Music Learning with Massive Open Online Courses (MOOCs)
L. Steels (Ed.)

© 2015 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution License.
doi:10.3233/978-1-61499-593-7-101

Chapter 8. Giant Steps in Jazz Practice with the Social Virtual Band

[a] *Sony CSL Paris, 6 rue Amyot, 75005 Paris, France*

Mathieu Ramona [a], François Pachet [a], Stanislaw Gorlow [a]

Abstract. This chapter deals with the issue of learning how to improvize. Traditional MOOCs provide jazz students with comprehensive theoretical and motivate students to practice intensively on their own. However, without a view of one's progress, and without feedback, individual practice is a long and winding road along which many students get lost. Indeed, most jazz learning systems lack these two crucial ingredients, resulting in high drop-out rates. This chapter addresses the issue of designing tools for supporting improvisation practice by bringing in a social dimension enabling peer-to-peer feedback, as well as a cloud-based infrastructure enabling arbitrary vizualisations of the evolution of the student performance. We introduce *Social Virtual Band*, a system that lets learners improvize solos on dynamically created accompaniments, and that records and archives all the training sessions along with the provided accompaniments on the cloud. Simple automatic feedback is presented to measure the evolution of his skills, based on a comparison between played note with scales obtained from automatic harmonic analysis. We describe the overall infrastructure underlying such a tool and discuss how such infrastructure opens up new possibilities for learning music.

Keywords. Jazz, Practice, Cloud, Automatic music accompaniment

Introduction

Jazz improvisation is a skill that requires both formal musical training, notably in harmony theory and melody development, and a lot of practice time (the 10,000 hour rule [9]) to integrate and literally embody the knowledge. Embodiment is crucial in jazz improvisation because the requirement of producing and playing music in real-time make it impossible to think at the symbolic level. The skills necessary to build solos, and to *play the changes* are unique in that respect, and have even been considered by some authors as key for a wide range of activities, including managerial ones [12]. Practicing jazz consists essentially in playing solos over existing jazz standards. The acquisition of solo building skills is difficult to formalize precisely but is acknowledged to require both an intimate mastery of the instrument [27] and the ability to communicate musically with others ([7]).

Several solutions have been proposed to assist jazz improvisation practice. The oldest one is probably the minus-one recordings, notably the Aebersold series [13]. With minus-ones, the student plays a recording of the accompaniment of a jazz standard, and can freely improvise on it, usually with top quality backing. Though immensely successful, this solution can be used only for a predefined set of recordings, and cannot be costumized either tempo-wise, style-wise or structure-wise (i.e. number of solos). Sotware solutions have been developed, such as the famous *Band-in-a-box* system [15], which generates medium quality accompaniments of arbitrary tunes, in various styles. More recently, apps running on portable devices have been developed, such as *iReal Pro* [28], and also provide medium quality accompaniments on arbitrary chord sequences. In all these cases however, the performance produced is lost, no feedback is given to the musician so there is no possibility of reflecting on past performances, comparing them, or building a synthetic view of the student's skills evolution.

This chapter describes a system that attempts precisely to push jazz practice a step beyond conventional practice, by providing musicians with a cloud-based system that enables not only the collection and analysis of performances, but also the production of feedbacks, either automatically or by peers, which are essential to guide the learning process.

1. Improvisation Practice on Jazz Standards

1.1. Structure of a lead sheet

The lead sheet is the basic element that defines a song in jazz. A lead sheet is a combination of a melody (usually monophonic) with a sequence of chord labels defining the harmonic progression. This harmonic progression can be seen as a realization of an underlying functional path which gives a particular flavor to the melody. The lead sheet also includes a time signature (e.g., 3/4 or 4/4) and possibly stylistic indications that give information about the tempo and the rhythmic patterns to be used. Figure 1 shows an example of such a lead sheet.

A trained musician can infer a lot of information from the lead sheet: the global tonality of the tune, the modulations (through the analysis of the chord sequence), the harmonic function of each chord in the progression, and consequently the scales which fit with the chord sequence that can be used for improvising. A lead sheet represents the *essence* of the song, and constitutes a reference knowledge shared by all jazz musicians ([7]).

A typical jazz player generally starts by playing the melody (aka the *theme*) and then elaborates his solo by paraphrasing the melody, while respecting the harmonic and structural constraints. In traditional jazz (typically be-bop) ensembles, the drum, bass and piano handle the rhythmic part and provide the harmonic support, while wind instruments play the solo parts in turns. Later extensions of the traditional jazz canvas, such as modal jazz, have progressively loosened up the traditional canvas and now any instrument can play solos in turn, including drums and bass.

1.2. The LSDB database

The first requirement of an online jazz practice system is to access digital representation of lead sheets. We use the comprehensive online database of jazz lead sheets [21] which contains all published jazz lead sheets, and is the first large scale database with both

Figure 1. A lead sheet for jazz standard *Giant Steps* by John Coltrane.

melodies and chord progressions in electronic format. The database currently contains about 11,000 songs coming from 62 song books, of which 20 are fake books or compilations of jazz standards, 32 are Bossa Nova song books, and 8 are composer specific song books (such as John Coltrane, Thelonious Monk or Bill Evans).

Such a database is a precious resource for Music Information Retrieval [3] and automatic music analysis [8] [11]. In the context of this experiment, it provides us with a comprehensive set of lead sheets for generating accompaniments to support the practice of jazz improvisation over nearly any standard a musician could choose to play on.

1.3. Features of a solo

The improvised solo is classically a paraphrase of the melody, that can lead to multiple sorts of variations (transposition to different keys or different scale, slight rhythmic variations, repetition over different chords) on motives (defined as the smallest melodic entities) possibly chosen from the melody or even cited from other sources.

What makes a good solo is of course a complex question, because it can be evaluated according to many musical dimensions. Jerry Coker states [4] that five factors concur in the outcome of a remarkable improvisation: intuition, intellect, emotion, sense of pitch and playing habits. Learning improvisation consists basically in developing a conscious control, through the intellect, of the other four factors, which should become inconscious with practice. From the point of view of the listener, a good improvisation can also be defined as a very subtle balance between predictability, as a mean to create attention, and surprise over the listener's expectations, to avoid boredom.

