

This algorithm tokenizes all canonical and human solo data (with human solo data appended throughout playback). During the learning phase, these tokens are put into a Markov chain model. We use the external *MarkovSharp*[2] C# library for Markov chain functionality. This Markov model bases its generation on continuations from n-grams of tokens. In our algorithm, we pick generation to be based on trigrams.

The generation phase consists of token generation using the procedure described in chapter B.1.3 with a maximum measure length of 10. These tokens are reconstructed and the resulting notes are played.

4.3.2. Token Markov v2

This algorithm replaces the Markov model used with an implementation of a variable-order Markov model (used by Pachet's *The Continuator* [15], described in chapter 2.2). We have ported this model from a Python implementation by Pachet[3]. For this algorithm, we use a maximum order of 6 tokens.

Similarly to Token Markov v1, we first tokenize all canonical and human solo data and put it into the model as it comes in. The procedure of the generation phase is described in chapter B.1.3 with a maximum measure length of 7. These tokens are reconstructed and the resulting notes are played.

4.4. Factor oracle-based algorithms

As well as Markov-based models, chapter 3 described the relevance of factor oracle-based algorithms that take inspiration from Mimi [22], OMax [21] and its derivatives. This section presents the algorithms we implemented that make use of this model (as explained in chapter 2.2).

In contrast to the algorithms based on Markov models, we saw no need to expand the amount of canonical solo's given to the algorithms at the start of playback.

4.4.1. Token Factor Oracle v1

This algorithm tokenizes the human solo data and the canonical solo corresponding to the currently-playing song. During the learning stage, these tokens are put into a factor oracle model (with human solo data appended throughout playback). The model considers every token to be a single 'character' of the factor oracle 'string'.

The algorithm keeps track of the index within the factor oracle where the just-played human four starts, which we call `HumanFourStart`. At generation, the algorithm starts out somewhere between this index and the end of the factor oracle, and continually traverses the oracle from there. The four measures of tokens are generated using the procedure described in chapter B.1.3 with a maximum measure length of 10. These tokens are reconstructed and the resulting notes are played.

4.4.2. Token Factor Oracle v2

This algorithm works similarly to Token Factor Oracle v1. However, instead of working with single tokens as the 'characters' in the factor oracle's 'string', we use 8 tokens as a single 'character'. We also make use of the second iteration of our token pipeline.

[2]GitHub: chriscore/MarkovSharp
[3]GitHub: fpachet/continuator

5

System Design

In chapter 3, we identified the need for a setup that facilitates the deployment of (existing) music improvisation models, in order to evaluate and compare them. In this chapter, we describe the tools used in the evaluation experiments: the music improvisation system, symbolic analysis framework and evaluation form. Finally, we shortly describe the implementation details of this setup. An abstract diagram of our software setup can be found in Figure 5.1.

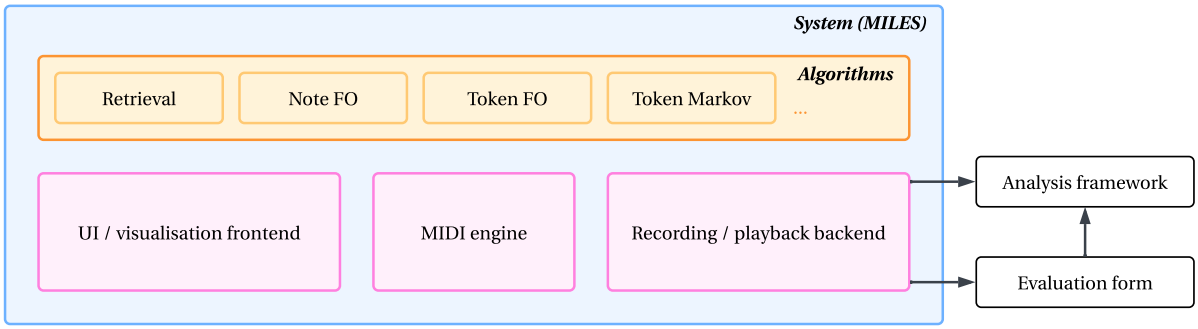


Figure 5.1: A diagram showing the components of our software setup.

5.1. Music improvisation system

We have developed *MILES* (*'Mixed-Initiative musicaL interactivE System'*), a music improvisation system that allows a musician to play with a selection of (existing) musical improvisation models. The system also features a visualisation component, and allows for recording and replaying performances. This section will describe these features and the design choices that led to them.

5.1.1. MIDI engine

There are many ways of handling music with computers. As we chose to limit our research to jazz piano improvisation and many existing music (improvisation) algorithms and piano keyboards work with the MIDI standard, we opted to use this as well. At the centre of all processing done by *MILES* lies its MIDI engine, which is responsible for keeping track of incoming MIDI data and scheduling outgoing MIDI data. Many components, from backing track playback to the session recording functionality detailed later, make use of this system.

5.1.2. Algorithms

In chapter 4, we discuss many music improvisation algorithms. Chapter 3.1 outlines the need for a level playing field when evaluating these algorithms. Thus, *MILES* facilitates the use and implementation of many algorithms.

MILES facilitates music improvisation algorithms, by giving these algorithms access to user content and symbolic transcriptions of Weimar Jazz Database solos. We reduce trading fours to a 'learning' and 'generating' phase, that respectively allow the algorithms to (potentially) train an underlying model and generate content using this model. A full overview of all algorithms that *MILES* currently supports is given in Appendix B.

5.1.3. UI and visualisation frontend

As discussed in chapter 2, there are various ways of visualising music. In order to fit the interaction to typical real-life jazz improvisation (as described in chapter 3.1), we have opted to show a lead sheet as the main focal point in our visualisation. Some extra pieces of visualisation are added in order to aid in improving the clarity of the structure of improvisation to the user. A screenshot of the visualisation can be found in Figure 5.2

We opted to highlight the currently-playing measure, similarly to playback mode in *iReal Pro* [32]. We also saw an opportunity in using colours to convey whose 'turn' it is in the trading procedure. In order to allow the musician to quickly gauge this state, the background of the visualisation switches from white to blue when the algorithm is playing. For extra clarity, every four measures played by the algorithm has a robot icon to the left of it. This icon scrolls along the screen to the right as the four is being played.

Additionally, we wish to give an option for musicians, and third-parties watching (and potentially rating) the performance, to see the generated notes as they are being played. To achieve that, we show a piano keyboard visualisation at the bottom of the screen, which highlights the notes that are played.

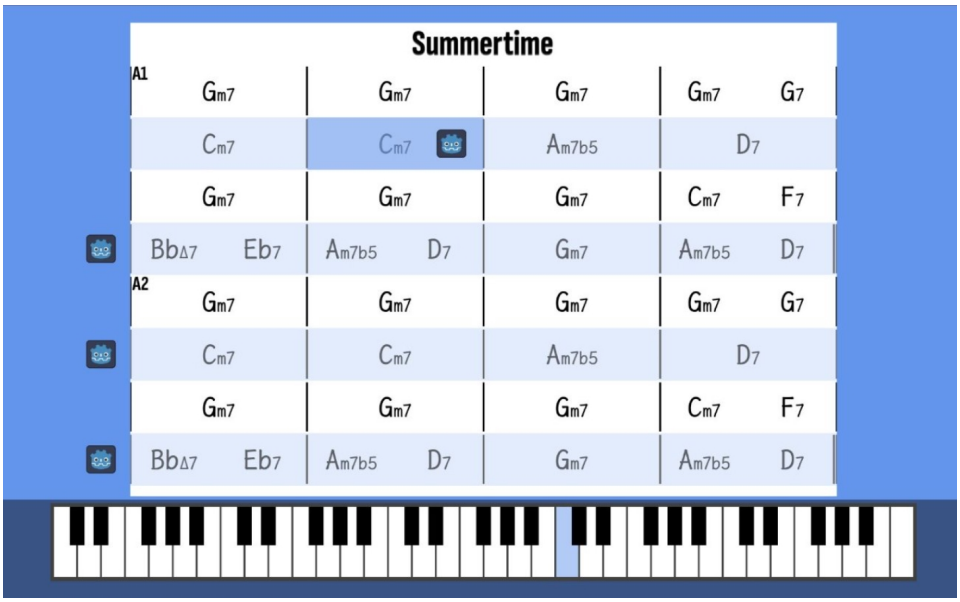


Figure 5.2: A screenshot of MILES during a performance.

5.1.4. Recording and playback backend

In order to reproduce past performances, a recording and playback approach is needed. To achieve this, MILES records all performances by default and allows playback of these recordings. By bringing a recording back into the program, the same visualisation can be seen as during the original performance. Using a screen recorder, exact reproductions of performances can be converted to video files for later use.

5.2. Evaluation

In order to facilitate evaluation of MILES' algorithms outlined in the experimental setup, we have implemented a range of useful features. These are discussed in this section.

5.2.1. Evaluation form

To facilitate the third-party feedback outlined in the experimental design, an evaluation form is needed with which recordings can be listened to (with corresponding visualisations) and can be assessed. We achieve this with an online form that can show the respondent informational text, comparative questions between two recordings, and Likert-scale rating questions of a single recording. Figure 5.3 shows a screenshot of the form, and the full form that was used in our experiments can be seen in Appendix A.

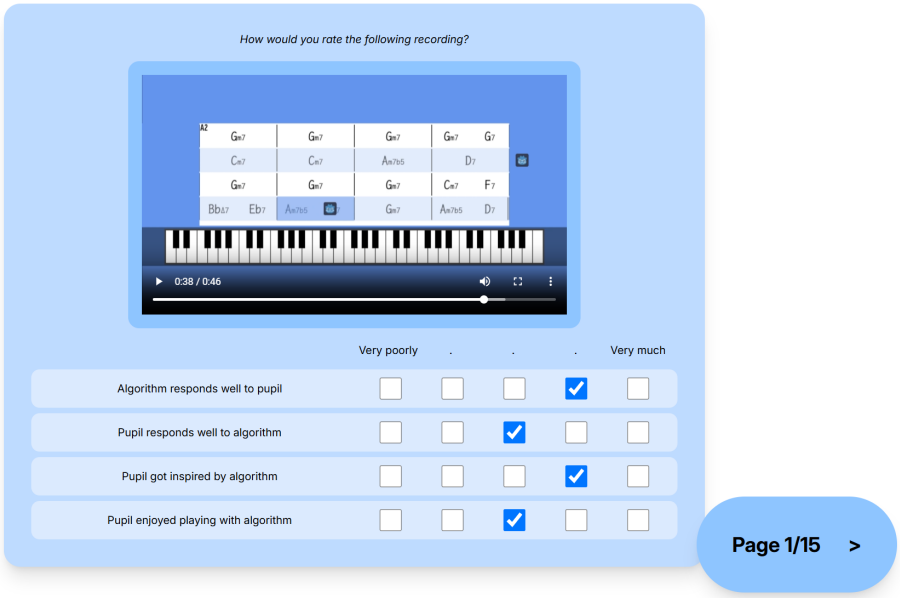


Figure 5.3: A screenshot of the evaluation form, which shows a recording and asks the user to rate it based on several categories.

5.2.2. Analysis framework

To provide the symbolic analysis described in chapter 3.3.3, a different component of the software setup is responsible for loading in recordings and processing them. This component has access to a wide range of symbolic analysis methods for monophonic solo data. As well as symbolic data, this analysis framework has access to the filled-in evaluation forms.

5.3. Implementation

This section describes some implementation details of *MILES*, the workflow for symbolic analysis and the evaluation form.

The core of the improvisation system is developed in the Godot game engine using the C# bindings.. By making the application in Godot, we could make use of its scene and UI system, file management APIs, visualisation rendering functionality, and much more.

As Godot does not support MIDI input and output, we chose to circumvent all Godot audio functionality and create a real-time MIDI system which can take in MIDI input, and facilitates immediate and scheduled MIDI output. The free digital audio workstation (*'DAW'*) Cakewalk handles audio synthesis from these MIDI signals. All music improvisation algorithms used in *MILES* work on top of the recording and playback backend, which are all implemented in C#.

The evaluation form is a website developed in React with TypeScript. Most of this website is procedurally generated via `form.config.ts`, a configuration file which specifies the sections of the form. These sections can be text sections (which support Markdown), rating sections (where a single video is shown with Likert questions) or comparing sections (which ask the user to select one of two shown videos). The website is deployed with Netlify, using NextJS for rendering.

The analysis framework is implemented with a Jupyter Notebook. All symbolic and evaluation form data is stored in a single JSON file, which the notebook has access to, in order to provide the analysis. The JSON file also stores the file paths of the recordings themselves, which the notebook opens and stores in a data structure that allows for querying individual 'fours'. The notebook uses equivalent Python ports of the C# classes used in MILES to ensure that recordings can be opened in the notebook and to provide a similar workflow between working in the two languages.

6

Experiment 1: Initial Comparison

Following the experimental design laid out in chapter 3.3.4, we conduct an experiment that has the main goal of tracking enjoyment and familiarity in improvisation of novice musicians ('pupils') with several algorithms, with evaluation by self-assessment by the pupils themselves, assessment by musicians with a high musical skill level ('experts'), and symbolic analysis methods. This section first describes the configuration and logistics of the experiment, and then presents the results and a discussion of these results.

6.1. Configuration

The experimental design we described in chapter 3.3.4 gives room for choosing an amount of pupils, experts, algorithms, songs and sessions. This section details the configuration of this experiment, by giving these amounts and listing the songs that we use in our improvisation sessions. A diagram visualising this configuration (including the algorithms that ended up being chosen) can be found in Figure 6.1.

We have chosen to ask **3 experts** to participate in the pilot study and third-party evaluation, and **5 pupils** to participate in the improvisation sessions. We decided to carry this experiment out with **3 algorithms**, **3 songs** and **4 sessions**, such that every algorithm-song pair can be evaluated by every pupil for the first three sessions, and the fourth session can have the same configuration as the first.

In order to not burden the experts with an unwieldy long evaluation form and because of the fact that we already ask them to compare the fourth session to the first, we have chosen to have the experts not directly rate the fourth session of performances.

This experiment was carried out with three songs that have a canonical solo in the *Weimar Jazz Database*. For the evaluated performances, we picked George Gershwin's *Summertime* with Sidney Bechet's solo, Jerome Kern's *Long Ago and Far Away* with Chet Baker's solo, and Charlie Parker's *My Little Suede Shoes* with his own solo. For the familiarization stage in each session, we make use of Charlie Parker's *Ornithology* with his own solo.

6.2. Logistics

This experiment needs participants and a location to conduct the one-on-one sessions in. This section describes the logistical setup of the experiments.

Much of the logistics was simplified by working with members from the *Delftse Studenten Jazz Vereniging Groover*, the jazz association of Delft. Groover has many types of members (beginner, novice, intermediate, advanced) that are interested in performing, teaching and learning about jazz. The association has many existing multi-week programs in place (called "Muppet Music" and "Jamfabriek") for beginner musicians to improve their jazz (improvisational) skills.

By letting Groover members participate, we can be certain that pupils have basic knowledge of jazz and are familiar with (trading) jazz solos. Because the experts have helped out with the jazz teaching programs, we can be certain that they know how to grade others' performances and can listen for progress in jazz skills over time.

