A Computational Framework Enhancing Jazz Creativity

Robert M. Keller and **David Morrison** and **Stephen Jones** and **Belinda Thom** and **Aaron Wolin** [1]

Abstract. A computational approach is described for helping jazz improvisors create solos. Initial facets of the approach have been embodied in a freely-available software tool called Impro-Visor ("Improvisation Advisor") which we developed. The paper describes the rationale and some of the computational challenges and issues associated with designing and implementing such a tool.

1 INTRODUCTION

Conventional wisdom toward educating jazz musicians in improvisation involves having them transcribe note-for-note recorded solos of famous artists. While this approach is no doubt contributory to the goal, it lacks an element of student ownership in that, regardless of how good it may sound, the student is ultimately parroting another soloist. The branching-off point, where the student begins inventing his or her own phrases to construct a solo may still pose challenges. There is also the issue of cost/benefit ratio, as transcription takes a great deal of time to get right, even if aided by software tools.

We have been exploring an alternative approach over the last several years in teaching a jazz improvisation course: have students compose solos off-line and expect them to be able to play them in class. How much of what the student composes is ultimately used in performance is up to him or her, but is expected that at least some fragments would likely be used, if from nothing more than latent memory. Moreover, composition requires a greater understanding of tune harmonic changes than does transcription, and performing such composition on-line is what improvisation is all about. Therefore being forced to understand harmonic aspects of a tune by composing a solo compatible with its harmony is expected to contribute a great deal to the ability to improvise on the tune.

The rest of the paper describes several technical aspects of a software tool, called Impro-Visor [3] intended to help musicians construct solos. We discuss some of the musical underpinnings and how they are used in the tool, followed by a discussion of issues currently being addressed in our research and development.

2 RELATED WORK

A widely-used commercial software tool is Band-in-a-Box [7]. It can generate complete jazz solos, but does not engage the user in their construction, other than as the provider of the chord changes. Consequently, the user does not learn from the process of solo generation. The methods employed by Band-in-a-Box are proprietary, but experience suggests that a preset database of melodic sequences is involved, rather than an algorithmic method for the generation of such sequences.

[1] Harvey Mudd College, Claremont, California, USA, email: improvisor@cs.hmc.edu

Notable other work includes Biles' GenJam [1] and the work of Papadopoulos and Wiggins [8] using genetic algorithms, which require some kind of fitness function to perform genetic selection, and Thom's BoB [9], which uses a statistical learning approach.

3 SOFTWARE SUPPORT FOR CREATIVITY

A jazz solo consists of a melody composed over a given chord sequence (known as "the changes" to the musician). The changes are chords chosen to support the original melody of a song, and usually are somewhat standardized, although not necessarily unique for that song. Common practice would be for the group to play the original tune (called the "head"), then each soloist would improvise some number of "choruses" over the changes, then the group would typically play the head a final time.

The challenge for the soloist is to have sufficient creative ideas to be able to play one or more choruses. Good preparation for this kind of creation entails knowing the melody and changes, as well as the theoretical underpinnings of the changes.

The educational tool that we have developed is designed to assist the budding soloist, who may not yet have a strong theory background, in the process of creating solos. Our intention is to provide this as a practical end-user tool, not simply a research vehicle, so we have included niceties such as MIDI-playback for usability. For context, Figure 1 provides a screen-shot of Impro-Visor.

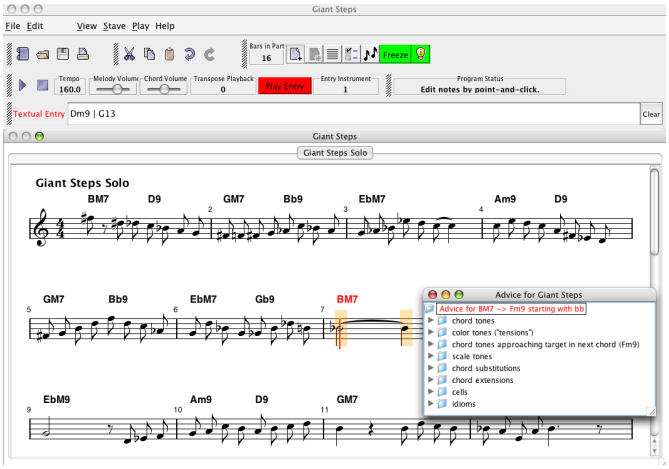

Figure 1. Screen shot of Impro-Visor

3.1 User Interaction

We want the user to be able to move freely through a chorus, filling in melodic parts as knowledge or inspiration dictate. Our approach to providing this ability is to display the full chorus as a "lead sheet", the term for a single melody line with chord symbols above. The user can then select any focal point and include notes by point-and-click. We have also developed an easy-to-use textual notation that combines both melody and chords, which provides an alternate form of entering melodic information.