In fine, automatic feedback necessarily relies on *features* that can be extracted from the audio signal by pure computation. The extraction of so-called *audio features* (i.e. numeric or symbolic values that can be deterministically calculated by a machine from an audio signal) is indeed a key issue in Music Information Retrieval [22], but low-level features (estimated with few computation steps from the signal, e.g. spectral centroid, temporal moments, RMS, etc.) are generally unable to catch perceptively relevant features. Although some approaches [19] can automatically infer high-level features that fulfill a given task, we focus here on the musical features that can be defined from a reliable transcription of the solo (i.e. the automatic extraction of the notes).

Supposing that the transcription is reliable, each played note is detected and characterized by its *pitch* and its *start* and *end times*. The following features are easily estimated:

- **In-scale rate**: the so-called *In-scale rate* measures the proportion of notes played in the scales expected from a harmonic analysis of the chord progression (this will be developed in Section 2.5). Figure 2 shows two examples of simple melodic phrases played on a D minor chord ; example (a) is perfectly in scale, while (b) has 33 % notes off scale.

(a) In-scale rate: 100% (all notes are in scale).

(b) In-scale rate: 66% (3 notes out of 9 are off scale).

Figure 2. Two examples of in-scale measure over simple melodic phrases played on a D minor chord.

  Of course a 100 % rate is not necessarily an objective since *playing out* (as it is often called in jazz jargon) is precisely a key part of what makes a solo enjoyable, for example through side-slips (see [14] for a pedagogical definition). The estimation of the expected scale in also a non-trivial issue [26], although as a first approximation, an ad-hoc scale can be associated with each chord of the sequence.

- **On-beat rate**: measures the mean time shift of notes close to the first and third beats. Of course, discriminating accidental time-shifts (out-of-beat notes) and intentional syncopation is a key issue for such a measure.

- **Continuity**: melodic continuity is considered a key aspect of a good *sense of melody*. It is a difficult challenge for a human when playing fast, as it requires the ability to find quickly short paths between the note currently being played and the next ones, which may be in a different scale. This ability is referred to as chord change negotiation, stressing its inherent problem-solving dimension.
  Note that continuity does not necessarily imply brownness, in the sense of [29], i.e. the sole use of small intervals. It rather implies that notes are glued together smoothly, and not made up of isolated elements or patterns, concatenated without care. For instance, the phrase in Figure 3 contains several large intervals but is

perfectly continuous. It is straightforward to measure objectively and makes a relevant indicator of one's skills evolution.

Figure 3. A virtuoso passage (152 bpm) in a chorus by John McLaughlin on Frevo Rasgado (1977), that contains several large intervals but is perfectly continuous.

Of course, in fine the main issue with automatic feedback does not consist in feeding the user with raw feature values but to compare their distribution with large corpuses of real solos played form professional musicians, such as the Weimar's Jazzomat solo corpus [1] or collections of Django Reinhardt's solo transcriptions.

This chapter focuses on the design on the full process-line implemented to provide an automatic feedback on the *in-scale rate* defined herebefore.

2. *Social Virtual Band*

Social Virtual Band (SVB) is an environment designed to provide the jazz student with a software support to record himself over realistic accompaniment and to store and manage his collection of solos over the Cloud.

With the Cloud architecture, the musician can retrieve and listen to his solos. He can possibly request automatic analysis and feedback from peers or from the algorithms. In other words, *Social Virtual Band reifies* the solos and builds a social network around this atomic piece of interaction, just like Facebook and SoundCloud work with text, images and audio tracks.

2.1. Architecture

The Lead Sheet DataBase (LSDB), introduced in Section 1.2, is the central element of *Social Virtual Band*, since the whole user experience gravitates around this collection of songs. Therefore, most of the intelligence involving the songs and the solos is provided by a back-end server, through web services. Figure 4 sums up the architecture of *Social Virtual Band*.

The jazz student interfaces with the back-end server through the client application. The application provides a simple interface for selecting a song among the LSDB collection. It then connects to the server to retrieve both a MIDI file with an accompaniment generated in a particular style, and the score of the chord progression. The application can then play the accompaniment while showing the score and recording the user's solo. The application holds the recorded solos, along with the generated accompaniment and some metadata, and sends the whole package to the back-end server, where it is stored.

The user can then connect to the web platform hosted by the remote server, and manage his collection of solos or follow the evolution of his playing skills.