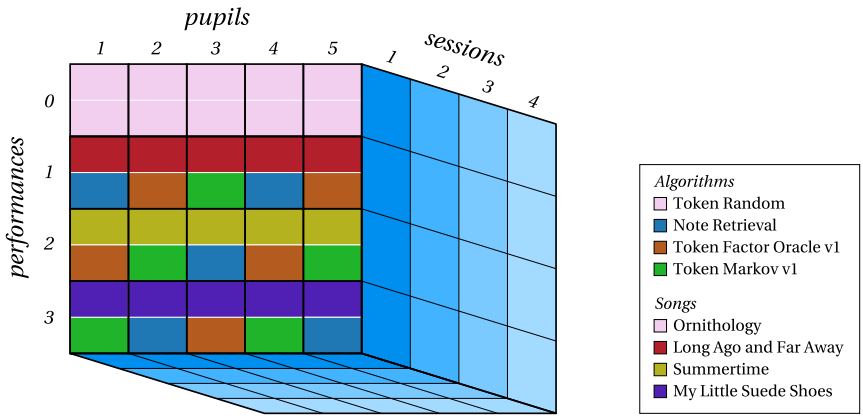


Figure 6.1: A diagram that illustrates the setup of pupils, performances and sessions used for experiment 1.

We were lucky to be able to use the Groover instrument storage room as the location for our experiments; a picture of the setup can be seen in Figure 6.2. Because the association has a weekly gathering on Wednesday evenings, this was the day most sessions took place.

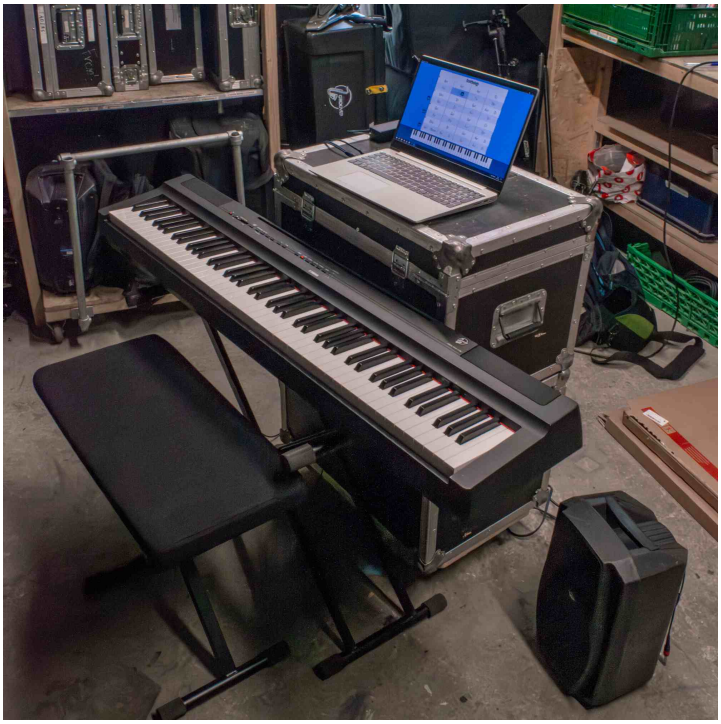


Figure 6.2: The physical setup used in the experiments.

6.3. Results

6.4. Pilot phase and algorithms

The following algorithms were considered in the pilot phase (note: these link to the corresponding descriptions):

- Note Random
- Note Retrieval

- Note Factor Oracle
- Token Random v1
- Token Factor Oracle v1
- Token Markov v1
- Token Shuffle v1

In the pilot phase, the algorithms that utilised tokens were generally seen by the experts as good contenders for algorithms out of the more sophisticated generative methods. Interestingly, the Note Retrieval algorithm was also favoured heavily, due to the sheer quality of musical output (even though the experts were informed of the inner workings of the algorithm). Based on these responses, we selected **Note Retrieval**, **Token Factor Oracle v1** and **Token Markov v1** as the algorithms to be used for the sessions of this experiment. For the familiarization performances, **Token Random v1** is chosen. Due to its relatively poor output and the high tempo of Ornithology, we aim to make the pupil put less effort into their improvisations and instead focus on the trading scheme.

Besides algorithm preference, the experts enjoyed the interaction and visualization of *MILES* and found the system intuitive to use. All of the experts noted that, as piano players, only playing monophonic material (with only one hand) felt unnatural for them.

6.4.1. Self-assessment

In general, the pupils seemed to quite enjoy the process of working with *MILES*. A few pupils struggled with having to play one note at a time, though others seemed to find it less stressful to only have to think about playing with one hand. Even though this took them slightly longer than the experts, the pupils seemed to respond to the different 'traits' of the algorithms. In particular, Note Retrieval for the solo on *My Little Suede Shoes* made some people visibly nervous, and made pupils play shorter, more quickly-paced notes.

With the collected self-reported (musicality, response and enjoyment) ratings of the pupils and experts, we answer the following questions:

Which of the songs was more enjoyable to play with?
Before looking into the algorithms, we consider the songs used in the experiment and compare all ratings of the performances played with these songs. Figure 6.3a shows a summary of the enjoyability ratings that pupils gave the algorithm, grouped per song. We see that the average enjoyability consistently is higher than 3 on a 5-point scale, with no song being rated poorly, and *Long Ago and Far Away* showing least variance.

Which of the algorithms was more enjoyable to play with?
Enjoyability ratings grouped by algorithm, as shown in figure 6.3b, show that the Note Retrieval algorithm is considered the most enjoyable to play with on average. We also see that, although the means of Token Factor Oracle and Token Markov algorithm ratings are close, the Token Factor Oracle ratings have a higher mode than those of Token Markov.

How do pupils view the back-and-forth dynamics of playing with the algorithms?
Figure 6.4 shows the response ratings given per algorithm. As can be seen, the pupils rate their own responsivity higher than that of the algorithms. The distribution of scoring the algorithm with regard to the pupil appears very wide, with a large negative rating given to Token Markov. However, the figure also states that pupils felt they could respond better to Token Markov than the other algorithms.

How did enjoyment of the sessions change over time?
By grouping the enjoyability reporting by session, we get an indication of the fun the pupils had throughout the entire experiment, irrespective of the song or algorithm. Figure 6.3c shows a plot of this grouping. Again, the average enjoyability is in the high range of values on the 5-point scale. We see that the pupils had the most fun in the first session, and after a drop in the second session we see a steady increase back to the value of the first session.

6.4.2. Expert reporting

With the ratings and orderings given by the experts, we aim to find out how the pupils' evaluation of the performances relates to to that of the experts.

Figure 6.5 shows an overview of all average ratings given by pupils and experts, grouped by algorithm. One of the first apparent differences is the low overall ratings of experts relative to the pupils. As well as that, the ratings of pupils are more varied than those of the experts, showing a more clear 'winner' for nearly every question. Figure 6.6 shows a similar chart, but grouped by session, where we see the same lower, less varied, scoring given by experts.

Figure 6.7 shows the relation between pupil-reported ratings and expert-reported ratings. Although the linear fit of the regression line shows a small positive slope, the scatter plot mostly shows a non-linear cloud of points.

As well as rating a selection of performances, we asked the experts to listen to pairs of recordings from the first and last session (where each pair has the same algorithm and song configuration). For each pair, they are asked to order them based on perceived progress. Figure 6.8 shows the correctness of this ordering. One could suspect that if a progress effect can be perceived, this ordering correctness would be better than guessing (a chance of 50%). As can be seen however, only the ordering correctness of the Note Retrieval algorithm is above this number, and the guessing correctness for the Token Factor Oracle algorithm lies far below this.

6.4.3. Symbolic analysis

In the research setup, we set out to apply melodic similarity analysis between successive fours to quantify to what degree pupils may react to the algorithms and vice versa, and to compare this to the human self-reporting data on this question.

Figure 6.9 shows a piano roll view of a single performance. The trading fours interaction is clearly visible, with the pupil taking the first four measures, the algorithm taking the next, et cetera. These recordings provide the symbolic data we use for analysis.

Figure 6.10 shows the average progression of the selected symbolic analyses over the course of a performance. In the analysis, we look for the apparent 'smoothness' of the graph, indicating that the features of each four get carried over to the next four and thus a form of reciprocity and inspiration. For the extrema ratio, a neat line can be seen for the first two algorithms. For Token Markov however, a discrepancy can be found at the start of the performance which shrinks over time.

The graphs for root/third/fifth note share show no visually apparent difference between the different algorithms. The NPVI shows a smooth graph for Token Factor Oracle, a relatively smooth graph for Token Markov, and a graph that shows no apparent human-to-algorithm reciprocity for Note Retrieval.

Next, by converting successive fours into tokens and performing the Levenshtein distance to the resulting strings, we get the data shown in Figure 6.11. Figure 6.11a shows that similarity between pupils and the algorithm is better with the song *Long Ago and Far Away*, indicating that this song might be easier to trade fours over. Figure 6.11b indicates that the Note Retrieval algorithm does indeed respond worse to the pupils than the Token Factor Oracle and Token Markov algorithms. However, the response of the pupil to the algorithm is also considerably worse for the Note Retrieval algorithm. Figure 6.11c shows that the token edit distance decreases from the first session to the second, and then increases slightly over time.

6.5. Discussion

Although the pupils felt that they could respond the best to Token Markov, they also indicated that it responded the worst to them out of the algorithms. This could be due to the fact that it does not simply start out generating its four somewhere along the human played four, like a Factor Oracle model does, but instead takes both the canonical solo and pupil fours into account holistically. As the share of the pupil's fours slowly increases over time during a performance, the style of the algorithm is thus influenced more and more by the pupil. The dynamics of the extrema ratio of the Token Markov model getting closer to that of the pupil over the course of the performance could also be explained by this notion of the Markov model slowly incorporating the pupil's musical material.

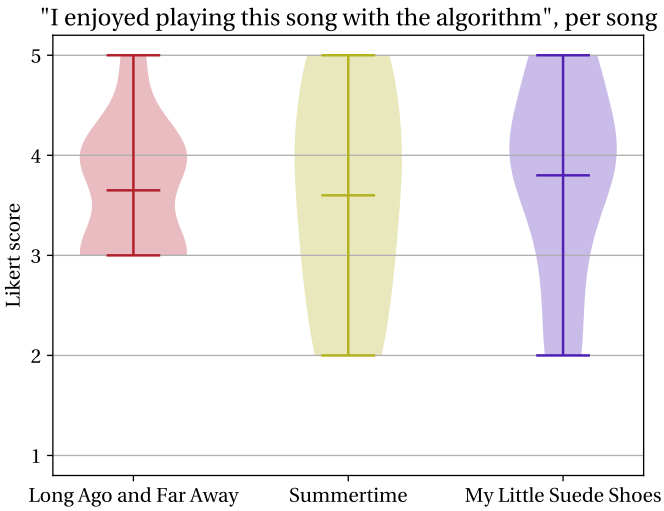
Aside from this anomaly of the extrema ratio in Token Markov performances and a fluctuating NPVI for Note Retrieval, no concrete conclusions can be drawn from using the symbolic analysis

metrics for evaluating the different algorithms.

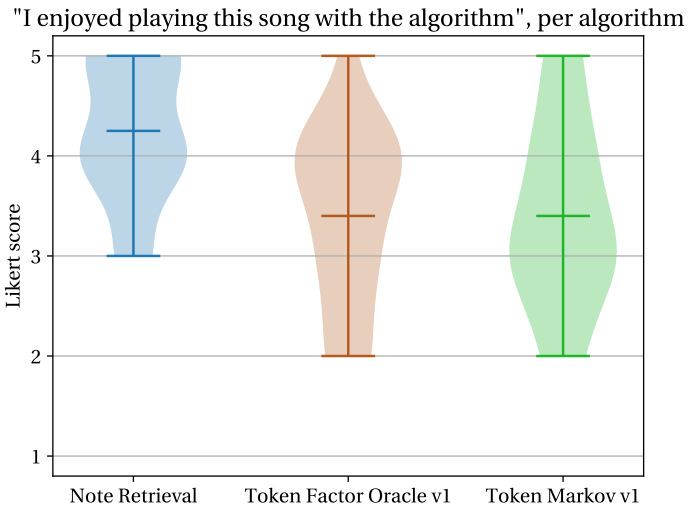
It is also remarkable that, although the Note Retrieval algorithm does not respond to the pupil, pupils prefer this algorithm in terms of its responsive quality, and the enjoyment of their performances with it. This might indicate that music improvisation systems do not necessarily have to take user input into account when generating new content, while still being engaging. The pupils' indication of their own responses to the algorithm did not seem to depend on the algorithm being used, indicating that this does not seem to matter that much.

Enjoyment ratings show that the novelty of working with a music improvisation system causes a large spike in reporting across all criteria. This can also be seen in the correctness of experts' ordering between the first and last session, showing generally poor results regardless of algorithm. Even though enjoyment decreased past the first session, the edit distances show that the reciprocity improved, remaining stagnant beyond the second session. Grouping enjoyment by algorithm, it becomes clear that the same algorithms that pupils indicate respond well to them are the same models that they enjoyed playing with.

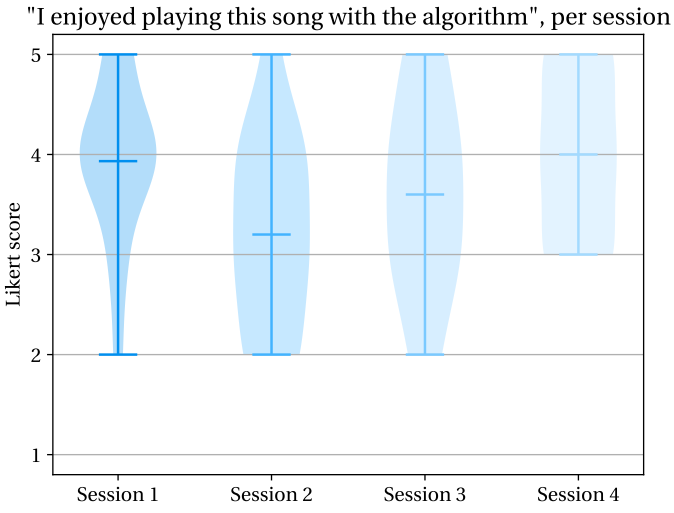
The difference between the results of expert rating and self-reported rating is striking: experts rated the performances quite poorly, with very low differences between the algorithms and sessions, and low correlation with the pupil-reported ratings. In the experimental design, we remarked that pupils have the most experience to gain in working with a music improvisation model. This same contrast in learned experience could be an explanation of the contrast in the pupil-reported ratings.



(a) Song enjoyability



(b) Algorithm enjoyability



(c) Session enjoyability

Figure 6.3: Violin plots with self-reported Likert ratings regarding the statement '*I enjoyed playing this song with the algorithm*', grouped respectively by song (a), session (b) and algorithm (c). Higher is better.