While the user is free to enter notes based on his or own devices, the tool provides various types of assistance on request, based on the chord that is in force at the selected insertion point (shown to the left of the dialog box in Figure 1).

The categories of assistance include:

1. **Spelling** of the current chord.
2. **Color tones** for the current chord. These are tones that are not in the chord, but which are individually sonorous with it. They are also often called "tensions".
3. **Approach tones**: non-chord tones that make good transitions to chord-tones. When a second chord follows sufficiently closely, say within the current or next measure, there is an additional option available: tones within the current chord that approach tones in the next chord. (See *Voice Leading* in the next section.)
4. **Scales** that are compatible with the current chord, and the notes of any selected scale.
5. Chords that **substitute** for the current chord, so that comparable information can be accessed as for this chord as an alternative.

Figure 2 shows the advice menu of Figure 1 opened to a particular scale option.

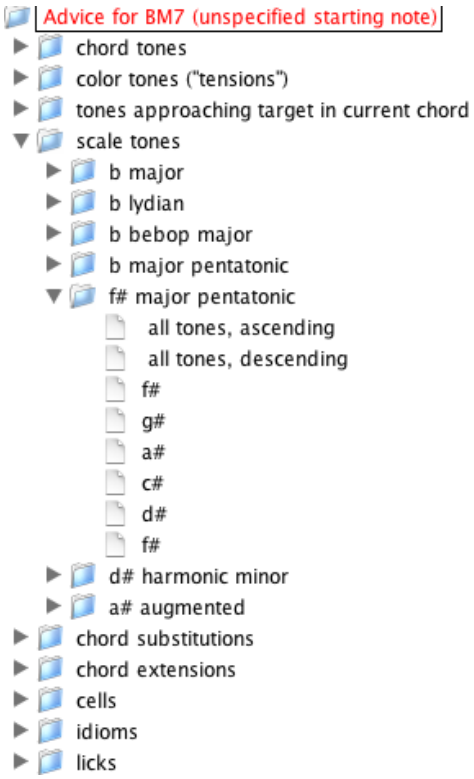

Figure 2. An expanded advice menu

3.2 Teacher/Administrator Support

As can be seen, there is quite a lot of relevant information that goes with a chord. One of the challenges faced was providing this information in a form that was easily changed or extended by the teacher or administrator. Another was to have as little redundancy in the information specification as possible. Accordingly, here are some of the ideas we used to meet these challenges:

1. The spelling for each chord and scale is given only with C as the tonic. The tool automatically translates them to other tonics as needed.
2. With each chord may be identified zero or more *extensions*, chords that include the same tones as the given chord, but add other tones. The notes that are added in extensions automatically become color tones of the given chord, but additional color tones can be specified.

Information regarding chords and scales is part of a single *vocabulary file* with entries in the form of S-expressions [5] so that the file can be easily read and modified outside the tool if desired. Figure 3 provides a glimpse at how this information is encoded. The sub-expressions under *approach* have the chord tone followed by the tones approaching it. Normally these are regarded as the tones one half-step below and one or one-half step above.

The *priority* attribute is used to suggest the relative importance of tones in the chords, which is theoretical knowledge. We also use priority to automate *voicing* in the played-back chord accompaniment. Voicing entails determining a sonorous stacking order for the chord

```
(chord
    (name CM7)
    (pronounce C major seven)
    (family major)
    (spell c e g b)
    (priority b e g c)
    (approach
        (c c# d) (e eb f)
        (g f# g#) (b bb c))
    (extensions CM9 CM7#11 CM7add13)
    (scales
        (C major)
        (C lydian)
        (C bebop major)
        (C major pentatonic)
        (G major pentatonic)
        (E harmonic minor)
        (B augmented)
       )
    (substitute CM69 Em7 Am9)
   )
(scale
    (name C major pentatonic)
    (spell c d e g a c))
```

Figure 3. Vocabulary chord specification

tones. It is also intimately related to *voice-leading* in harmony. We do not have space to expound on the harmonic aspect, but discuss voice-leading for melody in the next section. A simple but effective algorithm developed is to stack the first so many notes (say four or five, which may be specified by the user) from the priority list in the first octave beginning a certain distance above the root. This has the nice feature of automatically creating voice-leading on the inner voices of the chord progression.