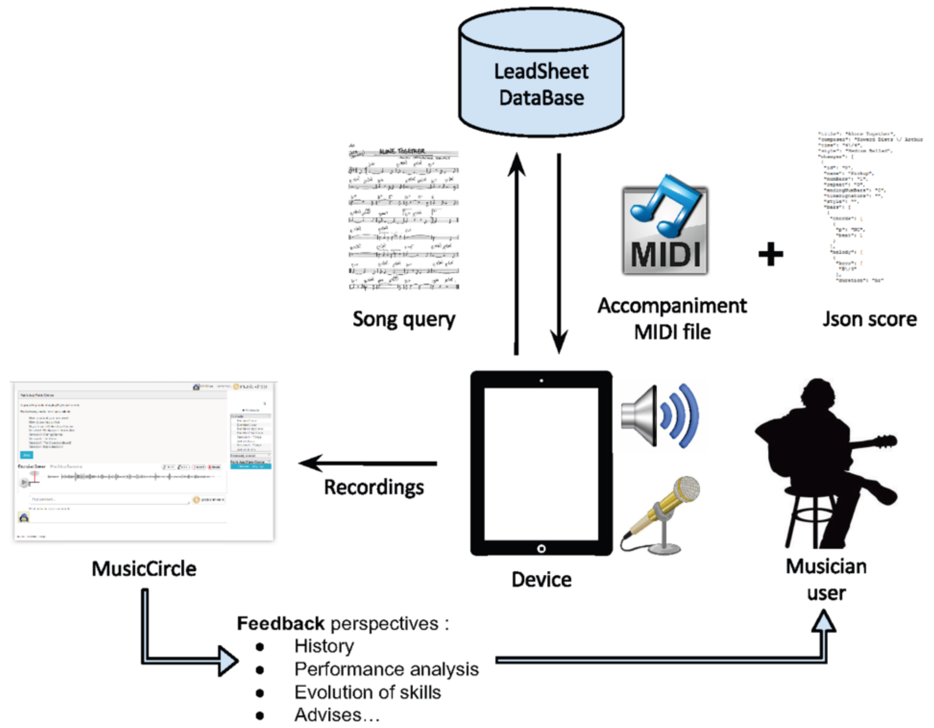

Figure 4. Architecture of the *Social Virtual Band* environment

2.2. Back-end server

The back-end server is based on an Apache Tomcat architecture[1] which provides various web services executed through Java code (designated as *servlets*).

The choice of using a web servlet is important in this context because it constraints us to define a clear, simple and yet versatile interface with the client, through the sole specifications of URL parameters. Indeed, each access to the server is performed with an HTTP request, that can be tested with any web browser, by typing the URL.

The web services involved in *Social Virtual Band*are implemented through so-called *Json Servlets*, i.e. servlets relying on the JSON (*JavaScript Object Notation*) for interfacing with the client. Figure 5 shows the list of servlets implemented in the *Social Virtual Band* server to communicate with the client application:

Database Listing Returns the list of songs that fulfill the input query, which can specify the composer, the song book source or the style.

Score Extraction Returns the chord progression of a lead sheet in the database, along with metadata associated with the song (time & key signature, style, tempo range, etc.). The client application uses that information to display the interactive score of the song.

[1] http://tomcat.apache.org

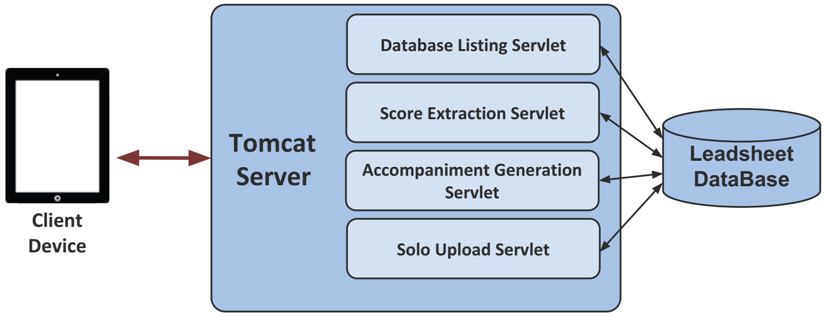

Figure 5. Architecture of the back-end Tomcat server

Accompaniment Generation This servlet is by far the most polyvalent provided by the server. It can generate a wide range of accompaniments by combining several algorithms developed by Sony CSL, detailed hereafter.

Solo Upload Retrieves the solo recorded with the application (with associated accompaniment and metadata), stores it into the database and performs the automatic processing to extract feedback.

2.3. Accompaniment Generation

The accompaniment is composed of three parts, following the usual convention for rhythmic sections in a jazz ensemble: drum, bass, and piano playing the voicing on the chord progression. In order to cope with the large variety of possible lead sheets to improvise on, it is natural to look for automatic accompaniment generators.

Several commercial systems propose automatic accompaniment systems (the most well-known being Band-in-a-Box and iReal Pro). Our architecture can cope with arbitrary accompaniment generators, taking as input a lead sheet and producing as output an accompaniment. Techniques vary greatly depending on the nature of the output. MIDI generators produce MIDI files which are then rendered by a embedded synthesizer, whereas audio generators generate directly audio files. We have experimented with both approaches.

Both the bass and piano part generation[2] are based on Markov constraints [20]. The Markov constraints is a technology for defining a Markov Model through a Constraint Satisfaction Problem (CSP). The CSP paradigm offers the possibility to specify hard constraints on sequences generated according to the Markov Model. This is particularly fitted for generating musical sequence, where many constraints apply for guaranteeing syntaxic and semantic consistency.

A MIDI generator was built using the technologies of Markov constraints, notably the meter constraint [23]. In a first step we have recorded accompaniments played by British jazz pianist Ray d'Inverno, over 20 jazz standards. In the generation step we take a given lead sheet as target and build a sequence of MIDI chunks (in this case, piano) that

[2]The generation of the drum part is still an on-going perspective, since the logic of chunks concatenation is very different when working on drums.

fit with the metrical structure of the lead sheet (the metrical location of chord changes). Then each chunk is adapted harmonically to fit with the target chords by changing the relevant pitches.

The accompaniment generated by the servlet is encoded into a MIDI file and sent to the client. The audio synthesis is performed in real-time by the client. Transmitting a symbolic MIDI file instead of raw audio dramatically reduces the volume of data downloaded by the remote client (typically, a 100 kB MIDI file can be equivalent to a 4 MB MP3 file encoded at 192 kbps bitrate).

2.4. Client Application

The client application provides a graphical interface for viewing and following a lead sheet and recording solos on generated accompaniments. It runs on Windows, MacOSX, Linux and iOS. The use of a portable device (smartphone or tablet) provides a very intuitive experience, since it relies on the embedded audio input and output devices, and requires no peripheral device.