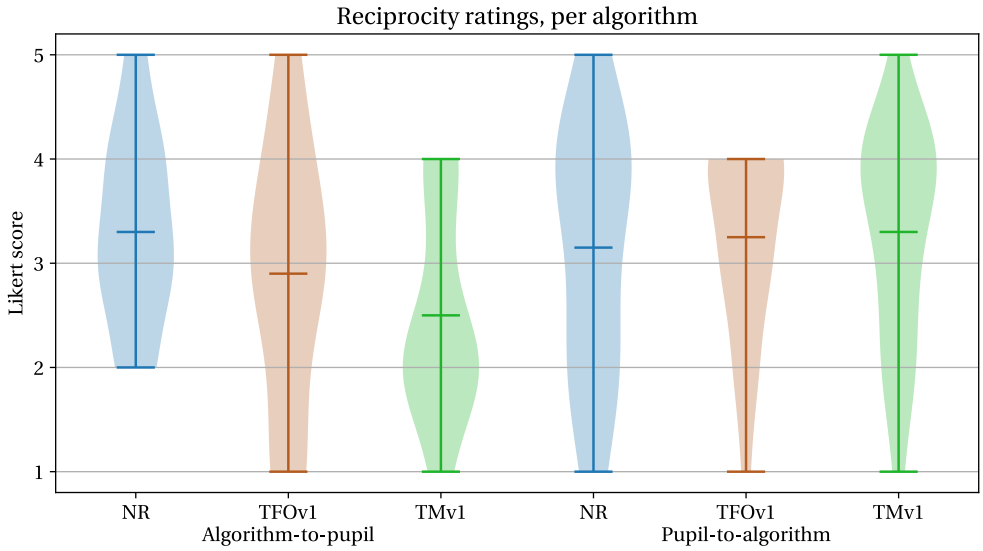


Figure 6.4: Violin plots with self-reported Likert ratings regarding the statements '*I feel that the algorithm could respond well to me*' (algorithm-to-pupil) and '*I feel that I could respond well to the algorithm*' (pupil-to-algorithm), grouped by algorithm. Higher is better.

	Algorithm responds well to pupil		**Pupil responds well to algorithm**		**Pupil got inspired by algorithm**		**Pupil enjoyed playing with algorithm**	
	self	expert	self	expert	self	expert	self	expert
NR	2,4	1,3	**3,2**	**2,4**	2,7	2	3,4	2
TFO	**3,2**	1,7	3,1	2,3	**4,3**	1,8	**4,1**	2,2
TM	2,8	**1,9**	3,1	**2,4**	2,7	**2,2**	3,2	**2,3**

Figure 6.5: A table showing the average self-reported and expert-reported Likert ratings regarding all statements given in the evaluation form, grouped by algorithm. Higher is better.

	Algorithm responds well to pupil		**Pupil responds well to algorithm**		**Pupil got inspired by algorithm**		**Pupil enjoyed playing with algorithm**	
	self	expert	self	expert	self	expert	self	expert
Session 1	2,8	1,9	3,3	2,4	3,5	2,1	3,9	2,4
Session 2	2,5	1,6	3,2	2,4	2,7	2,1	3,2	2,2
Session 3	3,1	1,5	3	2,2	3,4	1,8	3,6	2
Session 4	3,2		3,5		3,2		4	

Figure 6.6: A table showing the average self-reported and expert-reported Likert ratings regarding all statements given in the evaluation form, grouped by session. Higher is better.

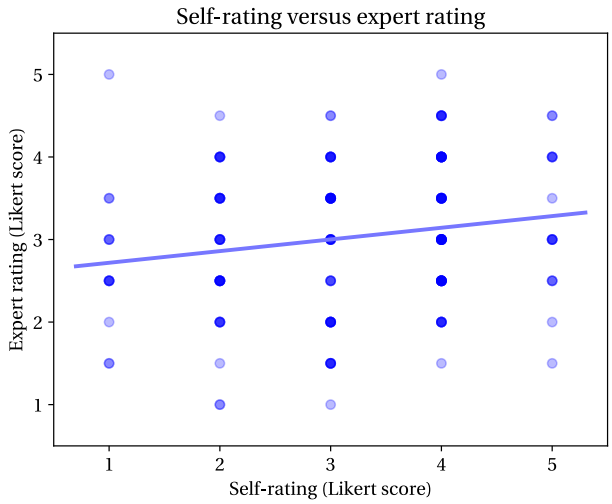


Figure 6.7: A scatter plot showing, for every question of all performances, the pupil's rating and the average expert rating of the same question. A least-squares linear regression is drawn on top.

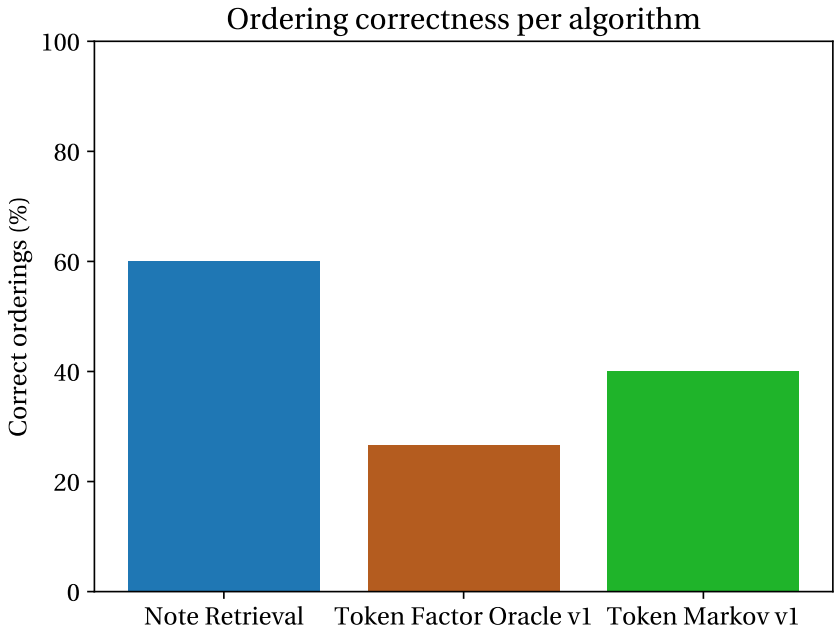


Figure 6.8: A bar chart showing the correctness of expert ordering, grouped by algorithm.

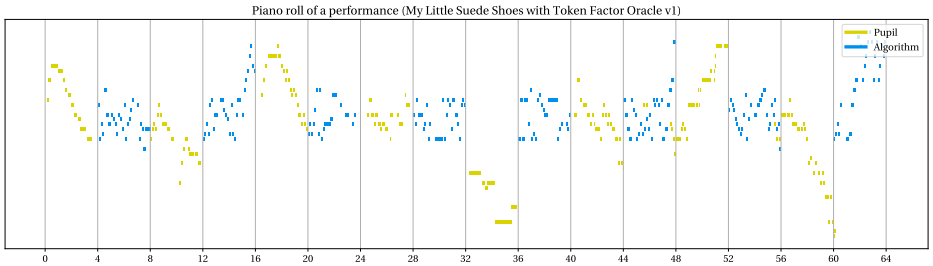
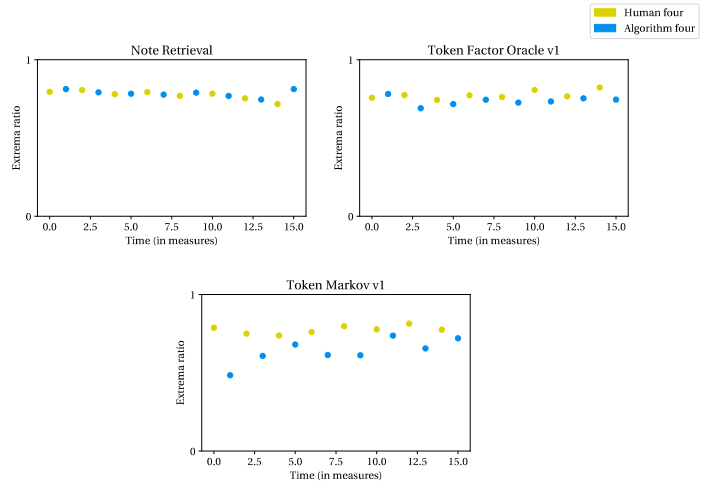
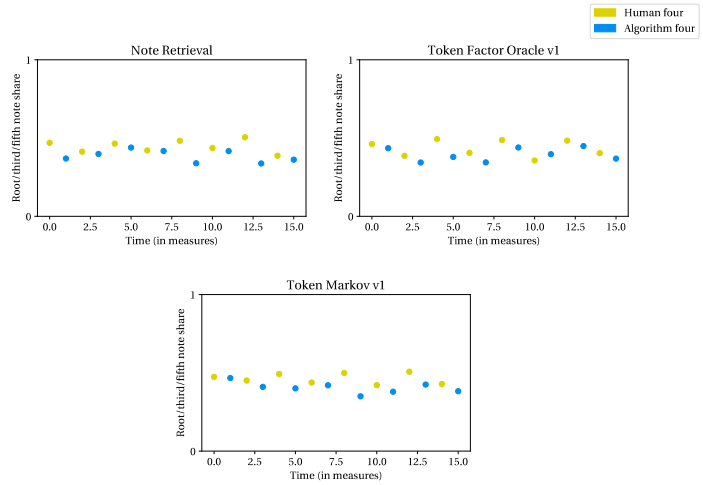


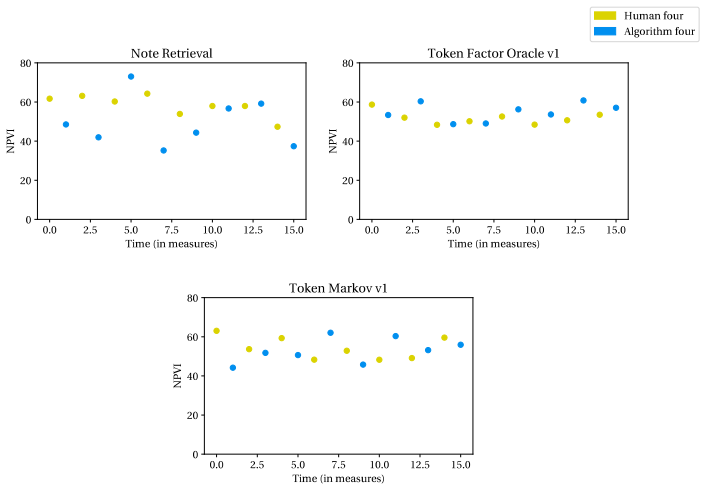
Figure 6.9: A piano roll visualisation of a performance.



(a) Average extrema ratio.

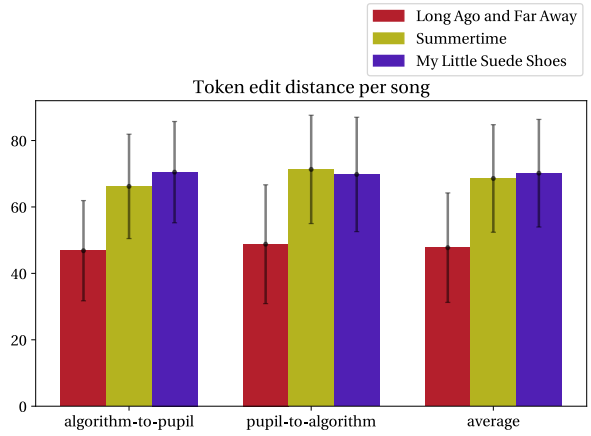


(b) Average root/third/fifth note share.

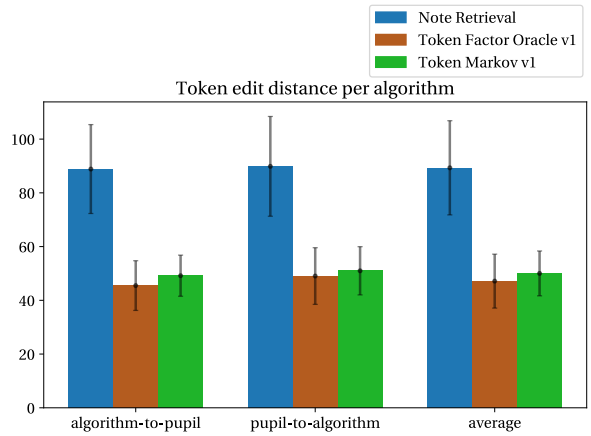


(c) Average NPVI.

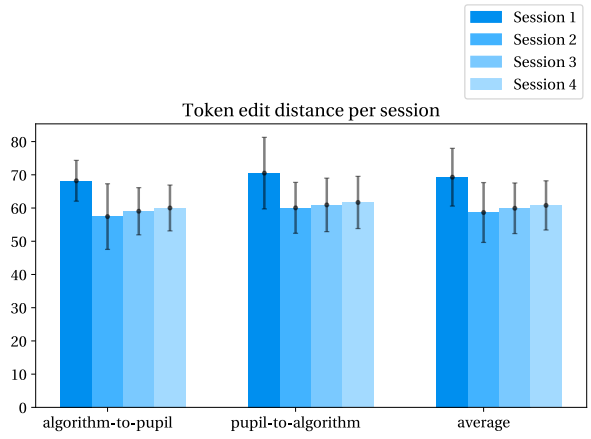
Figure 6.10: Plots respectively showing the average extrema ratio, root/third/fifth note share and NPVI per four over all performances.



(a) Average token edit distances, grouped by song. Lower is better.



(b) Average token edit distances, grouped by algorithm. Lower is better.



(c) Average token edit distances, grouped by session. Lower is better.

Figure 6.11: Average token edit distances of performances, respectively grouped by song, algorithm and session.

7

Experiment 2: Evaluating Iterations

Using the experimental design laid out in chapter 3.3.4, we now conduct a second experiment that has the main goal of carrying out iterative comparison of music improvisation algorithm with novice musicians ('pupils') using evaluation by self-assessment by the pupils themselves, peer assessment from the other pupils, and symbolic analysis methods.

This section first describes the iteration of the algorithms between the first and second experiment. Next, the setup and logistics of the experiment are detailed, and finally the results and a discussion of these results are presented.

7.1. Iteration of algorithms

Because quite a bit of time passed in-between the first and second experiment, and due to the fact that we received plenty of feedback from the pupils playing with the selected algorithms, this gave us a lot of room for experimentation and iteration upon the Token Factor Oracle v1 and Token Markov v1 algorithms. This section shortly discusses these changes that were made; a more thorough explanation of the second iteration of algorithms can be found in chapter 4.

Token Factor Oracle v1 has been iterated on by firstly using the second iteration of our tokenization strategy (described in appendix D). Secondly, this algorithm does not consider single tokens as 'characters' of the factor oracle's 'string', but instead uses segments of 8 tokens as a single 'character'.

Token Markov v1 experienced a similar iteration between the first and second experiment as Token Factor Oracle v1. Aside from using the new tokenization strategy, we happened to come across an implementation of the variable-order Markov model of Pachet's *The Continuator* [15]. We substituted the fixed-order Markov model of Token Markov v1 with a port of this model.

7.2. Configuration

The experimental design we described in chapter 3.3.4 gives room for choosing an amount of pupils and algorithms. This section details the configuration of this experiment, by giving these amounts and listing the songs that we use in our improvisation sessions. A diagram visualising this configuration (including the algorithms that ended up being chosen) can be found in Figure 7.1.

In this experiment, we have chosen to ask **3 pupils** to participate in the improvisation sessions and peer evaluation. These participants consisted of three pupils that also participated in the first experiment. Following the experimental design, we only conduct **1 session** of performances. These performances will take place with **4 algorithms**, namely:

- Token Factor Oracle v1
- Token Markov v1
- Token Factor Oracle v2
- Token Markov v2

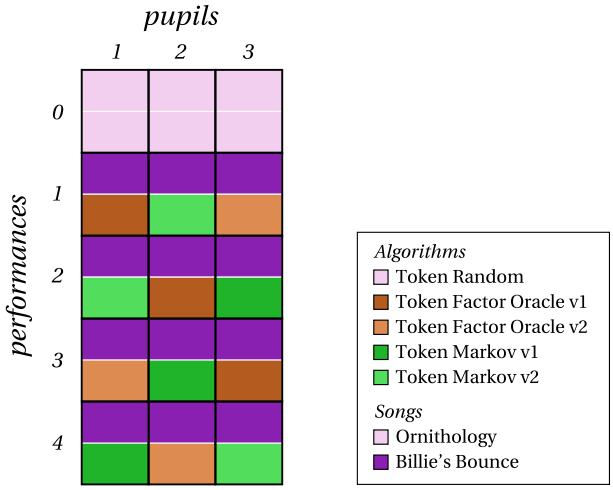


Figure 7.1: A diagram that illustrates the setup of pupils, performances and sessions used for experiment 2.