In addition to the basic information listed above, we provide sets of *pre-constructed melodic segments* that the user can use as is or modify. These segments are provided by the administrator or teacher, but can be augmented by the student herself. They are classified into the following categories:

1. **Cells** are the most basic category. They are sequences of notes, usually of the same duration and associated with a single chord.
2. **Idioms** are like cells, but are identifiable sequences that have been recorded. Consequently, their use is cautioned if novelty is desired.The notes don't necessarily have the same duration.
3. **Licks** are coherent melodic sequences over one or more chords.
4. **Quotes** are melodic sequences from familiar songs or famous jazz solos. Using a quote within ones solo is regarded as a form of humor by the jazz musician and knowledgeable listener.

If there already is a note at the selected position, the tool restricts the sequences offered to ones that contain that note at the indicated position. As with the chord and scale repertoire, the melodic segments are entered only relative to one first chord, then automatically translated to others by the tool.

As with chord and scale definitions, sequences are also part of the vocabulary file, represented as S-expressions. An example is:

```
(lick
    (notes ab-8 c8 f8 d8 e8 ab8 c+8 e+8
         bb8 a8 ab8 g8 f#8 f8 eb8 b-8 d8)
    (sequence Dm7b5 | G7b9)
    (name minor 2-5))
```

This defines the lick, as it would be rendered in the key of C minor, as shown in Figure 4.

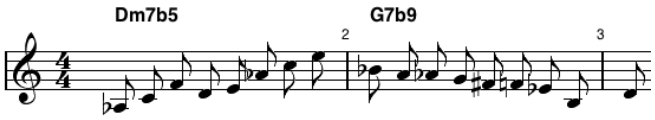

Figure 4. Example of a vocabulary lick

To give a little more explanation to the notation for notes, the first letter, possibly followed by # for sharp or *b* for flat, gives the pitch class. Any immediately following + or - characters raise or lower the actual pitch from the octave above middle-C. The number 8 indicates an eighth-note, 4 would indicate a quarter-note, etc. Duration values can be added together using further + characters, and there is a notation for dotted notes and triplets as well.

4 THEORETICAL ASPECTS

4.1 Voice Leading

One of the main examples of such a theoretical basis is the concept known as "voice-leading". This entails creating a melody in which the notes of the melody align with chord tones in the underlying changes.

A very common example of voice leading can be seen in the chord sequence known as II-V ("two five"), meaning the chord built on the second note of the scale followed by the chord built on the fifth note. In the key of C major, for example, the II would be a Dm7 (D minor-seven) chord and the V would be a G7 (G seven) chord. To illustrate voice leading, it is common in jazz to extend those chords to a Dm9 (D minor-nine) and a G13 chord. Setting aside the bass notes for the moment, the essential notes of these chords can be closely aligned to illustrate voice leading:

$$Dm9 : FACE \quad (1)$$

$$G13 : FABE \quad (2)$$

(In the G13 the 5th, D, is often omitted from the voicing, but typically is still heard as part of the overtone series of the root G.) As can be seen, the only difference between (1) and (2) is C vs. B. Hence in exploiting voice leading in the solo, one might choose a melodic line in which the C appears, followed by B. In musical terminology, this is termed "resolution" of the dissonance between the root D and C in the Dm9 chord, since the D-C interval of a minor seventh is more dissonant than the D-B interval of a major sixth. (A stronger F-B dissonance is introduced consequently, and that would normally be resolved in a separate step after the G13 chord, e.g. by changing to a C chord.) Figure 5 illustrates this type of resolution in a jazz melody.

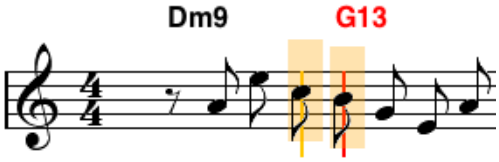

Figure 5. Resolution of the D-C interval (D is not shown, but implied as the root of Dm9) by a C to B transition

This succession could be intermediate or one or more notes could be interposed between them, as shown in Figures 6 and 7.