2.4.1. User Interface

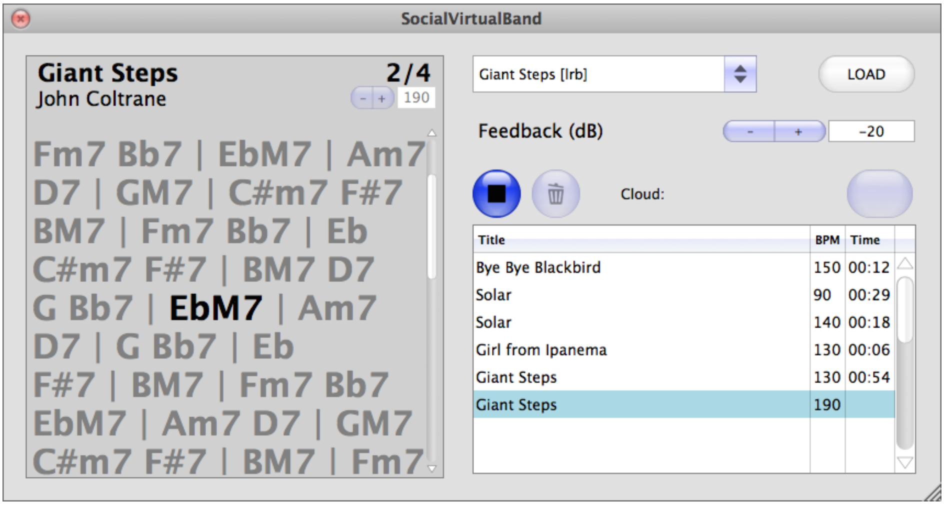

Figure 6. Graphical User Interface of the *Social Virtual Band* Client

Figure 6 shows a capture of the client user interface. The left panel shows the current lead sheet and follows the score during the recording process, while the right panel provides all the controls. The user can either select a pre-loaded song (from the upper choice list) or remotely long any song from LSDB by clicking the *LOAD* button. The lower table shows the list of solos recorded so far, and allows to replay them, and eventually upload them to the server.

2.4.2. Recording Process

The main issue dealt with is the recording process itself. Indeed, the application needs to play the accompaniment and to record the solo part (without the accompaniment) at the same time. And the recorded solo needs to be precisely synchronized with the accompaniment.

Figure 7 sums up the three possible recording configurations. The most straightforward configuration (a) consists in recording the solo while listening to the accompaniment played with the device loudspeaker. But this implies that the recorded solo track also captures the played accompaniment. This issue is fixed (b) by using headphones for the accompaniment. This way, the solo track is clean, but the configuration is less comfortable for the player, and the accompaniment might cover the hearing of his own playing. This latest issue is solved (c) by adding a feedback of the recorded solo to the played accompaniment. The *feedback* selector, visible in the GUI (Figure 6) lets the player modulate the feedback level to choose the right balance with the accompaniment. Of course this configuration only makes sense if the audio latency of the device is very low.

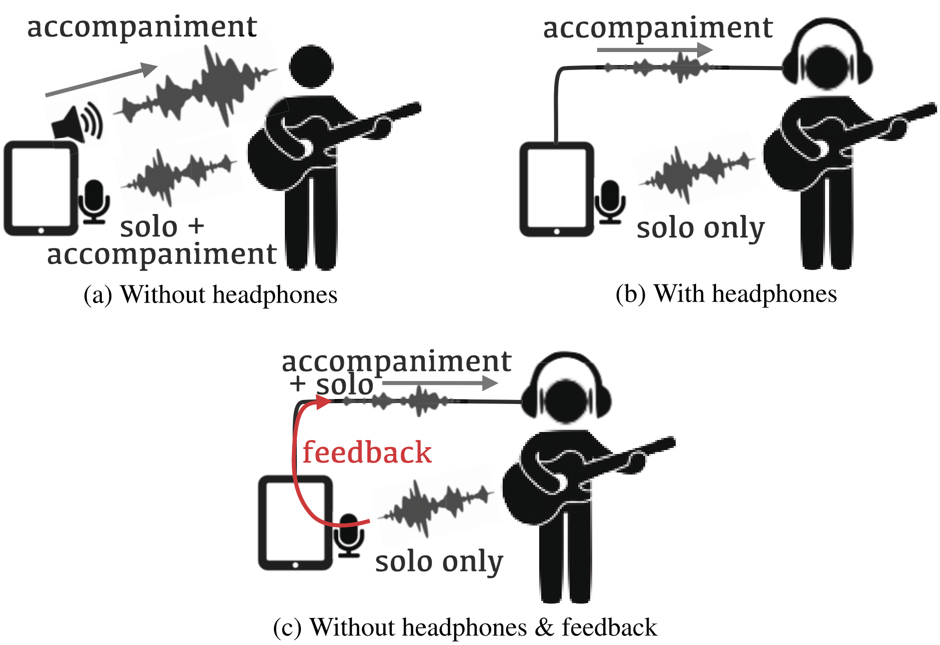

Figure 7. Comparison of the different recording configurations when using the SVB client application

In the three situations depicted in Figure 7, maintaining the synchronicity between the generated accompaniment and the recorded solo is crucial for guaranteeing the alignment of the solo record with the chord sequence, and provide a precise temporal and harmonic analysis of the performance.

As illustrated by Figure 8, there is an inevitable latency (due to the sound device buffers[3]) between the instants the application generates a sound and the sound that is

[3] rather than the travel of sound in the air, marginal here considering the short distance between the player and the device.

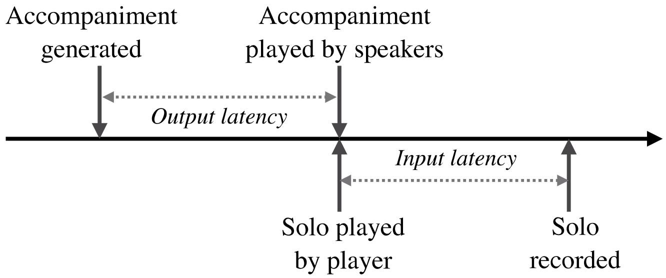

Figure 8. Output and Input latencies in the recording process

actually broadcast by the speakers (or headphones), and between the instants the player plays a note and that note is actually recorded by the application. These are respectively labeled as *output* and *input latency*, and are provided by the sound device itself. Supposing that the player plays simultaneously with the accompaniment, the recorded audio track has a delay that equals the sum of both latencies, and must be compensated.