All performances will take place over Charlie Parker's *Billie's Bounce* with his own solo as the canonical material. Similar to the previous experiment, we make use of Charlie Parker's *Ornithology* with his own solo for the familiarization stage of each session.

7.3. Logistics

This experiment uses the same physical setup (shown in figure 6.2) and logistical considerations. The experiment was again carried out at the Groover instrument storage room, on a Wednesday evening.

7.4. Results

During the performances of this experiment, it was quite clear that the improvements of the algorithms were noticeable by the pupils. The fact that the new algorithms sometimes carried their improvisation over the fourth barline was a pleasant surprise for some of the pupils, and this made them more playful in their improvisation. The peer feedback round also proceeded smoothly, with the online form being sent out the day after the performances, and most of the pupils handing in their results on that same day.

7.4.1. Self-reporting

With the self-reporting data, we aim to answer a few questions:

Which of the songs was more enjoyable to play with?
Figure 7.2 shows the enjoyability scores per algorithm. Similarly to the first experiment, Token Factor Oracle v1 is rated higher than Token Markov v1. As for the improved versions of the algorithms, it is very clear to see that the newer versions of the algorithms rank higher in enjoyment than their respective older versions.

How do pupils view the back-and-forth dynamics of playing with the algorithms?
In Figure 7.3, both reciprocity scores can be seen for the four algorithms. A similar progression between the four algorithms can be seen as the enjoyability scores: the second versions of the algorithm rank higher than the first. Interestingly, it does not seem to matter too much what algorithm the pupil uses when it comes to their own ability to respond: only Token Markov v1 shows some variance in scoring, but the means lie pretty close to one another.

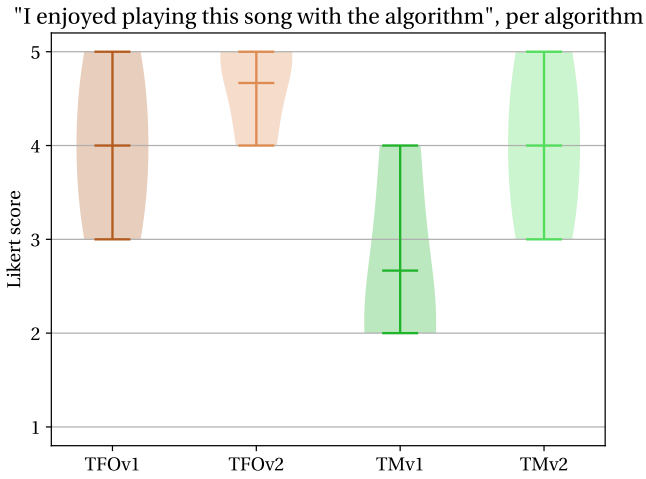


Figure 7.2: Violin plots with self-reported Likert ratings regarding the statement '*I enjoyed playing this song with the algorithm*', grouped by algorithm. Higher is better.

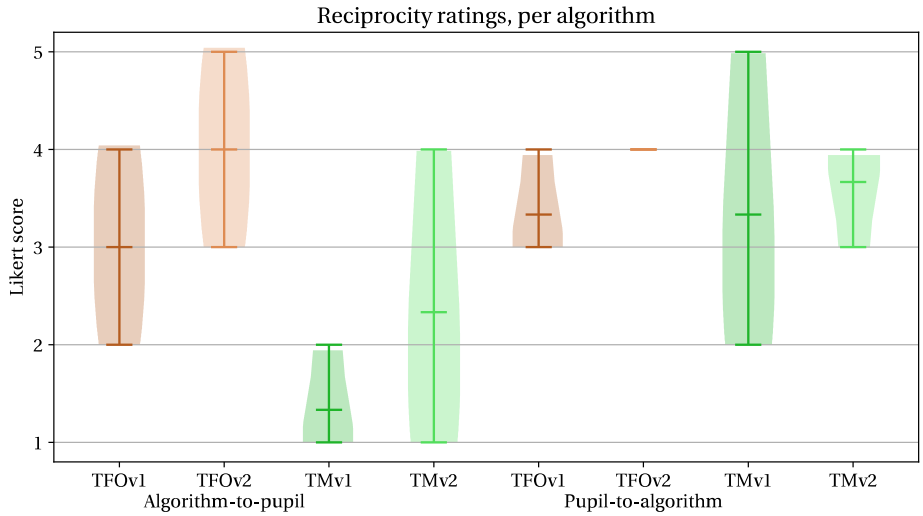


Figure 7.3: Violin plots with self-reported Likert ratings regarding the statements '*I feel that the algorithm could respond well to me*' (algorithm-to-pupil) and '*I feel that I could respond well to the algorithm*' (pupil-to-algorithm), grouped by algorithm. Higher is better.

7.4.2. Peer reporting

As given in the experimental design, by comparing the ratings given by a pupil to those given by their peers, we aim to find out the differences between self-evaluation and third-party evaluation.

Figure 7.4 shows the scores given by self-reporting next to the peer reporting, per algorithm. Peers and pupils seem to agree that Token Factor Oracle v2 scores the highest in all categories. Again, an apparent improvement is present between the scores of the first and second version of both algorithms. The table appears to indicate a high sense of agreement between the pupils and their peers, especially for the first question ("Algorithm responds well to pupil"), which also shows the highest variance. All other questions exhibit less variance between the algorithms in both self-assessed and peer-assessed ratings. The highest discrepancy between self-assessed and peer-assessed ratings can be found in the last question ("Pupil enjoyed playing with algorithm"), with peers rating performances ower than the pupils themselves. The scatter plot in Figure 7.5 shows a small amount of correlation between all scores given by the pupils and their respective peer scores.

	Algorithm responds well to pupil			Pupil responds well to algorithm			Pupil got inspired by algorithm			Pupil enjoyed playing with algorithm		
	self	peer	combined	self	peer	combined	self	peer	combined	self	peer	combined
TFOv1	3	3	3	3,3	2,5	2,8	3,3	1,7	2,2	4	2,3	2,9
TFOv2	**4**	**3,3**	**3,6**	**4**	**3,5**	**3,7**	**3,7**	**2,3**	**2,8**	**4,7**	**2,7**	**3,3**
TMv1	1,3	1	1,1	3,3	2,5	2,8	2	1,8	1,9	2,7	1,8	2,1
TMv2	2,3	1,5	1,8	3,7	2,7	3	2,3	2	2,1	4	1,7	2,4

Figure 7.4: A table showing the average self-reported and peer-reported Likert ratings regarding all statements given in the evaluation form, grouped by algorithm. Higher is better.

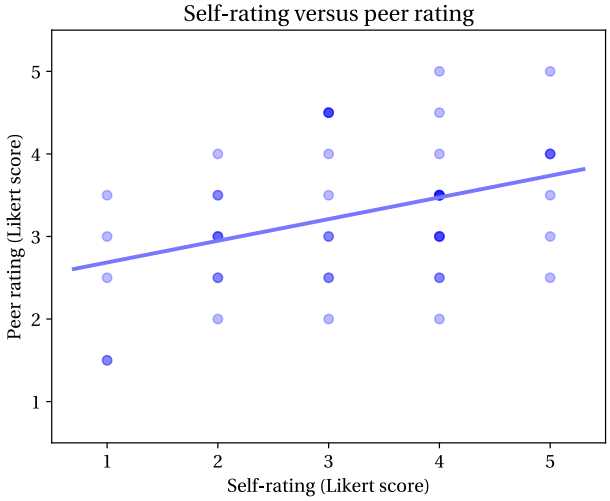


Figure 7.5: A scatter plot showing, for every question of all performances, the pupil's rating and the average peer rating of the same question. A least-squares linear regression is drawn on top.

7.4.3. Symbolic analysis

A similar symbolic analysis procedure will be carried as the first experiment, using the same three global features (extrema ratio, root/third/fifth note share, NPVI). For illustrative purposes, Figure 7.6 shows a piano roll view of the symbolic data contained in a performance carried out in this experiment, using Token Factor Oracle v2.

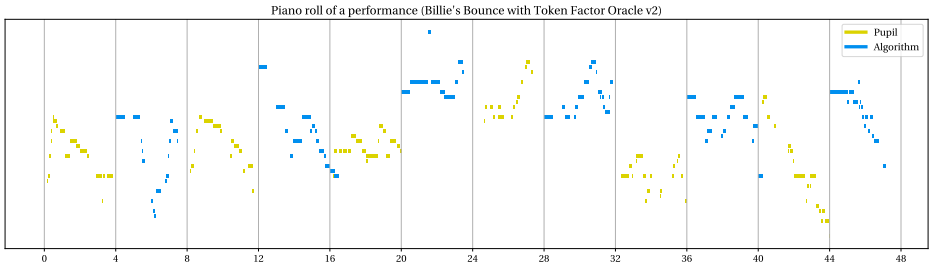


Figure 7.6: A piano roll visualisation of a performance.

Figure 7.7a shows the progression of the extrema ratio for every algorithm. The data of Token Factor Oracle v1 and Token Markov v1 looks similar to the data obtained in experiment 1 (see Figure 7.7a): Token Factor Oracle v1 looks 'steady', and Token Markov v1 needs some time to 'catch up' with the metrics of the human-played fours. Strikingly, this 'catching up' motion, which we concluded to be inherent to Markov models in experiment 1, is not visible with the Token Markov v2 algorithm. Figure 7.7b shows the progression of the root/third/fifth note share. Besides a few anomalies in the graphs for the Factor Oracle algorithms, we see no noteworthy features in these graphs. Figure 7.7c shows the progression of the NPVI for the four algorithms. In contrast to the findings of the first ex-

periment, the Token Factor Oracle v1 graph shows a considerable discrepancy between the human- and algorithm-played fours. This discrepancy is less visible for Token Markov v1, and even less so for the improved algorithms.

Figure 7.8 shows the token edit distances per algorithm. As can be seen in the previous experiment, the edit distances of the Token Factor Oracle and Token Markov algorithms lie very close to one another. Due to the low amount of samples, the variance is slightly higher than the previous experiment. The figure shows an advantage of the Token Markov models, whereas the Token Factor Oracle algorithm was slightly better in the first experiment.

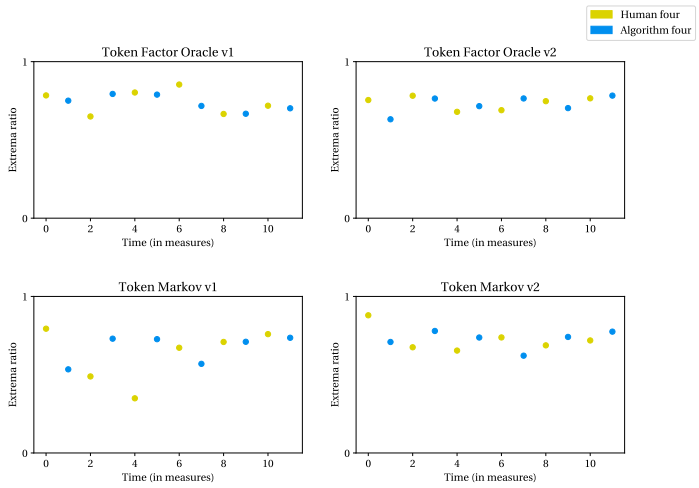
7.5. Discussion

In our results, we see that when compared to the self-assessment peers are slightly pessimistic about some qualities of improvisation, but not overly so. This can be explained by the fact that all pupils have similar improvisational experience, and the same amount of experience in working with *MILES*. This effectively levels their frame of reference and level of expectations. The consensus can be most directly seen for the algorithm-to-pupil reciprocity score.

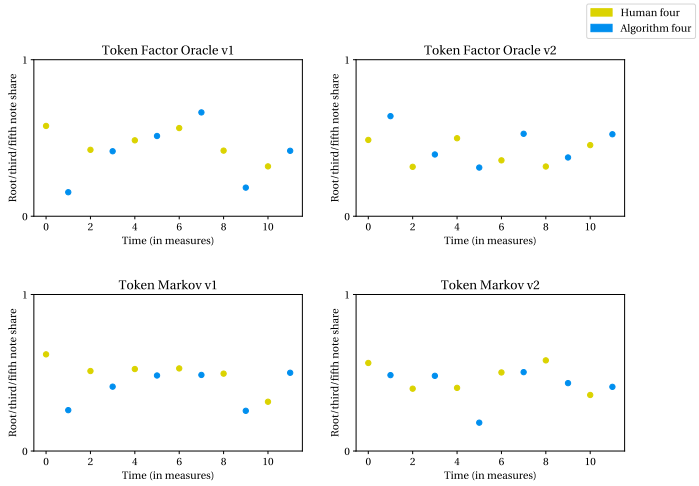
Both in self-reported and peer scoring, we see a preference for Token Factor Oracle v2 across the board. We attribute this to the higher musical stability that generating the solos in larger 'chunks' offers. The slight preference of Token Markov v2 over Token Markov v1 can similarly be attributed to the fact that the variable-order Markov model used in the second iteration has a higher 'maximal order' than the fixed-order Markov model used in the first iteration.

Similarly to the first experiment, the algorithms that the pupils found the most enjoyable were also found to respond best to them. When it came to the quality of their own response, the algorithm did not seem to matter much to the pupils.

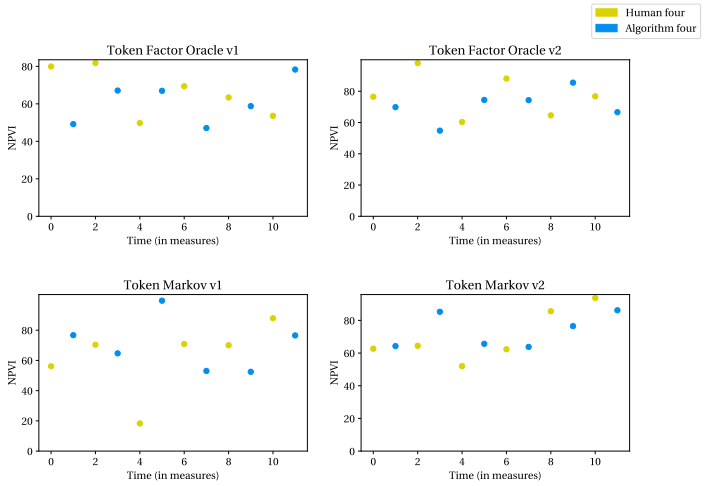
Comparing the symbolic analyses of Token Factor Oracle v1 and Token Markov v1 along both experiments gives a lot of inconsistencies, indicating that interpreting these metrics proves to be difficult. The edit distances of the two prior algorithms align similarly in this experiment as in the first, but the fact that they are nearly equal does not allow for a very comparison.



(a) Average extrema ratio.



(b) Average root/third/fifth note share.



(c) Average NPVI.

Figure 7.7: Plots respectively showing the average extrema ratio, root/third/fifth note share and NPVI per four over all performances.