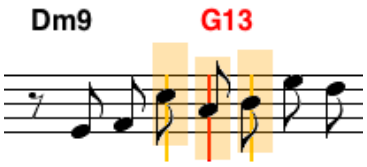

Figure 6. Resolution by C-B, delayed by one note

In both the direct and deferred resolution cases, the chosen notes resonate with the underlying harmony, which is usually being articulated by the accompanying rhythm section while the soloist is playing.

4.2 Modularity

Impro-Visor exploits *modularity*, by which we mean that melodic lines can often be broken into parts, then mixed or matched with parts from other to form new lines. This is similar in some ways to

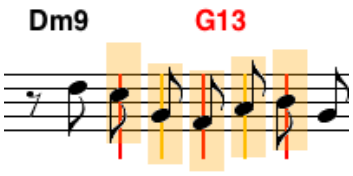

Figure 7. Resolution by C-B, delayed by more than one note

children's *mix-and-match* books that make silly faces, creatures, or stories by combining segments of normal pictures or stories that are offered in parallel (*cf.* [6]). An early musical equivalent can be found in Mozart's dice game [2].

A benefit of modularity is *combinatorial economy*. If M different cells over one chord can be combined with N cells over another chord, we can obtain MN melodic segments using storage space of just M + N. At the present time, we have not fully exploited this benefit, insofar as the repertoire of licks over two chords is not currently represented as combinations of sub-licks. However, the user is constantly exploiting modularity in focusing on providing melodies for a small number of chords, typically one or two, at a time, some of which can be constructed from cells.

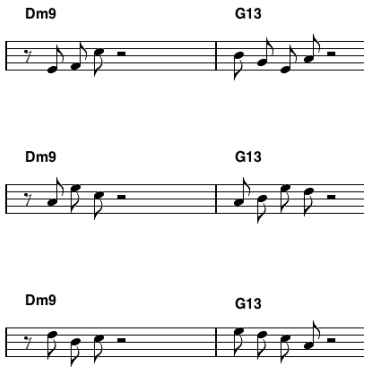

Figure 8. Three cells over each of two chords

Figures 8 and 9 illustrate this form of modularity. Figure 8 shows three cells for Dm9 on the left and three cells for G13 on the right. If the ending note of the Dm9 cell has the right interval in relation to the starting note of the G13, we can juxtapose the two cells to get a lick. In present example, we may be able to get as many as nine good-sounding licks from those six cells, and in fact we do get this many, the Cartesian product shown in Figure 9.

The main issue to resolve here is what constitutes an acceptable interval connecting two cells. A half-step apart is very common and desirable in jazz. But other intervals such as two, three, or four half steps generally sound fine as well. Further research is needed to determine whether we can simply restrict to certain connecting intervals or that additional context is needed to establish connectability.

4.3 Pattern Matching

Pattern matching between chords occurring in the tune and chords indexing stored melodic segments is one of the major problems in developing this software, for the following reasons:

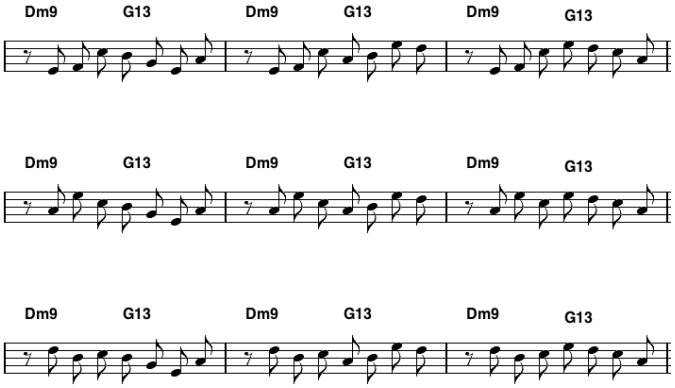

Figure 9. A product of nine licks generated from the six cells in Figure 8

1. A chord in the tune might not exactly equal a chord in the vocabulary segment, but rather imply it or be a discretionary alternative. For example, a Dm7 in a stored melodic segment would usually work with any of Dm7, Dm9, Dm11, Dm13 in a tune. Fortunately, these are *extensions* of the chord, and this information is represented in the vocabulary and thus is available for use in the matching algorithm.