In order to store the solo performance on the server without the accompaniment, the headphones solution, which is discussed in the preceding paragraph, is both simple and straightforward. Nevertheless, especially with respect to the playing comfort, it might be hindering and a set of headphones may not always be within reach. For this, Sony CSL in Paris has developed an alternative solution based on signal processing, using the frequency scale of the auditory system [10].

As a general rule, any cancellation algorithm works best when the solo in the recording is heard louder than the accompaniment. This can be easily achieved by means of a pickup attached directly to the instrument.

2.4.3. Upload to Server

As explained in previous Section, audio latencies are compensated to ensure a proper synchronization between the recorded solo track and the generated accompaniment. At the end of the recording process, the application holds both contents mutually aligned, along with the exact positions of chord changes, as depicted on Figure 9.

This information is sent to the remote server through a dedicated servlet that will analyze the audio track and use the aligned metadata to automatically extract relevant information.

2.5. Solo Analysis

2.5.1. Archiving

Each uploaded solo, when received on the server, is stored with the associated metadata. The server provides a Social Network (Figure 10) where each user can browse his own collection of solos.

The server keeps track of the whole history of solos previously recorded by the musician. It provides him an with statistical estimation of the evolution of his skills on a given track, for instance the evolution of the played tempo, as shown on Figure 11.

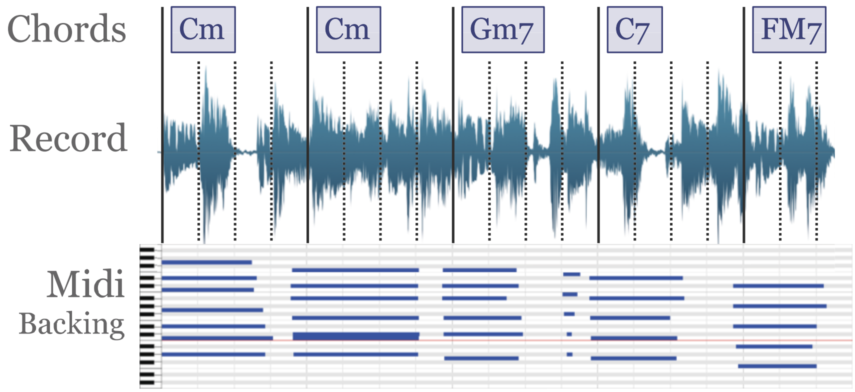

Figure 9. The recorded solo is sent to the remote server with properly aligned accompaniment and chord sequence.

Figure 10. Interface of the Social Network coupled with the *Social Virtual Band* application.

A more detailed analysis is provided on each uploaded solo, based on an automatic transcription process.

2.5.2. *Transcription*

Many contributions in the literature cover the subject of audio transcription. The general problem of polyphonic transcription involving several musical instruments is complex [5] [30], often tackled with source-separation related methods, such as Non-Negative

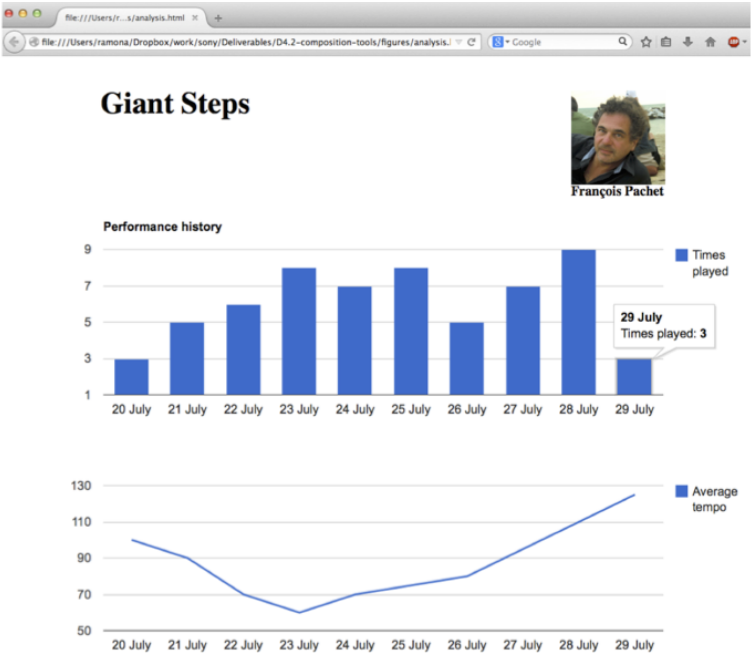

Figure 11. The *Social Virtual Band* web server provides statistical charts on the history of recorded solos. This figure shows the fluctuation of tempo used by the musician on the song Giant Steps.

Matrix Factorisation (NMF), which jointly estimates the dictionary and the decomposition of the audio signal into this dictionary [25] [2].

The polyphonic poly-instrumental problem is ill-posed, in the absence of (and even with) prior knowledge of the instrument timbre, because different instruments can play overlapping notes, and these notes usually share most of their harmonics (typically related by an octave or a fifth interval) because of usual musical assonance. A common way to avoid these ambiguities is to extract only the prominent melody [24].

A better-posed problem is the monophonic pitch estimation. The absence of overlapping between concurrent pitch harmonics simplifies the problem and results in a dramatic increase of state-of-the-art performance [6].

The *Social Virtual Band* use-case remains fairly simple because it is limited to mono-instrumental transcription of mostly-monophonic solos. The guitar and the piano allow the user to play chords in the solos, but we take the assumption that these polyphonic strokes are occasional and strictly vertical (as opposed to the horizontal polyphony of counterpoint). The system is currently using the guitar transcription algorithm developed by IIIA [16], but any other transcription algorithm could easily be plugged into the process line.

The melody extracted by the transcription algorithm is a sequence of notes, possibly polyphonic, bounded by start and end time values. In order to provide a symbolic representation (the score) of the transcribed solo, these temporal boundaries are quantized with a quantification step of 1/72 beat, which allows complex rhythmic divisions including sixteenth notes and triplets.