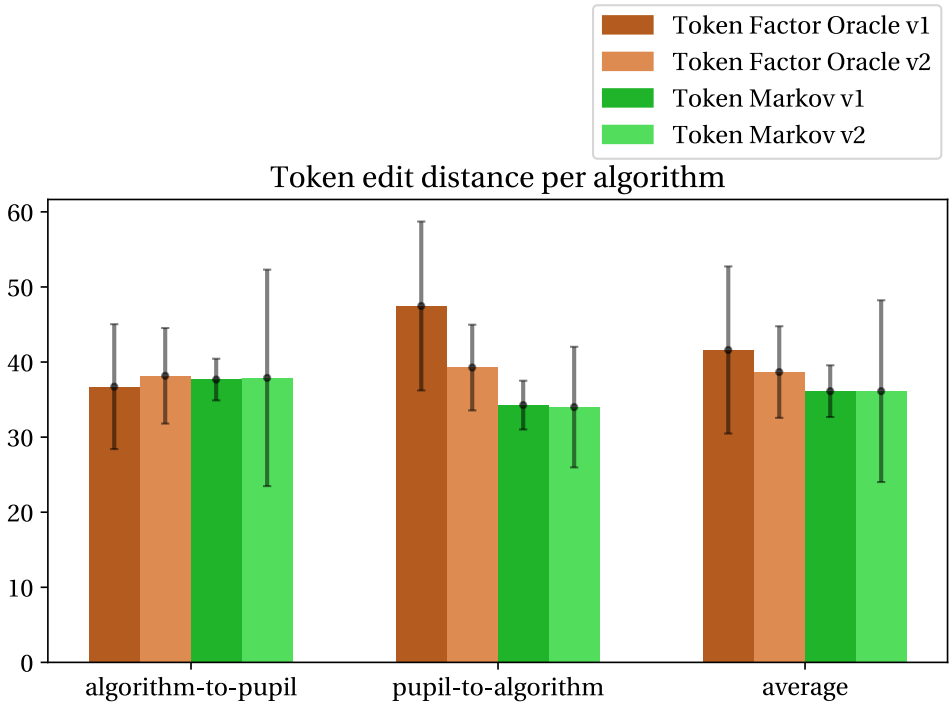


Figure 7.8: Average token edit distances, grouped by algorithm. Lower is better.

8

Conclusions and Future Work

8.1. Conclusions

In this research, we aim to answer the following question:

'To what extent can algorithmic music improvisation systems facilitate a trading fours interaction with a novice musician on their own?'

In chapter 1, we divided this question into several sub-questions. In this chapter, we aim to formulate an answer to the main research question by answering these sub-questions.

2. *'How can familiarity and enjoyment in improvisation be measured for novices over time?'*

In chapter 6, we show that familiarity and enjoyment in improvisation can be measured in several different ways. We find that reciprocity and enjoyment are indeed two different factors of improvisation, as reciprocity is not necessarily needed for an enjoyable improvisation interaction. In our experiments, we see a fairly immediate sense of engagement and we find that self-assessed enjoyment ratings may be subject to a large novelty effect, which might skew results when comparing to later sessions. Although we see a dip after this first session, our musical similarity metric gave decent reciprocity results after this first session, indicating that pupils did indeed become more familiar with the improvisation interaction.

3. *'How can iterative comparison of music improvisation algorithms be carried out?'*

When it comes to measuring enjoyment and reciprocity of iterations of results, chapter 7 shows us that a large-scale experiment is not necessary to get relatively conclusive results. Although our musical similarity metric reflected our intuition of the potential reciprocity of the different algorithms, we find that these differences need to be very large for similarity analysis to give conclusive results.

4. *'What kinds of algorithms and evaluation techniques are best suited for an improvisation interaction between improvisation algorithms and novice musicians on their own?'*

When it comes to evaluating an improvisation interaction between improvisation algorithms and novice musicians on their own, we find that self-assessed enjoyment gives the most conclusive results across our two experiments. However, when self-assessed data is not available, our results in chapter 7 show peer evaluation to be closest to self-assessed opinions of all analysis techniques, being especially clear in response quality of the algorithm.

As for the actual algorithms used in the experiments, we find that Note Retrieval was received very positively, though the question still stands if improvising with the same canonical solo multiple times

still yields the same results. As for the more 'intricate' algorithms we considered, we conclude that pupils had the most fun with the factor oracle-based algorithms, and often find these models to react to them nicely. We are especially proud of our Token Factor Oracle v2 algorithm.

In conclusion, we have presented an experimental setup where multiple algorithms may be evaluated in different contexts. By taking inspiration from existing music improvisation algorithms and existing musical (improvisation) evaluation techniques, we find that although different algorithms functionally express different qualities of the improvisation interaction, these algorithms are in fact suited for working with novice musicians on their own.

8.2. Future Work

The experimental setup we propose has potential for future work, from altering the configuration of our experiments to extending and enhancing *MILES*' capabilities. This chapter details these possible ventures for further research.

8.2.1. Experiments

In the first experiment, we found that when we asked experts to order pupils' recordings from the first and last session, the experts often incorrectly perceived the last session as the first. We believe that this occurred due to the experts picking the recording they liked best to be of the first session, where enthusiasm and novelty was at its highest with the pupils. In order to potentially change the results and better balance novelty, we propose to extend the familiarisation efforts in the experiments to not just include a familiarisation performance at the start of each session, but to also include an entire familiarisation session that is not considered for any analysis.

Due to the constraints of this research, the number of participants of the experiments we conducted was relatively low. We believe that involving more pupils and experts would give a higher quality distribution of opinions, which allows for better analysis and more general, better-founded, conclusions. Moreover, combining expert and peer assessment in a single experiment could potentially give an explanation of the large difference in self-reporting correlation that we found in our first experiment.

Aside from increasing the amount of participants, we believe that more insights can be gained from looking at enjoyment over a longer time span. A possible plan for a new experiment could involve looking at a small group of pupils improvising with *MILES* over the course of a few months, to research i.e. reciprocity and inspiration over time. This can then of course be combined with an additional layer of peer feedback.

Direct feedback is often part of a learning experience, but was lacking from our research due to the intent of the research focusing on musicians that are on their own, and our lack of expertise in music teaching research. Nonetheless, we believe that an additional notion of teaching could result in a more didactically sound familiarisation and learning experience. For instance, each pupil could be assigned an expert that gives written feedback based on their performances. Analysing this feedback over time could give an indication of progression in trading fours abilities. Another potentially interesting study could involve splitting pupils into a group that only partakes in 'real-life' trading fours, and another that solely works with *MILES*, to compare their learning progress.

In this research, we have not proved that familiarisation with *MILES* correlates with familiarisation of 'real life' jazz improvisation; we acknowledge a didactic angle is needed to show the similarities, which we are not in a position to test. We of course welcome further research to explore this, and in the meantime we make an educated guess that the skills taught by playing with *MILES* transfer over to a traditional trading fours interaction.

8.2.2. *MILES*

When scaling up research, it could prove beneficial to have pupils work with *MILES* without a researcher present. While it is possible to have pupils set up and run *MILES* by themselves, the implementation currently has a lot of room for improvement in usability. For instance, the installation procedure of the system is not really suited for a non-technical audience, requiring the user to install many additional programs and build the system from the C# source code. Getting rid of Cakewalk (the digital audio workstation responsible for generating audio) and MIDI loopback software as de-

pendencies, and packaging *MILES* as a simple, potentially cross-platform, executable could eliminate the need for a researcher to be present during improvisation sessions.

The modular nature of *MILES* allows the system to be extended with many more musical generation algorithms. The quality of the trading fours interaction with *MILES* heavily depends on which algorithm is used. We present a selection of algorithms which have been evaluated as enjoyable to play with and (to some degree) respond to the input of the novice musician. However, we still see room for improvement in the level of reciprocity of the algorithms, though we believe that this does not impact the enjoyment above a certain level.

A rudimentary *Token Neural Net* algorithm is present in *MILES*, but was deemed not musically interesting enough to be used in any of the experiments. Improvement of this algorithm and implementation of additional algorithms could potentially give interesting results for improvisation. The current implementation of *MILES* works by executing the algorithm responsible for generating four measures at the start of these four measures. In order to support more compute-intensive algorithms and lower the hardware requirements of the system, this code can be reworked to support more concurrency. This could allow the system to (partly) generate the four measures ahead of time, or to allow an algorithm to refine its musical output as it is being played.

A

Form Overview

This appendix contains all (self-evaluation, expert evaluation and peer evaluation) question forms that were used in the two experiments.

A.1. Informed consent form

Before participants were asked to do anything in our research, they were asked to fill in and sign informed consent forms, adhering to standard-practice TU Delft human research ethics. Figures A.1a to A.1d show the pages of the informed consent forms for the experts and pupils of experiment 1, and Figures A.2a to A.2c show the informed consent forms for the pupils of experiment 2.

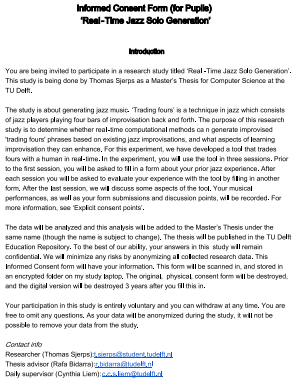
A.2. Self-evaluation form

Self-evaluation was carried out with paper forms throughout the two experiments, with one evaluation being used per session. At the top, some basic info is filled in (the pupil's ID, the session number and the current date and time). Next, for every performance, a field for initial thoughts and the four Likert-scored questions are shown. At the end of the session, a field for closing thoughts is present.

The forms themselves can be seen in Figures A.3 and A.4, with the difference being the fact that the first experiment consisted of three performances per session, whereas the second experiment consisted of four performances.

A.3. Third-party evaluation form

As described in chapter 5, third-party evaluation was carried out with an online evaluation form. This form consists of an introductory section (found in Figures A.5 and A.6 for experiment 1 and 2 respectively), sections where recordings of two different sessions must be ordered based on perceived progress (found in Figure A.7), and sections where recordings are rated along the same lines as the self-evaluation form (found in Figure A.8).



Informed Consent Form (for Pupils)
'Real-Time Jazz Solo Generation'

Introduction

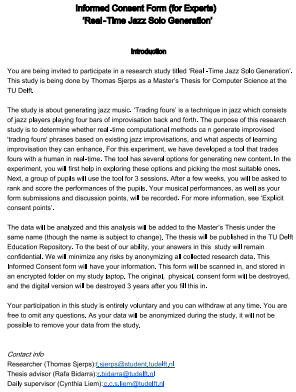
You are being invited to participate in a research study titled 'Real-Time Jazz Solo Generation'. This study is being done by Thomas Sjerps as a Master's Thesis for Computer Science at the TU Delft.

The study is about generating jazz music. 'Trading fours' is a technique in jazz which consists of jazz players playing four bars of improvisation back and forth. The purpose of this research study is to determine whether real-time computational methods ca n generate improvised 'trading fours' phrases based on existing jazz improvisations, and what aspects of learning improvisation they can enhance, For this experiment, we have developed a tool that trades fours with a human in real-time. In the experiment, you will use the tool in three sessions. Prior to the first session, you will be asked to fill in a form about your prior jazz experience. After each session you will be asked to evaluate your experience with the tool by filling in another form, After the last session, we will discuss some aspects of the tool. Your musical performances, as well as your form submissions and discussion points, will be recorded. For more information, see 'Explicit consent points'.

The data will be analyzed and this analysis will be added to the Master's Thesis under the same name (though the name is subject to change), The thesis will be published in the TU Delft Education Repository. To the best of our ability, your answers in this study will remain confidential. We will minimize any risks by anonymizing all collected research data. This Informed Consent form will have your information, This form will be scanned in, and stored in an encrypted folder on my study laptop, The original, physical, consent form will be destroyed, and the digital version will be destroyed 3 years after you fill this in.

Your participation in this study is entirely voluntary and you can withdraw at any time. You are free to omit any questions. As your data will be anonymized during the study, it will not be possible to remove your data from the study.

Contact info
Researcher (Thomas Sjerps) t.sjerps@student.tudelft.nl
Thesis advisor (Rafa Bidarra) r.bidarra@tudelft.nl
Daily supervisor (Cynthia Liem) c.c.s.liem@tudelft.nl



Informed Consent Form (for Experts)
'Real-Time Jazz Solo Generation'

Introduction

You are being invited to participate in a research study titled 'Real-Time Jazz Solo Generation'. This study is being done by Thomas Sjerps as a Master's Thesis for Computer Science at the TU Delft.

The study is about generating jazz music. 'Trading fours' is a technique in jazz which consists of jazz players playing four bars of improvisation back and forth. The purpose of this research study is to determine whether real-time computational methods ca n generate improvised 'trading fours' phrases based on existing jazz improvisations, and what aspects of learning improvisation they can enhance, For this experiment, we have developed a tool that trades fours with a human in real-time. The tool has several options for generating new content. In the experiment, you will first help in exploring these options and picking the most suitable ones, Next, a group of pupils will use the tool for 3 sessions, After a few weeks, you will be asked to rank and score the performances of the pupils. Your musical performances, as well as your form submissions and discussion points, will be recorded. For more information, see 'Explicit consent points'.

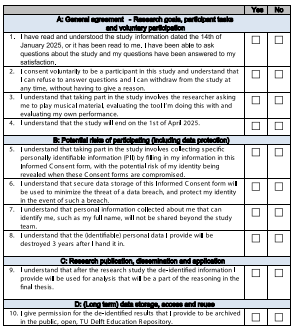
The data will be analyzed and this analysis will be added to the Master's Thesis under the same name (though the name is subject to change), The thesis will be published in the TU Delft Education Repository. To the best of our ability, your answers in this study will remain confidential. We will minimize any risks by anonymizing all collected research data. This Informed Consent form will have your information, This form will be scanned in, and stored in an encrypted folder on my study laptop, The original, physical, consent form will be destroyed, and the digital version will be destroyed 3 years after you fill this in.

Your participation in this study is entirely voluntary and you can withdraw at any time. You are free to omit any questions. As your data will be anonymized during the study, it will not be possible to remove your data from the study.

Contact info
Researcher (Thomas Sjerps) t.sjerps@student.tudelft.nl
Thesis advisor (Rafa Bidarra) r.bidarra@tudelft.nl
Daily supervisor (Cynthia Liem) c.c.s.liem@tudelft.nl

(a) The first page for pupils, showing introductory information.

(b) The first page for experts, showing introductory information.

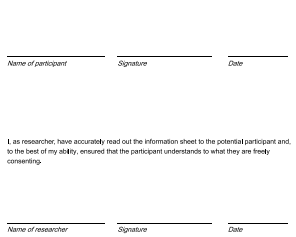


Explicit consent points

Please tick the appropriate boxes.

	Yes	No
A: General agreement – Research goals, participant tasks and voluntary participation		
1. I have read and understood the study information dated the 14th of January 2025, or it has been read to me, I have been able to ask questions about the study and my questions have been answered to my satisfaction.	☐	☐
2. I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason.	☐	☐
3. I understand that taking part in the study invokes the researcher asking me to play musical material, evaluating the tool I'm doing this with and evaluating my own performance.	☐	☐
4. I understand that the study will end on the 1st of April 2025.	☐	☐
B: Potential risks of participating (including data protection)		
5. I understand that taking part in the study invokes collecting specific personally identifiable information (PII) by filling in my information in this Informed Consent form, with the potential risk of my identity being revealed when these Consent forms are compromised.	☐	☐
6. I understand that secure data storage of this Informed Consent form will be used to minimize the threat of a data breach, and protect my identity in the event of such a breach.	☐	☐
7. I understand that personal information collected about me that can identify me, such as my full name, will not be shared beyond the study team.	☐	☐
8. I understand that the (identifiable) personal data I provide will be destroyed 3 years after I hand it in.	☐	☐
C: Research publication, dissemination and application		
9. I understand that after the research study the de-identified information I provide will be used for analysis that will be a part of the reasoning in the final thesis.	☐	☐
D: (Long term) data storage, access and reuse		
10. I give permission for the de-identified results that I provide to be archived in the public, open, TU Delft Education Repository.	☐	☐

(c) The second page for both pupils and experts, showing a consent point checklist.