2. A chord in the tune might have the chord in the vocabulary segment as *its* proper extension, or just be a cousin of the other chord. This case is not as straightforward as the above, because the tones assumed in the vocabulary chord might not work well in the context of the more limited chord in the tune. For this reason, we separate the two types of match when providing advice to the user, as follows:

   (a) with same first chord type (or having this type as an extension)

   (b) with related first chord type (or extending this chord type)

   Here the concept of *related* currently means having the same *family*, where the latter is optionally given for each chord in the vocabulary. For example, *dominant* is the family to which G7, G9, G13, G7b9, etc. belong. Complications are introduced by the possibility of some chords, in principle, being able to be in more than one family. For example, G7b9 could also be regarded as part of a *diminished* family, because the part above the root forms a diminished-seventh chord. A chord with no family specified defaults to a family of itself alone. At the moment, we assume only one family per chord, but plan to extend the matcher to multiple families in the future.

3. Chords are often used to *embellish* a chord sequence, in a non-essential way. The presence of embellishing chords gets in the way of pattern matching, because it may be unclear that they are serving an embellishing function. It is not hard for an experienced jazz musician to recognize this use, but it is another matter for software. Our plan for dealing with this problem is to design a non-intrusive annotation to the chord specification for tunes, which will enable embellishments to be stripped off in layers. For example, in Dm7 G7, the Dm7 can be regarded as an embellishment approaching the G7 chord. It is also common for a chord to be preceded by its relative V chord as an embellishment. So if Dm7 is present, there could be an A7 before it. Thus the notation might read {{A7} Dm7} G7,

meaning: Dm7 can appear before G7, and *if* it appears, an A7 can appear before that. Pattern matching in the presence of this kind of embellishment is yet to be worked out, but the concept is familiar in computer science for syntax specification. For a more detailed musical explanation, please refer to chapter 2 of [4].

4. **Inter-chord timing** is a major issue. A segment designed for a 1-measure Dm9 G13 won't generally work for a 2-measure version of the same chord sequence. This issue is complicated by the fact that chords are not always placed on beat boundaries in the published versions of songs. The chords themselves can be syncopated, sometimes at the level of half-beats, to align with the original melody.

5 ON AUTOMATING MELODY GENERATION

We would like our tool to provide a very large repertoire of melodic segments. Up until recently, our approach was to construct these by hand or transcribe them from recorded sources. By-hand construction is enjoyable, and turns out to be as educational, if not more educational, than using the segments to construct complete solos. However, by-hand construction is a slow process. It would be much better if we could automate the generation of these segments, either on-line while the tool is being used, or off-line, saving the segments in the vocabulary file in which such segments currently reside.

On-line generation is somewhat risky. Unless one has a very reliable algorithm for doing it, the results can be poor and therefore misleading to the student. Of course the ear can be used as the ultimate selection device, and it is easy to backtrack to a different sequence, but the beginner's ear is often not sufficiently developed, so it would be better to have every generated segment sounding good.

Creation of a melody generator is a topic of current research. We have been considering a variety of machine learning approaches. However, one associated problem is that there needs to be a corpus of segments available to train the learning mechanism. Moreover, there has to be a training set for each commonly-occurring chord sequence, which is a lot of sets. So in a way it is a chicken-and-egg problem. We are aware of the success of genetic approaches, but would like to find something different.