2.5.3. Harmonic Analysis

In order to evaluate the solo performances, we first analyze the lead sheets to extract the scales to be used for each chord. Obviously, real improvisation should not be perfect with regards to the underlying harmony, and "there is arguably some excess in the way improvisation is taught in jazz schools, focusing too much on the 'right' and 'wrong' notes" [Gilad Atzmon, personal conversation]. However, in our case we can use this information to produce an estimation of how far or close the improvisation is to the target harmony.

A lot of approaches have been used to analyze harmony (see, e.g., [18] for some references). In our case we use a simple dynamic programming approach, consisting in finding the harmonic analysis, for each chord label, which minimizes the number of modulations, i.e. scale changes. This process is performed through the following steps:

1. **Computation of possible harmonic analysis**
   For each chord label, we first compute (with the *MusES* library) the list of all possible harmonic analyses. A harmonic analysis is basically a scale (out of 3 possible scales types and 12 possible roots) and a degree. This process is described in [17]. We consider, for 3 basic scales (major, harmonic minor (hMinor) and melodic minor (mMinor), see Figure 12) all the scale-tone chords built by stacking up a number n of thirds, e.g.:

   Eb M7 ($n = 3$): [**I** of Eb Major, **VI** of G hMinor, **V** of Ab Major, **V** of Ab hMinor, **V** of Ab hMinor, **IV** of Bb Major, **IV** of Bb mMinor]

   C m7b5 ($n = 4$): [**VII** of Db Major, **VII** of Db mMinor, **VI** of Eb mMinor, **IV** of G hMinor, **II** of Bb hMinor]

2. **Definition of a dynamic programming problem**
   We introduce a cost function which assigns the following cost to a given transition between two harmonic analysis:

```
transitionCost(HarmonicAnalysis x, HarmonicAnalysis y) {
    if (x.getScale().equals(y.getScale()))
        return x.getDegree();
    else
        return 20 + x.getDegree();
}
```

3. **Computation of an optimal solution**
   In order to model the fact that tunes usually loop over themselves, we add the first chord at the end of the sequence.

As an example, Table 1 shows the possible analysis for each chord of the Giant Steps chord sequence. The dynamic approach described here produces the analysis shown in Table 2, which can be considered perfect on that example.

2.5.4. In-Scale playing rate

The harmonic analysis is not just an exercice for musicologist. It is in fact capital for the jazz musician because it defines the sequence of scales over which he can play without having to modulate. Indeed, the harmonic analysis defines a chord as the degree of a

B	**V** Eb hMin	**V** E Maj	**V** E hMin	**V** E mMin	**III** Gb Maj	**III** Gb mMin	**I** B Maj
D 7	**I** D Maj	**V** Gb hMin	**V** G Maj	**V** G hMin	**V** G mMin	**IV** A Maj	**IV** A mMin
G	**V** C Maj	**V** C hMin	**V** C mMin	**IV** D Maj	**IV** D mMin	**I** G Maj	**VI** B hMin
Bb 7	**VI** D hMin	**V** Eb Maj	**V** Eb hMin	**V** Eb mMin	**IV** F Maj	**IV** F mMin	**I** Bb Maj
Eb	**I** Eb Maj	**VI** G hMin	**V** Ab Maj	**V** Ab hMin	**V** Ab mMin	**IV** Bb Maj	**IV** Bb mMin
A m7	**VI** C Maj	**V** Db hMin	**IV** E hMin	**III** F Maj	**II** G Maj	**II** G mMin	**I** A hMin
D 7	**I** D Maj	**V** Gb hMin	**V** G Maj	**V** G hMin	**V** G mMin	**IV** A Maj	**IV** A mMin
G	**V** C Maj	**V** C hMin	**V** C mMin	**IV** D Maj	**IV** D mMin	**I** G Maj	**VI** B hMin
Bb 7	**VI** D hMin	**V** Eb Maj	**V** Eb hMin	**V** Eb mMin	**IV** F Maj	**IV** F mMin	**I** Bb Maj
Eb	**I** Eb Maj	**VI** G hMin	**V** Ab Maj	**V** Ab hMin	**V** Ab mMin	**IV** Bb Maj	**IV** Bb mMin
F# 7	**III** Db Maj	**III** Db mMin	**VII** Gb Maj	**V** Bb hMin	**V** B Maj	**V** B hMin	**V** B mMin
B	**V** Eb hMin	**V** E Maj	**V** E hMin	**V** E mMin	**III** Gb Maj	**III** Gb mMin	**I** B Maj
F m7	**IV** C hMin	**III** Db Maj	**II** Eb Maj	**II** Eb mMin	**I** F hMin	**I** F mMin	**VI** Ab Maj
Bb 7	**VI** D hMin	**V** Eb Maj	**V** Eb hMin	**V** Eb mMin	**IV** F Maj	**IV** F mMin	**I** Bb Maj
Eb	**I** Eb Maj	**VI** G hMin	**V** Ab Maj	**V** Ab hMin	**V** Ab mMin	**IV** Bb Maj	**IV** Bb mMin
A m7	**VI** C Maj	**V** Db hMin	**IV** E hMin	**III** F Maj	**II** G Maj	**II** G mMin	**I** A hMin
D 7	**I** D Maj	**V** Gb hMin	**V** G Maj	**V** G hMin	**V** G mMin	**IV** A Maj	**IV** A mMin
G	**V** C Maj	**V** C hMin	**V** C mMin	**IV** D Maj	**IV** D mMin	**I** G Maj	**VI** B hMin
C# m7	**VII** Db hMin	**VII** Db mMin	**VI** E Maj	**V** F hMin	**III** Ab hMin	**III** A Maj	**II** B Maj
F# 7	**III** Db Maj	**III** Db mMin	**VII** Gb Maj	**V** Bb hMin	**V** B Maj	**V** B hMin	**V** B mMin
B	**V** Eb hMin	**V** E Maj	**V** E hMin	**V** E mMin	**III** Gb Maj	**III** Gb mMin	**I** B Maj
F m7	**IV** C hMin	**III** Db Maj	**II** Eb Maj	**II** Eb mMin	**I** F hMin	**I** F mMin	**VI** Ab Maj
Bb 7	**VI** D hMin	**V** Eb Maj	**V** Eb hMin	**V** Eb mMin	**IV** F Maj	**IV** F mMin	**I** Bb Maj
Eb	**I** Eb Maj	**VI** G hMin	**V** Ab Maj	**V** Ab hMin	**V** Ab mMin	**IV** Bb Maj	**IV** Bb mMin
C# m7	**VII** Db hMin	**VII** Db mMin	**VI** E Maj	**V** F hMin	**III** Ab hMin	**III** A Maj	**II** B Maj
F# 7	**III** Db Maj	**III** Db mMin	**VII** Gb Maj	**V** Bb hMin	**V** B Maj	**V** B hMin	**V** B mMin