Signatures

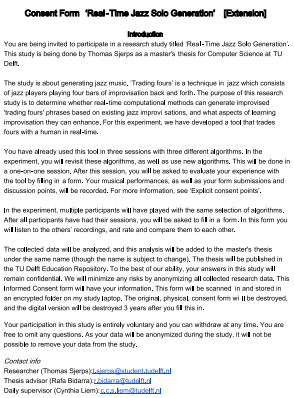
Name of participant	*Signature*	*Date*

I, as researcher, have accurately read out the information sheet to the potential participant and, to the best of my ability, ensured that the participant understands to what they are freely consenting.

Name of researcher	*Signature*	*Date*

(d) The third page for both pupils and experts, showing regions for signatures.

Figure A.1: The informed consent forms used for pupils and experts in experiment 1.



Consent Form 'Real-Time Jazz Solo Generation' [Extension]

Introduction

You are being invited to participate in a research study titled 'Real-Time Jazz Solo Generation'. This study is being done by Thomas Sjerps as a master's thesis for Computer Science at TU Delft.

The study is about generating jazz music, 'Trading fours' is a technique in jazz which consists of jazz players playing four bars of improvisation back and forth. The purpose of this research study is to determine whether real-time computational methods can generate improvised 'trading fours' phrases based on existing jazz improvisations, and what aspects of learning improvisation they can enhance. For this experiment, we have developed a tool that trades fours with a human in real-time.

You have already used this tool in three sessions with three different algorithms. In the experiment, you will revisit these algorithms, as well as use new algorithms. This will be done in a one-on-one session. After this session, you will be asked to evaluate your experience with the tool by filling in a form. Your musical performances, as well as your form submissions and discussion points, will be recorded. For more information, see 'Explicit consent points'.

In the experiment, multiple participants will have played with the same selection of algorithms. After all participants have had their sessions, you will be asked to fill in a form. In this form you will listen to the others' recordings, and rate and compare them to each other.

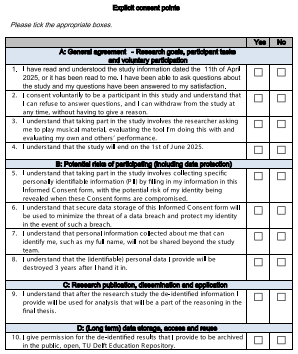
The collected data will be analyzed, and this analysis will be added to the master's thesis under the same name (though the name is subject to change). The thesis will be published in the TU Delft Education Repository. To the best of our ability, your answers in this study will remain confidential. We will minimize any risks by anonymizing all collected research data. This Informed Consent form will have your information. This form will be scanned in and stored in an encrypted folder on my study laptop. The original, physical, consent form will be destroyed, and the digital version will be destroyed 3 years after you fill this in.

Your participation in this study is entirely voluntary and you can withdraw at any time. You are free to omit any questions. As your data will be anonymized during the study, it will not be possible to remove your data from the study.

Contact info

Researcher (Thomas Sjerps): t.sjerps@student.tudelft.nl
Thesis advisor (Rafa Bidarra): r.bidarra@tudelft.nl
Daily supervisor (Cynthia Liem): c.c.s.liem@tudelft.nl

(a) The first page for pupils, showing introductory information.



Explicit consent points

Please tick the appropriate boxes.

	Yes	No
A: General agreement – Research goals, participant tasks and voluntary participation		
1. I have read and understood the study information dated the 11th of April 2025, or it has been read to me. I have been able to ask questions about the study and my questions have been answered to my satisfaction.	☐	☐
2. I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions, and I can withdraw from the study at any time, without having to give a reason.	☐	☐
3. I understand that taking part in the study involves the researcher asking me to play musical material, evaluating the tool I'm doing this with and evaluating my own and others' performance.	☐	☐
4. I understand that the study will end on the 1st of June 2025.	☐	☐
B: Potential risks of participating (including data protection)		
5. I understand that taking part in the study involves collecting specific personally identifiable information (PII) by filling in my information in this Informed Consent form, with the potential risk of my identity being revealed when these Consent forms are compromised.	☐	☐
6. I understand that secure data storage of this Informed Consent form will be used to minimize the threat of a data breach and protect my identity in the event of such a breach.	☐	☐
7. I understand that personal information collected about me that can identify me, such as my full name, will not be shared beyond the study team.	☐	☐
8. I understand that the (identifiable) personal data I provide will be destroyed 3 years after I hand it in.	☐	☐
C: Research publication, dissemination and application		
9. I understand that after the research study the de-identified information I provide will be used for analysis that will be a part of the reasoning in the final thesis.	☐	☐
D: (Long term) data storage, access and reuse		
10. I give permission for the de-identified results that I provide to be archived in the public, open, TU Delft Education Repository.	☐	☐

(b) The second page for pupils, showing a consent point checklist.

Signatures

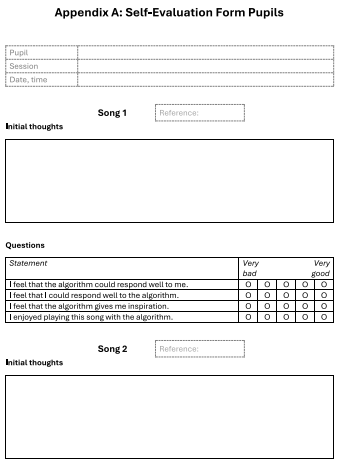
Name of participant	*Signature*	*Date*

I, as researcher, have accurately read out the information sheet to the potential participant and, to the best of my ability, ensured that the participant understands what they are freely consenting to.

Name of researcher	*Signature*	*Date*

(c) The third page for pupils, showing regions for signatures.

Figure A.2: The evaluation form used for the pupils and experts in experiment 1.



Appendix A: Self-Evaluation Form Pupils

Pupil	
Session	
Date, time	

Song 1 Reference:

Initial thoughts

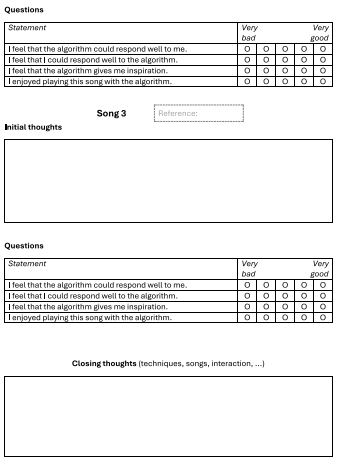
Questions

Statement	*Very bad*				*Very good*
I feel that the algorithm could respond well to me.	O	O	O	O	O
I feel that I could respond well to the algorithm.	O	O	O	O	O
I feel that the algorithm gives me inspiration.	O	O	O	O	O
I enjoyed playing this song with the algorithm.	O	O	O	O	O

Song 2 Reference:

Initial thoughts

(a) The first page of the evaluation form.



Questions

Statement	*Very bad*				*Very good*
I feel that the algorithm could respond well to me.	O	O	O	O	O
I feel that I could respond well to the algorithm.	O	O	O	O	O
I feel that the algorithm gives me inspiration.	O	O	O	O	O
I enjoyed playing this song with the algorithm.	O	O	O	O	O

Song 3 Reference:

Initial thoughts

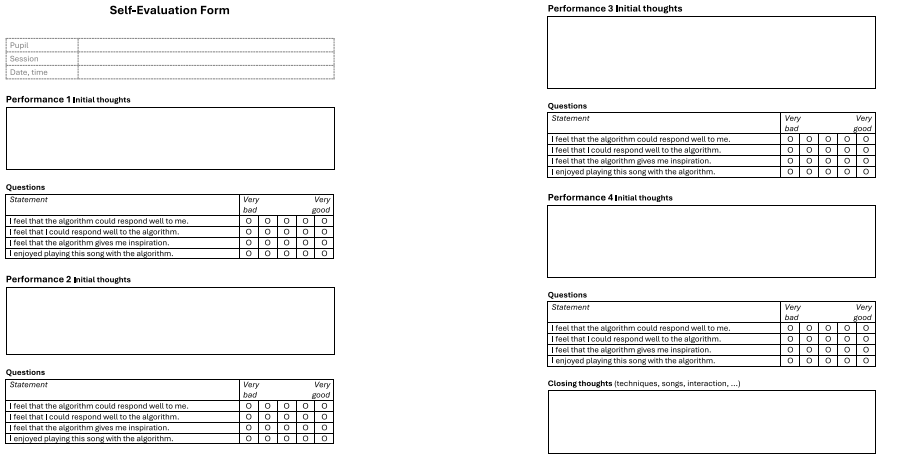
Questions

Statement	*Very bad*				*Very good*
I feel that the algorithm could respond well to me.	O	O	O	O	O
I feel that I could respond well to the algorithm.	O	O	O	O	O
I feel that the algorithm gives me inspiration.	O	O	O	O	O
I enjoyed playing this song with the algorithm.	O	O	O	O	O

Closing thoughts (techniques, songs, interaction, ...)

(b) The second page of the evaluation form.

Figure A.3: The evaluation form used for the pupils in experiment 1.



Self-Evaluation Form

Pupil	
Session	
Date, time	

Performance 1 Initial thoughts

Questions

Statement	*Very bad*				*Very good*
I feel that the algorithm could respond well to me.	O	O	O	O	O
I feel that I could respond well to the algorithm.	O	O	O	O	O
I feel that the algorithm gives me inspiration.	O	O	O	O	O
I enjoyed playing this song with the algorithm.	O	O	O	O	O

Performance 2 Initial thoughts

Questions

Statement	*Very bad*				*Very good*
I feel that the algorithm could respond well to me.	O	O	O	O	O
I feel that I could respond well to the algorithm.	O	O	O	O	O
I feel that the algorithm gives me inspiration.	O	O	O	O	O
I enjoyed playing this song with the algorithm.	O	O	O	O	O

Performance 3 Initial thoughts

Questions

Statement	*Very bad*				*Very good*
I feel that the algorithm could respond well to me.	O	O	O	O	O
I feel that I could respond well to the algorithm.	O	O	O	O	O
I feel that the algorithm gives me inspiration.	O	O	O	O	O
I enjoyed playing this song with the algorithm.	O	O	O	O	O

Performance 4 Initial thoughts

Questions

Statement	*Very bad*				*Very good*
I feel that the algorithm could respond well to me.	O	O	O	O	O
I feel that I could respond well to the algorithm.	O	O	O	O	O
I feel that the algorithm gives me inspiration.	O	O	O	O	O
I enjoyed playing this song with the algorithm.	O	O	O	O	O

Closing thoughts (techniques, songs, interaction, ...)

(a) The first page of the evaluation form.

(b) The second page of the evaluation form.

Figure A.4: The evaluation form used for the pupils in experiment 2.

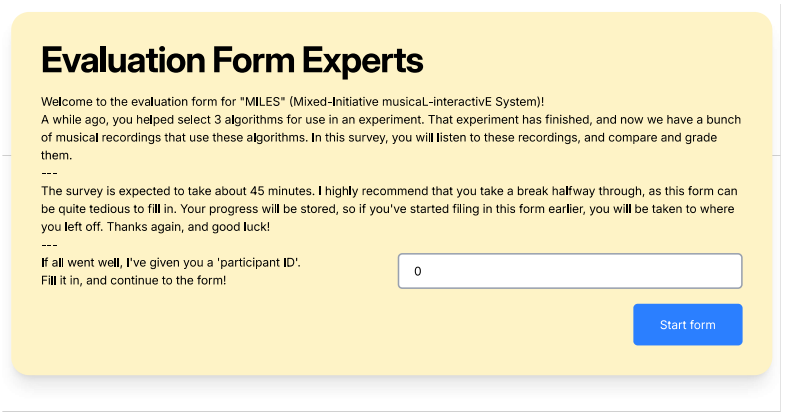


Figure A.5: The introductory page of the expert evaluation form of experiment 1.

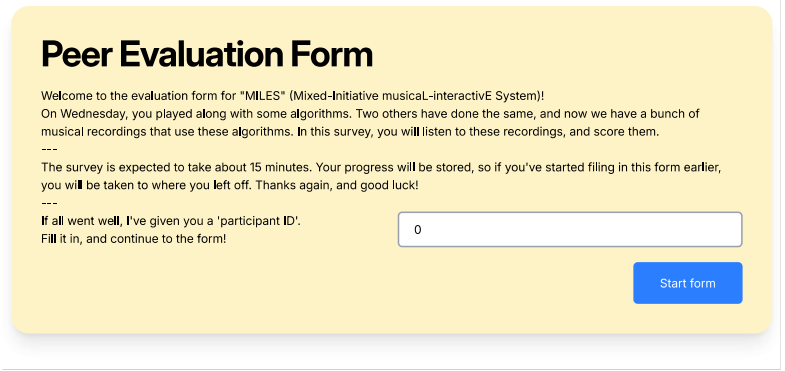


Figure A.6: The introductory page of the expert evaluation form of experiment 2.

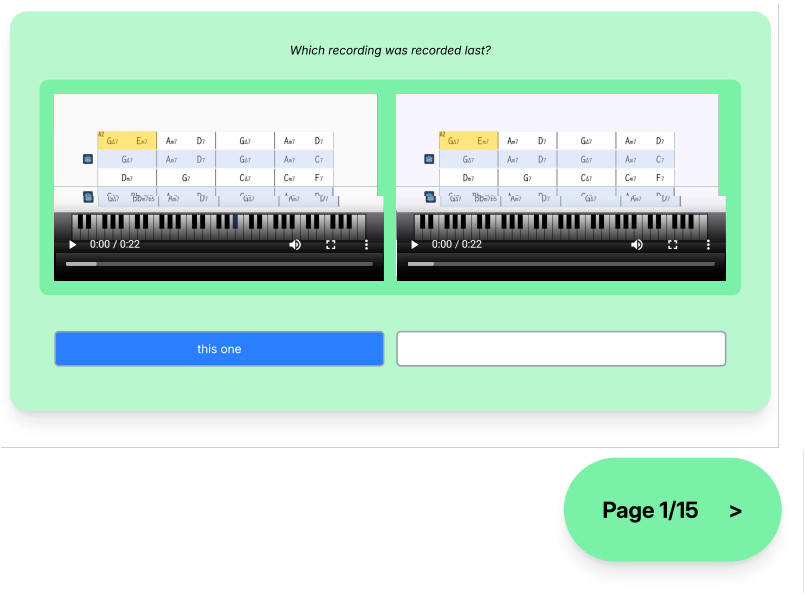


Figure A.7: An ordering page of the expert evaluation form of experiment 1.