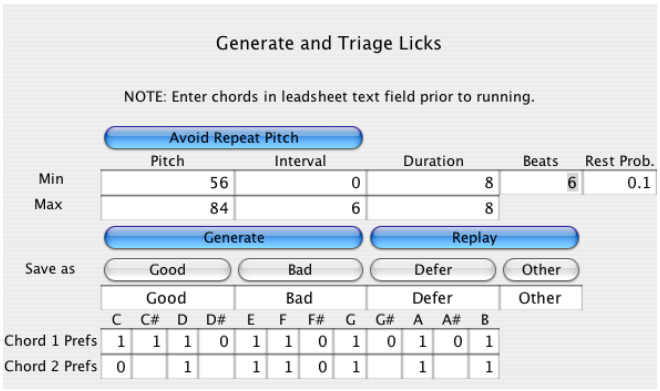

Figure 10. Lick generation and triage device

As a step toward creating a corpus, we constructed a "lick triage" device as part of Impro-Visor, as shown in Figure 10, from which we learned some things about the problem, as we now describe. The tool allows one to specify (unordered) sets of preferred tones over two-chord sequences, although the idea could easily be extended to more chords. The preferred tones are essentially scale tones and color tones, information that is already available, as described earlier. Minimum and maximum duration for notes are set by the user. When triggered by the user, the tool generates a random sequence of pitches within the realm of the parameter settings. The user can then indicate good or bad, and this indication is saved in the vocabulary along with the sequence itself. Our intention was to use the resulting corpus to train a machine learning method.

One surprise that resulted is that, when we specified *avoid repeated notes* and used a uniform duration of eighth-notes, most of the segments were acceptable, at a level above 90 percent. When the duration was not uniform, say a mixture of eighth and quarter notes, the acceptance rate was not as high, but still encouraging. Some interesting syncopated melodies were observed, consistent with the intended use for jazz. The acceptance rate also deteriorated when longer licks were stipulated. This suggests that rhythmic aspects are at least as important as pitch aspects when characterizing acceptable jazz melodies, which is perhaps not so surprising in retrospect.

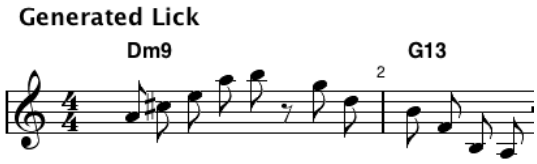

Figure 11. A lick generated by the device

We are encouraged by the results from fairly simple technology, and believe that it will be possible to develop a fairly proficient melody generation tool by algorithmic techniques. This will be the subject of future research. Options would include means of specifying the preference of certain intervals, such as fourths, which are generally regarded as "modern-sounding" by jazz musicians, for example, or the avoidance of certain intervals. Enforced pairing of notes of small duration is another trait that is probably desirable. A grammatical approach to melody generation is being considered in this regard.

6 EVALUATION

A fair number of solos have been constructed by the first author as demonstrations, and these are available for viewing and listening on the website [3] currently. At the time of writing, we have not had the opportunity to evaluate the approach fully in a course setting, since the author who teaches improvisation has been on sabbatical since the inception of the project. However, we plan to use the tool throughout the 2006-2007 academic year at Harvey Mudd College, and will be reporting on the usefulness of our tool in a future publication or on the Impro-Visor website.

7 CONCLUSION

We have described some of the motivation and workings behind a software tool for assisting students to learn jazz improvisation or for assisting intermediate players in becoming more proficient. We have also indicated future approaches toward automating the generation of useful melodic sequences to be made available in the tool.

ACKNOWLEDGEMENTS

This work was supported by a Faculty Enhancement grant from the Mellon Foundation.

REFERENCES

[1] J. A. Biles, 'Genjam: A genetic algorithm for generating jazz solos', in *Proc. of the 1994 International Computer Music Conference*, pp. 131–137, Aarhus, Denmark, (1994).
[2] Mozart's Dice Game. http://webplaza.pt.lu/public/mbarnig/pages/dicemus.html.
[3] Impro-Visor. http://www.cs.hmc.edu/~keller/jazz/improvisor/.
[4] Henry Martin, *Charlie Parker and Thematic Improvisation*, Scarecrow Press, Lanham, Maryland and London, 2001.
[5] John McCarthy, 'Recursive functions of symbolic expressions and their computation by machine', *Communications of the ACM*, 184–195, (1960).
[6] Norman Messenger, *Famous Faces*, Dorling Kindersley Publishing Inc., New York, 1995.
[7] PG Music. http://www.band-in-a-box.com/.
[8] George Papadopoulos and Geraint Wiggins, 'A genetic algorithm for the generation of jazz melodies', in *Proceedings of STeP 98*, Jyvskyl, Finland, (1998).
[9] Belinda Thom, 'BoB: An interactive improvisational music companion', in *Proceedings of the Fourth International Conference on Autonomous Agents*, eds., Carles Sierra, Maria Gini, and Jeffrey S. Rosenschein, pp. 309–316, Barcelona, Catalonia, Spain, (2000). ACM Press.