Table 1. Giant Steps chord sequence with possible analysis for each chord

B	**I**	of B Major
D 7	**I**	of D Major
G	**IV**	
Bb 7	**I**	of Bb Major
Eb	**IV**	
A m7	**II**	of G Major
D 7	**V**	
G	**I**	
Bb 7	**I**	of Bb Major
Eb	**IV**	
F# 7	**V**	of B Major
B	**I**	
F m7	**II**	of Eb Major
Bb 7	**V**	
Eb	**I**	
A m7	**II**	of G Major
D 7	**V**	
G	**I**	
C# m7	**II**	of B Major
F# 7	**V**	
B	**I**	
F m7	**II**	of Eb Major
Bb 7	**V**	
Eb	**I**	
C# m7	**II**	of B Major
F# 7	**V**	

Table 2. Result of the harmonic analysis for Giant Steps

tonic chord when the content of their scales are almost identical. The player can thus improvise on a D Major and still *sound* like a G as degree IV of D. The three scales used to characterize the modulations are defined in Figure 12.

For this purpose, the system uses the result of the harmonic analysis (i.e. the sequence of modulation scales) to verify that the transcribed solo fits with it. Each modulation covers a set of bars over which we compute the rate of played notes that belong to the scale. This so-called *In-scale rate* is provided as feedback to the user, along with the transcription of his solo.

Figure 13 shows an example of solo transcription, that also shows the chord sequence of the lead sheet, the harmonic analysis (each modulation is indicated by a trans-

(a) Major (b) Harmonic Minor (c) Melodic Minor

Figure 12. Three basic scales of the harmonic analysis

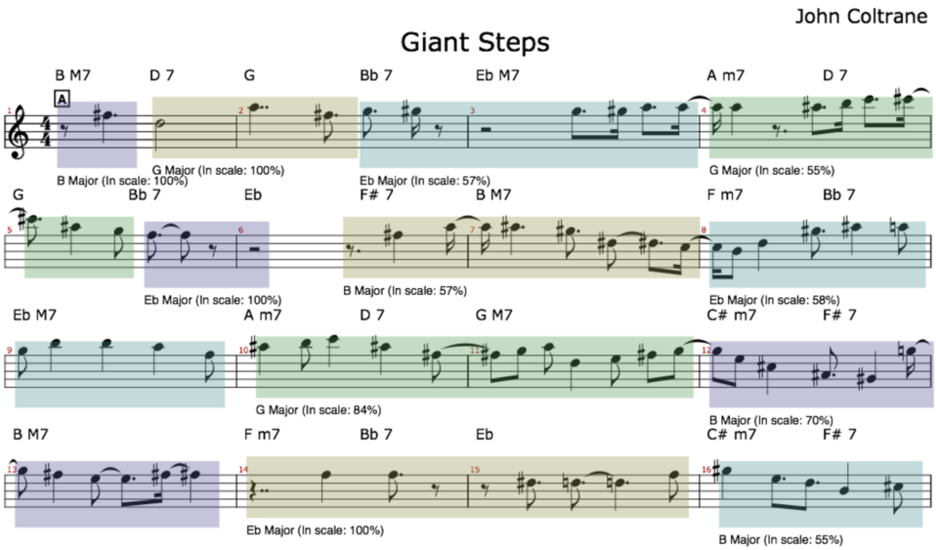

Figure 13. Result of the solo transcription process, displayed as a lead sheet score.

parent colored bar) and the *In-scale* playing rate, indicated for each modulation area.

3. Discussion

Social Virtual Band raises many issues related to jazz pedagogy and its social aspects. The system described here is a first step to provide the learning musician with an automatic feedback on his performance. Our focus here is to provide a measure of the student's skills estimated on a solo he performed. The in-scale rate, described herebefore, is an important measure that will soon be completed by the on-beat rate and the continuity measure, as defined in Section 1.3.

However, it is crucial to determine how this feedback does impact the evolution of the music student's skills. Such information will help us define future directions for music pedagogy support, based on automatic feedback. Future experiments are planned that will involve jazz students (from jazz schools) practicing improvisation with *Social Virtual Band* during a long period of a few months. Interestingly, the guitar practice video game Rocksmith 2014 claims that anyone can learn guitar with their system within only two months by playing one hour a day. Contrary to Rocksmith, *Social Virtual Band* is not designed as a substitute to the music teacher, but rather as a complementary support, that would be ideally nested within a social network.

Indeed, providing automatic feedback to the music student is a true innovation in the field of music pedagogy, since most existing systems tend to only provide an environment for practicing music. Nevertheless, no matter how relevant a machine can get, feedback from one's peers with always prevail, especially from trustworthy contacts such as a teacher or an experienced friend. The platform presented here is ideally suited for the emergence of such a social network, based on music practice, because it provides all the tools needed: a portable playalong system, a cloud server for archiving one's solo collection, and automatic tools to provide feedback on such collections. The social dimension will turn the solo into a social object, just like text, photos and audio tracks are today, thanks to existing social networks.