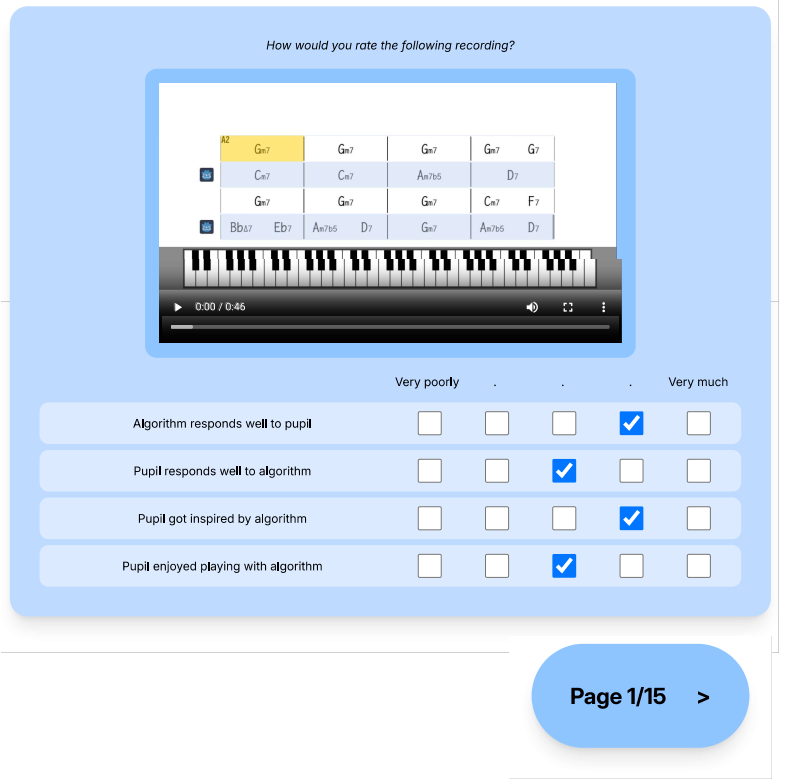


Figure A.8: A rating page of the expert evaluation form of experiment 1.

B

Miscellaneous Algorithms

This appendix contains an overview of all algorithms present in *MILES* as of writing. Further implementation details can be found in our GitHub repository[1], which contains all code used for *MILES*, including tokenization of *Weimar Jazz Database* and *Lakh MIDI Dataset* material (see chapter 2), training procedures, et cetera.

Algorithms are labelled "v1" and "v2" to reflect the tokenization strategy that was used. More information about tokenization can be found in appendix D.

B.1. Common functionality

Many algorithms use the same procedures for achieving common functionality, as described in chapter 4.1. This section details further common functionality that was used in the algorithms of *MILES*.

B.1.1. Chord-scale notes

When playing over a lead sheet, it is important to not play notes that clash with the currently playing chord. We have two implementations of techniques that keep notes within the chord of the lead sheet that they are played in. We describe this procedure as a function we call $ChordScaleNote(n)$, which converts an integer into a note ranging from 1 to 12.

The first iteration of chord scale note resolution implements chord-scale notes in a similar way to the technique used in *GenJam* [16]. In this iteration, n ranges from 1 to 7. Per chord type, *MILES* stores 7-note scales that fit over every type of chord. During playback, these scales are transposed to the currently-playing chord and shifted in such a way, that $ChordScaleNote(1)$ is either a C, a C♯ or a C♭. This prevents discontinuities when a melody is playing during a drastic change of chord. Phrased differently, this iteration takes notes that are stripped of accidentals, and reapplies the accidentals of the currently-playing chord.

The second iteration of the procedure (which is only used for second-iteraton token algorithms) allows for n to range from 1 to 12. The first part of the procedure transposes n such that it is in the key of C. Instead of using chord-scale relations, we apply substitutions per chord type (eg. for C major: E♭→E). Then, n is transposed back to its original key.

B.1.2. Applying swing

Triplet swing is a way of musical rhythm where eighth-notes are not played 'straight' (taking up half of a quarter note), but where the first note in an eighth-note pair takes up twice the time as the second note. As some songs in *MILES* are played in swing tempo, we have implemented functionality to transform the timing of generated (straight) notes to fit the straight rhythm of these songs. Applying swing of some time t is done with a piecewise linear function we call `ApplySwing`, where values of t that fall on whole notes are mapped to themselves, but notes that fall in-between whole notes are mapped to a time value two-thirds along the way. The pseudocode and plot of this function can be found in Figures B.1 and B.2 respectively. The inverse of this function (`RemoveSwing`) is also present in *MILES* for turning swung notes into straight notes.

[1] GitHub: sjerpsthomas/miles

function APPLYSWING(t)
    // Transform to within quarter note
    $t \leftarrow t * 4$
    $quarterNote \leftarrow Int(t)$
    $t \leftarrow t - quarterNote$
    // Piecewise function
    if $t < \frac{1}{2}$ **then**
        $t \leftarrow t/\frac{1}{2} * \frac{2}{3}$
    else
        $t \leftarrow t/\frac{1}{2} * \frac{1}{3}$
    // Transform back with quarter note
    $t \leftarrow t + quarterNote$
    $t \leftarrow t * 4$
    // Return **return** t

Figure B.1: The general procedure used to generate measures of a maximum number of tokens

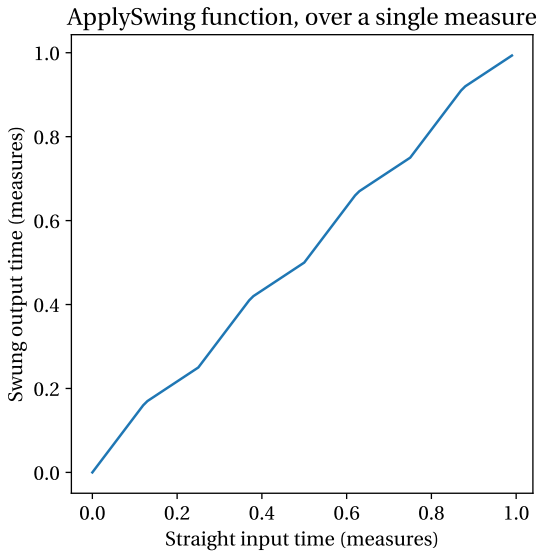


Figure B.2: A plot that shows the result of applying swing.

B.1.3. Generating measures

In our tokenization approach, (approximate) measure boundaries are signified by a measure token. However, some models can produce many tokens without generating a measure token, which results in unpleasantly short notes. In order to avoid generation of too many tokens per measure, we propose the following algorithm to force measure tokens every n tokens:

B.2. 'Baseline' algorithms

In this section, we detail 'baseline' algorithms which do not process any user data but still aim to produce musically correct outputs.

B.2.1. Note Random

This algorithm generates notes with a random length and pitch within the currently-playing chord. All generated notes have velocity level 96, and swing is applied when applicable. Generation stops when the four measures are filled in.

```
function ForceTokenMeasureLength(n)
    measureCount ← 0
    currentMeasureSize ← 0
    while true do
        // Generate token
        newToken ← Generate(...)
        // Force measure tokens
        if currentMeasureSize > n then
            yield Measure
            (potentially force token in model)
            newToken ← Measure
            currentMeasureSize ← 0
        // Increment measure number, return
        if newToken = Measure then
            measureCount = measureCount + 1
            if measureCount = 4 then return
```

Figure B.3: The general procedure used to generate measures of a maximum number of tokens

```
function NoteRandom.Generate
    t ← 0.0
    while t < 4.0 do
        // Generate values
        note ← RandomInt(40, 50)
        length ← RandomFloat(0.125, 0.325)
        restLength ← RandomFloat(0, 0.25)
        // Create note
        chordScaleNote ← ChordScaleNote(note)
        yield Note(t, length, chordScaleNote, 96)
        // Increment time
        newTime ← t + length + restLength
        newTime = ApplySwing(newTime)
        t = newTime
```

Figure B.4: The generate function used by the Note Random algorithm.

B.3. Markov-based algorithms

Chapter 4.3 details the implementation of Markov-based algorithms. This section details an additional Markov-based algorithm.

B.3.1. NoteRep Markov v2

This algorithm uses the variable-order Markov model used in Token Markov v2 on a less-compressed note representation than the one present in our tokenization pipeline. In our so-called `NoteRep` pipeline, note pitches are clamped between two octaves, two levels of velocity are present, and only three levels of inter-onset interval (length and rest length combined) are present.

During the learning phase, canonical and human solo `NoteRep` material are put into the variable-order Markov model. Generation makes the model produce notes until the four measures are filled in.

B.4. Factor oracle-based algorithms

Chapter 4.4 details the implementation of factor oracle-based algorithms. This section details an additional factor oracle-based algorithm.

B.4.1. Note Factor Oracle

This algorithm considers all human solo data and the canonical solo corresponding to the currently-playing solo. In the learning stage, the algorithm first transforms this data into note number, velocity, length and rest length values, which we call `MelodyNote`. These `MelodyNote` values are put into a factor oracle model (with human solo data appended throughout playback). This factor oracle model considers (edge) values to be equal whenever the pitch of the `MelodyNote` is equal, eliminating unnecessary edges. The algorithm keeps track of the index within the factor oracle where the just-played human four starts, which we call `HumanFourStart`.

At generation, the algorithm starts out somewhere between this index and the end of the factor oracle, and continually traverses the oracle from there. Differing from the Token Factor Oracle v1 and v2 algorithms described in chapter 4, the odds of continuing to the next node (instead of skipping forward) are slightly biased, by only going to a random edge 50% of the time. When the end of the factor oracle is reached, traversal continues from the first node of the oracle. If applicable, swing is applied to all notes. Generation stops when all four measures are filled in.

function NOTEFACTORORACLE.GENERATE
    $index \leftarrow RandomInt(HumanFourStart, |FO.Nodes|)$
    $t \leftarrow 0.0$
    while $t < 4.0$ **do**
        // Traverse
        $(melodyNote, newIndex) \leftarrow FO.Nodes[index].Traverse()$
        // Loop back to start
        if No note found **then**
            $(melodyNote, newIndex) = FO.Nodes[0].Traverse()$
        // Create note
        $absoluteNote \leftarrow AbsoluteNote(melodyNote.Note)$
        yield $Note(t, melodyNode.Length, absoluteNote, melodyNote.Velocity)$
        // Increment time
        $newTime \leftarrow t + melodyNote.Length + melodyNote.RestLength$
        $newTime = ApplySwing(newTime)$
        $t = newTime$

Figure B.5: Caption

B.5. Other algorithms

This section details miscellaneous other algorithms that were implemented for *MILES*.

B.5.1. Token Shuffle v1

This algorithm is inspired by the genetic mutations described in Biles' *Interactive GenJam* [17] for mutating human-played music into novel material. This algorithm only considers the last four that the user played. After tokenizing this material, it selects a region of tokens and applies one of the following permutations:

- **Mirror**: Reverse the selected list of tokens, keeping the length of notes and rests. This is done by first generating intermediate tokens that also encode the current speed and leave out speed change tokens, reversing this list, and then adding the speed tokens to the result again.
- **Flip**: Flip all notes of the selected list of tokens vertically (note: tokens are represented over only a single octave).
- **Transpose**: Transposes all notes of the selected list of tokens by between -4 and 3 chord-scale steps, wrapping along the octave.

This process is repeated for a total of three times. Finally, the resulting tokens are reconstructed back to note material again, and these resulting notes are played.

B.5.2. Token Transformer v1

This algorithm tokenizes the last four measures played by the user and inserts it into the input sequence of a Transformer model. This Transformer model has an internal sequence length of 10, and 3 layers of encoder layers.

The training set of the Transformer consists firstly of tokenized tracks from the *Lakh MIDI Dataset*. These tracks were filtered by monophonic relevancy to only include melodic MIDI instruments with 10 or more unique MIDI notes and not too much note overlap.

Generation consisted of generating tokens using the procedure described in appendix B.1.3 with a maximum measure length of 10. Whenever a forced measure token is inserted, this is added to the input sequence of the model. These tokens are then reconstructed and the resulting notes are played.

B.5.3. Token Neural Net v2

This algorithm tokenizes the last four measures played by the user and inserts it into the input sequence of a MLP model. This model has a context window of 30 tokens, and works as follows:

- An embedding layer turns every token into a 3-dimensional embedding;
- After flattening, these embeddings are brought down to a single 5-dimensional tensor;
- A ReLU layer, another 5-to-5 linear layer and a dropout layer provide further transformation;
- Finally, this 5-dimensional tensor is brought to 16 logits.

During the generation phase, measure size is not limited like the other algorithms, as this was not deemed necessary. The generated tokens are reconstructed and the resulting notes are played.

C

Miscellaneous Symbolic Features

The analysis framework of *MILES* contains a selection of pre-existing symbolic analysis features. This appendix gives an overview of these features. These features analyse *MILES*' output files, which contain n notes over m measures that have the following properties:

- `OutputName` (type: enum): the instrument or player that played this note;
- `Time` (type: double): the time (in measures) of the onset of the note;
- `Length` (tpye: double): the time (in measures) of the length of the note;
- `Note` (type: int): the MIDI note number of the note;
- `Velocity` (type: int): the velocity (0-127) of the note.

All features that do not directly state a source are inspired by work by McKay [45], who designs a system that predicts a genre of a song by using MIDI features of these songs.

C.1. Note pitch features

MILES implements two features that work on note pitch directly. Firstly, the **note standard deviation**, which takes the standard deviation of all note values:

$$\sigma_{note} = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{N} (\texttt{Note}(N_i) - \mu_{note})^2}$$

Where we compute the average to be:

$$\mu_{note} = \frac{1}{n} \cdot \sum_{i=1}^{N} \texttt{Note}(N_i)$$

As well as that, we define **note range** to be the difference between the highest and lowest note:

$$note\ range = max_{i=1}^{N} \texttt{Note}(N_i) - min_{j=1}^{N} \texttt{Note}(N_j)$$

C.2. Interval features

The features discussed in this section make use of the list of all note intervals I, which can be computed by calculating all differences between each note and the next:

$$I = \{\texttt{Note}(N_{i+1}) - \texttt{Note}(N_i) \mid i \in \{0, 1, \ldots, n-2\}\}$$

Keep in mind that this list has one less element than N. We define the **interval average** to be the average of this list:

$$\mu_{interval} = \frac{1}{n-1} \cdot \sum_{i=1}^{n-1} I_i$$

Furthermore, we define the **interval standard deviation** to be the standard deviation of this list:

$$\sigma_{interval} = \sqrt{\frac{1}{n-1} \cdot \sum_{i=1}^{n-1} (I_i - \mu_{interval})^2}$$

C.3. Note length features

We also define features based on the lengths of notes, like **length standard deviation**, which takes the standard deviation of all note lengths:

$$\sigma_{length} = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{N} (\texttt{Length}(N_i) - \mu_{length})^2}$$

Where we compute the average to be:

$$\mu_{length} = \frac{1}{n} \cdot \sum_{i=1}^{N} \texttt{Length}(note)$$

As well as that, we define **note density** (assuming that all notes have been transformed to be monophonic):

$$note\ density = \frac{1}{m} \cdot \sum_{note \in \mathrm{N}} \texttt{Length}(note)$$

C.4. Melodic arc features

The following features express rhythmic qualities of the input melody and are defined on 'arc lengths', as described in chapter 3.3.3. The algorithm to get these arc lengths can be found in Figure C.1. We notate BPM-dependent notes with N'.