4. Conclusion

This chapter introduced the first system for providing support and feedback for practicing jazz improvisation. While most existing play-along softwares only provide prerecorded accompaniment to train on, *Social Virtual Band* extends that experience by collecting the history of recorded solos and embedding them inside a dedicated social network. As shown in this chapter, the collection and analysis of solos implies solving several technical issues, e.g. the precise synchronization of the recorded solo with the accompaniement, or the accompaniement cancellation in the recording process. The social ecosystem built around the solos, allows the musician to receive feedback on his skills, both from his community and from ad-hoc automatic analysis. We presented here an example of relevant automatic measure of the solos quality, based on the comparison of the transcribed notes with the expected scales deduced from harmonic analysis.

Nevertheless, two key issues remain open, that will be considered through long-term experiments involving a first community of users: what makes a good solo? and what kind of advice can make one improve his skills? Both questions will find answers from the analysis of social exchanges in tutor-teacher pairs.

5. Acknowledgements

This research is conducted within the Flow Machines project which received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 291156, as well as the Praise project (EU FP7 number 388770), funded by the European Commission under program FP7-ICT-2011-8.

We also thank and Daniel Martín and Timotée Neullas for their contribution to the web services.

References

[1] Jakob Abesser, Klaus Frieler, Martin Pfleiderer, and Wolf-Georg Zaddach. Introducing the jazzomat project - jazz solo analysis using music information retrieval methods. In *10th International Symposium on Computer Music Multidisciplinary Research, CMMR 2013*, pages 653–661, Marseille, Octobre 15-18th 2013.

[2] Nancy Bertin, Roland Badeau, and Emmanuel Vincent. Fast bayesian nmf algorithms enforcing harmonicity and temporal continuity in polyphonic music transcription. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 29–32, New Paltz, New York, USA, October 2009.

[3] Michael A. Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, April 2008.

[4] Jerry Coker. *Improvising Jazz*. "Simon & Schuster", 1964.

[5] Alain de Cheveigné. Multiple f0 estimation. In DeLiang Wang and Guy J. Brown, editors, *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*. John Wiley and sons, Inc., September 2006.

[6] Alain de Cheveigné and Hideki Kawahara. Yin, a fundamental frequency estimator for speech and music. *Acoustical Society of America Journal*, 111(4):1917–1930, avril 2002.

[7] Robert R. Faulkner and Howard S. Becker. *Do you know ? The Jazz Repertoire in Action*. MIT PRess, 2009.

[8] Jon Gillick, Kevin Tang, and Robert M. Keller. Machine learning of jazz grammars. *Computer Music Journal*, 34(3):56–66, 2010.

[9] M. Gladwell. *Outliers: the Story of Success*. Back Bay Books, 2011.

[10] Stanislaw Gorlow, Mathieu Ramona, and François Pachet. SISO and SIMO accompaniment cancellation for live solo recordings based on short-time ERB-band Wiener filtering and spectral subtraction. 2015. submitted.

[11] Thomas Hedges, Pierre Roy, and François Pachet. Predicting the composer and style of jazz chord progressions. *Journal of New Music Research*, 43(3):276–290, 2014.

[12] M.B. Holbrook. *Playing the Changes on the Jazz Metaphor*. Foundations and trends in marketing. Now Publishers, Incorporated, 2008.

[13] Gary Kennedy and Barry Kernfeld. *The new Grove dictionary of jazz, vol. 1 (2nd ed.)*. New York: Grove's Dictionaries Inc., 2011.

[14] Mark Levine. *The Jazz Theory Book*. Sher Music Company, 1995.

[15] PG Music. Band-in-a-Box. http://www.pgmusic.com/.

[16] Tan Hakan Ozaslan, Enric Guaus, Eric Palacios, and Josep Lluis Arcos. Identifying attack articulations in classical guitar. In *Computer Music Modeling and Retrieval. Exploring Music Contents*, volume 6684, pages 219–241. Springer Verlag, 2011.

[17] François Pachet. An object-oriented representation of pitch-classes, intervals, scales and chords: The basic muses. In *Proceedings of Journées d'Informatique Musicale (JIM)*, Bordeaux (France), 1994. Université de Bordeaux.

[18] François Pachet. Computer analysis of jazz chord sequences: Is it Solar a blues ? In E. Miranda, editor, *Readings in Music and Artificial Intelligence*. Harwood Academic Publishers, 2000.

[19] François Pachet and Pierre Roy. Analytical features: a knowledge-based approach to audio feature generation. *EURASIP Journal on Audio, Speech, and Music Processing*, 2009(1), February 2009.

[20] François Pachet and Pierre Roy. Markov constraints: steerable generation of markov sequences. *Contraints*, 16:148–172, March 2011.

[21] François Pachet, Jeff Suzda, and Daniel Martin. A comprehensive online database of machine-readable leadsheets for jazz standards. In *Proc. ISMIR '13*, pages 275–280, Curitiba (Brazil), November 2013.

[22] Geoffroy Peeters. A large set of audio features for sound description (similarity and classification) in the CUIDADO project. Technical report, IRCAM.

[23] Pierre Roy and François Pachet. Enforcing meter in finite-length markov sequences. In *Proceedings of AAAI*, July 14-18 2013.

[24] Justin Salamon, Emilia Gomez, Daniel P.W. Ellis, and Gaël Richard. Melody extraction from polyphonic music signals: Approaches, applications and challenges. *IEEE Signal Processing Magazine*, 31(2):119–134, March 2014.

[25] Paris Smaragdis and Judith C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 177–180, New Paltz, New York, USA, October 19-22 2003.

[26] Mark J. Steedman. A generative grammar for jazz chord sequences. *Music Perception*, 2(1):52–77, Fall 1984.

[27] D. Sudnow. *The ways of the hand. The organization of improvised conduct*. MIT PRess, 1993.

[28] Technimo. iReal Pro. http://irealpro.com/.

[29] Richard F. Voss and John Clarke. ”1/f noise” in music: Music from 1/f noise. *The Journal of the Acoustical Society of America*, 63:258–261, 1978.

[30] Chunghsin Yeh, Axel Röbel, and Xavier Rodet. Multiple fundamental frequency estimation and polyphony inference of polyphonic music signals. *IEEE Trans. on Audio, Speech and Language Processing*, 18(6):1116–1126, August 2010.