Apart from the extrema ratio, we implemented the **melodic arc duration average**:

$$\mu_{duration} = \frac{1}{|A|} \cdot \sum_{i=1}^{|A|} \texttt{Duration}(A_i)$$

We similarly implement the **melodic arc height average**:

$$\mu_{height} = \frac{1}{|A|} \cdot \sum_{i=1}^{|A|} \texttt{Height}(A_i)$$

C.5. Rhythm features

In chapter 3.3.3, we define the IOI to be the difference in onset time between notes. Aside from NPVI, *MILES*' symbolic analysis component also includes the average IOI:

$$\mu_{ioi} = \frac{1}{n-1} \cdot \sum_{i=1}^{n-1} IOI_i$$

function GETARCLIST(N', I)

// Get intervals and notes

$intervals \leftarrow I$

$notes \leftarrow N'$

// Get places where interval switches sign

$boundaries \leftarrow \{i+1 \mid intervals_{i+1} - intervals_i \neq 0\}$

// Build arcs list based on boundaries

$arcs \leftarrow \{\}$

$arcStartIndex = 0$

// Iterate over boundaries

for all $boundary \in boundaries$ **do**

$newArc \leftarrow \{notes_i \mid i \in \{arcStartIndex, arcStartIndex+1, \ldots, boundary\}\}$

$Append(arcs, newArc)$

$arcStartIndex \leftarrow boundary$

// Append last arc

$Append(arcs, \{notes_i \mid i \in \{arcStartIndex, arcStartIndex+1, \ldots, n\}\})$

// Get durations and heights

$$durations \leftarrow \{\begin{cases} (\texttt{Time}(arcs_i[-1]) + \texttt{Length}(arcs_i[-1]) - \texttt{Time}(arcs_i[0]), & \text{if arc}_i \neq \emptyset \\ 0.0, & \text{otherwise} \end{cases} \mid i \in 0, 1, \ldots, |arcs|\}$$

$$heights \leftarrow \{\begin{cases} max_{j=1}^{|arcs|}\texttt{Note}(arcs_i[j]) - min_{k=1}^{|arcs|}\texttt{Note}(arcs_i[k]), & \text{if arc}_i \neq \emptyset \\ 0.0, & \text{otherwise} \end{cases} \mid i \in 0, 1, \ldots, |arcs|\}$$

// Return

return $Zip(durations, heights)$

Figure C.1: The procedure that creates 'melodic arcs' based on a melody

D

Token Overview

When dealing with algorithms like Markov chains or factor oracle models, symbolic music data must be compressed in some way. Some algorithms do this by chunking material together, or by only considering the pitches of input notes. We have implemented a method that converts a monophonic symbolic melody into a sequence of 'tokens', which can be used for implementing many generative techniques. These tokens are human-readable and aim to encode musical features in such a way that conversion from symbolic data to tokens ('tokenization') is musically precise, and conversion back to symbolic data ('reconstruction') should give musically valid data regardless of what tokens it is given. The tokens of our technique support chord-scale classes relative to the key of the song, a passing tone and rest indicator, speed and dynamics indications, and a measure delimiter. Tokenization and reconstruction take place in real time and tokens can be stored for later playback and analysis.

In order to enhance the clarity of our tokenization strategy implementation and increase modularity, we decided to implement the tokenization and reconstruction procedures as several phases. This chapter summarizes the functionality of these phases.

D.1. Pitch stage

The pitch stage of our tokenization strategy is responsible for chord-scale notes and potentially passing tones.

D.1.1. First iteration

The pitch stage of tokenization goes over every note in the melody. If the pitch interval with its preceding note is the same as the one with its succeeding note, this note is replaced with a passing tone token. If not, the (first-iteration) chord-scale mapping (described in chapter B.1.1) is applied to the note.

Reconstruction uses the chord-scale mapping to recover the original notes. The pitches of passing tone tokens that remain are determined by linearly interpolating between the notes that lie at the boundaries of groups of these tokens.

D.1.2. Second iteration

In the second iteration of our tokenization strategy, the pitch stage only applies the (second-iteration) chord-scale mappings, omitting the use of passing tone tokens.

D.2. Octave stage

The octave stage of our tokenization strategy is responsible for stripping away and re-applying octave information to notes. This limits all tokens to be within a single octave.

D.2.1. First iteration

The octave stage of tokenization consists of a single modulo operation, which transposes all notes to lie in a single octave.

Reconstruction consists of defining 'octave events', which signal an upward or downward jump in octave, to occur in between notes with a large interval (about half an octave). A selection of these octave events is made based on a heuristic priority function which considers the length of these two notes and the size of the interval between them. Finally, these octave events are applied such that notes can now be present in multiple octaves.

D.2.2. Second iteration

The second iteration of the octave stage is similar to that of the first. The only difference occurs during reconstruction: no filtering of of the determined 'octave events' occurs, and thus all of the octave transpositions are applied.

D.3. Timing stage

The timing stage of our tokenization strategy consists of converting the *time* and *length* properties of notes into a sequence of speed indications and note/rest tokens without any extra temporal information, such that the velocity indication tokens modify the length of subsequent notes until updated by another indication token.

D.3.1. First iteration

In the first iteration, the tokenization of the timing stage starts out by potentially removing swing. Next, rests are recognized and tokens are placed in groups in such a way that the reconstructed end times of these groups align with a quantized time grid, based on the speeds of these groups.

Reconstruction is achieved by grouping note, passing tone and rest tokens based on their indicated speeds and separated by measure. Next, these groups are scaled to be proportional to their indicated speeds, and are aligned based on a quantized time grid, based on the speeds of these groups. Finally, swing is potentially added again to the start and end times of notes.

D.3.2. Second iteration

Similar to the first iteration, the tokenization of the timing stage starts out by potentially removing swing. Next, rests are recognized and tokens are emitted based on the lengths of these notes and rests.

Reconstruction is achieved by defining an optimization problem where the starting points of all notes/rests are considered. The objective function of this optimization problem balances notes that are not too short and are of their specified length with note start time quantization and 'overtime' (where notes are allowed to be placed slightly beyond the four measures). We use the Cobyla ('constrained optimization by linear approximation') class of the Accord.Math.Optimization[1] C# package to implement this optimization.

D.4. Velocity stage

The velocity stage of our tokenization strategy is responsible for replacing explicit velocity information with quantized velocity indication tokens that modify the length of subsequent notes until updated by another indication token.

D.4.1. First iteration

The tokenization of the velocity stage emits velocity indication tokens by considering whether or not notes are louder than a threshold velocity.

Reconstruction consists of re-applying these velocity indication tokens and setting the velocities of emitted notes accordingly.

D.4.2. Second iteration

The second iteration of the velocity stage is identical to that of the first.

[1] http://accord-framework.net/

Bibliography

[1] P. T. Sowden, L. Clements, C. Redlich, and C. Lewis, "Improvisation facilitates divergent thinking and creativity: Realizing a benefit of primary school arts education.," *Psychology of Aesthetics, Creativity, and the Arts*, vol. 9, no. 2, p. 128, 2015.

[2] F. P. Brooks, A. L. Hopkins, P. G. Neumann, and W. V. Wright, "An experiment in musical composition," *IRE Transactions on Electronic Computers*, vol. EC-6, no. 3, pp. 175–182, 1957. DOI: 10.1109/TEC.1957.5222016.

[3] P. Manning, *Electronic and computer music*. Oxford University Press, 2013.

[4] A. Horner and D. E. Goldberg, *Genetic algorithms and computer-assisted music composition*. Ann Arbor, MI: Michigan Publishing, University of Michigan Library, 1991, vol. 51.

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

[6] J. Gillick, K. Tang, and R. M. Keller, "Machine learning of jazz grammars," *Computer Music Journal*, vol. 34, no. 3, pp. 56–66, 2010, ISSN: 01489267, 15315169. [Online]. Available: http://www.jstor.org/stable/40963033 (visited on 05/09/2025).

[7] N. Trieu and R. Keller, "Jazzgan: Improvising with generative adversarial networks," in *MUME workshop*, 2018.

[8] S. H. Hakimi, N. Bhonker, and R. El-Yaniv, "Bebopnet: Deep neural models for personalized jazz improvisations.," in *ISMIR*, 2020, pp. 828–836.

[9] H.-W. Dong, K. Chen, S. Dubnov, J. McAuley, and T. Berg-Kirkpatrick, "Multitrack music transformer," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023, pp. 1–5.

[10] A. Liapis, G. Yannakakis, and J. Togelius, "Sentient sketchbook: Computer-assisted game level authoring," English, in *Proceedings of the 8th International Conference on Foundations of Digital Games*, Society for the Advancement of the Science of Digital Games, 2013, pp. 213–220, ISBN: 978-0-9913982-0-1.

[11] J. E. Gain, P. C. Marais, and W. Straßer, "Terrain sketching.," in *SI3D*, 2009, pp. 31–38.

[12] P. P. Varga and R. Bidarra, "Harmony in hierarchy: Mixed-initiative music composition inspired by wfc," in *Entertainment Computing - ICEC 2024, Proceedings of the International Conference on Entertainment Computing*, Springer LNCS, Sep. 2024, pp. 1–11. [Online]. Available: http://graphics.tudelft.nl/Publications-new/2024/VB24.

[13] A. B. Loyall, "Believable agents: Building interactive personalities," Ph.D. dissertation, Carnegie Mellon University., 1997.

[14] B. Thom, "Bob: An interactive improvisational music companion," in *Proceedings of the fourth international conference on Autonomous agents*, 2000, pp. 309–316.

[15] F. Pachet, "The continuator: Musical interaction with style," *Journal of New Music Research*, vol. 32, no. 3, pp. 333–341, 2003.

[16] J. Biles *et al.*, "Genjam: A genetic algorithm for generating jazz solos," in *ICMC*, Ann Arbor, MI, vol. 94, 1994, pp. 131–137.

[17] J. A. Biles, "Interactive genjam: Integrating real-time performance with a genetic algorithm," in *ICMC*, 1998.

[18] C. Allauzen, M. Crochemore, and M. Raffinot, "Factor oracle: A new structure for pattern matching," in *SOFSEM'99: Theory and Practice of Informatics: 26th Conference on Current Trends in Theory and Practice of Informatics Milovy, Czech Republic, November 27—December 4, 1999 Proceedings 26*, Springer, 1999, pp. 295–310.

[19] G. Assayag and S. Dubnov, "Using factor oracles for machine improvisation," *Soft Computing*, vol. 8, no. 9, pp. 604–610, 2004.

[20] G. Assayag, G. Bloch, M. Chemillier, A. Cont, and S. Dubnov, "Omax brothers: A dynamic topology of agents for improvization learning," in *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, 2006, pp. 125–132.

[21] G. Assayag and M. Chemillier, *The OMax project page*. [Online]. Available: `http://recherche.ircam.fr/equipes/repmus/OMax/`.

[22] E. Chew, *Human-machine improvisations with mimi: Interrogating performance and listening processes and outcomes*, Invited plenary speaker, Perspectives on Musical Improvisation II, Department of Music, Oxford University, Oxford, UK, Sep. 2014.

[23] A. R. François, I. Schankler, and E. Chew, "Mimi4x: An interactive audio-visual installation for high-level structural improvisation," *International Journal of Arts and Technology*, vol. 6, no. 2, pp. 138–151, 2013.

[24] J. Nika, M. Chemillier, and G. Assayag, "Improtek: Introducing scenarios into human-computer music improvisation," *Computers in Entertainment (CIE)*, vol. 14, no. 2, pp. 1–27, 2017.

[25] I. Corintha, L. Outeiro, R. Dias, and G. Bernardes, "Am-i-blues: An interactive digital music instrument for guiding novice pianist in the improvisation of jazz melodies," in *Meeting of Research in Music, Arts and Design*, Springer, 2020, pp. 689–698.

[26] R. M. S. S. Dias, "Interfacing jazz: A study in computer-mediated jazz music creation and performance," Ph.D. dissertation, Instituto Politecnico de Castelo Branco (Portugal), 2018.

[27] I. Corintha, G. Cabral, and G. Bernardes, "Amigo: An assistive musical instrument to engage, learn and create music," in *Proceedings of the international conference on New Interfaces for Musical Expression*, 2019.

[28] J. Foote, "Visualizing music and audio using self-similarity," in *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, 1999, pp. 77–80.

[29] M. Wattenberg, "Arc diagrams: Visualizing structure in strings," in *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.*, IEEE, 2002, pp. 110–116.

[30] E. Chew, "Out of the grid and into the spiral: Geometric interpretations of and comparisons with the spiral-array model.," *Computing in musicology*, vol. 15, 2007.

[31] R. Cohn, "Neo-riemannian operations, parsimonious trichords, and their" tonnetz" representations," *Journal of Music Theory*, vol. 41, no. 1, pp. 1–66, 1997.

[32] Technimo, *Ireal pro*, version 2025.4, Apr. 3, 2025. [Online]. Available: `https://www.irealpro.com/`.

[33] R. Keller, *"You Go to My Head", Al Biles and GenJam*, May 2019. [Online]. Available: `https://www.youtube.com/watch?v=RDgJw2kiuWU`.

[34] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: A fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020.

[35] P. Van Kranenburg, M. De Bruin, and A. Volk, "Documenting a song culture: The dutch song database as a resource for musicological research," *International Journal on Digital Libraries*, vol. 20, pp. 13–23, 2019.

[36] L. Crestel, P. Esling, L. Heng, and S. McAdams, "A database linking piano and orchestral midi scores with application to automatic projective orchestration," *arXiv preprint arXiv:1810.08611*, 2018.

[37] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.

[38] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011, pp. 591–596.

[39] M. Pfleiderer, K. Frieler, J. Abeßer, W.-G. Zaddach, and B. Burkhart, "Inside the jazzomat," *New Perspectives for Jazz Research*, 2017.

[40] F. Höger, K. Frieler, M. Pfleiderer, and S. Dixon, "Dig that lick: Exploring melodic patterns in jazz improvisation," in *Late Breaking/Demo at the 20th International Society for Music Information Retrieval Conference, Delft, The Netherlands*, 2019.

[41] H. Leonard, *Omnibook Series Hal Leonard online*. [Online]. Available: `https://www.halleonard.com/series/OMNIBK`.

[42] V. Madaghiele, P. Lisena, and R. Troncy, "Mingus: Melodic improvisation neural generator using seq2seq.," in *ISMIR*, 2021, pp. 412–419.

[43] K. Frieler and W.-G. Zaddach, "Evaluating an analysis-by-synthesis model for jazz improvisation," *Transactions of the International Society for Music Information Retrieval*, vol. 5, no. 1, pp. 20–34, 2022.

[44] L.-C. Yang and T. Grilo, *RichardYang40148/mgeval*. GitHub, Aug. 2, 2021. [Online]. Available: `https://github.com/RichardYang40148/mgeval`.

[45] C. McKay and I. Fujinaga, "Automatic genre classification using large high-level musical feature sets.," in *ISMIR*, vol. 2004, 2004, pp. 525–530.

[46] D. R. Abrams, *Rhythmic complexity in jazz: An information theory approach*, 2023.

[47] D. Müllensiefen, "Fantastic: Feature analysis technology accessing statistics (in a corpus): Technical report v1," *London, England: Goldsmiths, University of London. Retrieved from http://www. doc. gold. ac. uk/isms/m4s/Google Scholar*, 2009.

[48] D. Müllensiefen, K. Frieler, *et al.*, "Cognitive adequacy in the measurement of melodic similarity: Algorithmic vs. human judgments," *Computing in Musicology*, vol. 13, no. 2003, pp. 147–176, 2004.

[49] F. Pachet, P. Roy, and R. Foulon, "Do jazz improvisers really interact?: The score effect in collective jazz improvisation," in *The Routledge companion to embodied music interaction*, Routledge, 2017, pp. 167–176.

[50] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1990, pp. 249–256.

[51] G. T. Toussaint, "The pairwise variability index as a tool in musical rhythm analysis," in *Proceedings of the 12th International Conference on Music Perception and Cognition & the 8th Conference of the European Society for the Cognitive Sciences of Music*, 2012, pp. 1001–8.