Generating guitar solos by integer programming

Nailson dos Santos Cunha*[1], Anand Subramanian†[1], and Dorien Herremans‡[2,3]

[1]Centro de Informática, Universidade Federal da Paraíba, João Pessoa–PB, Brazil
[2]Information Systems Technology and Design Pillar, Singapore University of Technology and Design, Singapore
[3]Centre for Digital Music, School of Electronic Engineering and Computer Science, Queen Mary University of London, UK

Abstract

In this paper, we present a framework for computer-aided composition (CAC) that uses exact combinatorial optimization methods to generate guitar solos from a newly proposed dataset of licks over an accompaniment based on the 12-bar blues chord progression. An integer programming formulation, which can be solved to optimality by a branch-and-cut algorithm, was developed for this problem whose objective is to determine an optimal sequence of a set of licks given a matrix of transition costs derived from user preferences. The generated solos are displayed in tablature format. Outputs of the system were evaluated in an empirical experiment with 173 participants. The results show that the solos whose licks were optimally sequenced were significantly more enjoyed than those randomly sequenced. We project that the developed framework could be of potential use to guitarists looking for original material; as an educational tool for future composers; and to support composers in discovering unique and novel compositional ideas.

Keywords— Optimization, computer-aided-composition, guitar solos, integer programming, automatic composition.

1 Introduction

Automatic music generation, or algorithmic composition, has gained a lot of interest recently. Events such as the recent release of Google's Magenta project, "a research project to advance the state of the art in machine intelligence for music and art generation" (Eck, 2016) are bringing it to the eye of the public. Although the field is relatively young, large strides have been made since its conception. In the 1950s, the first computer systems for automatic composition were developed. The most famous composition that resulted from this early work was the Illiac Suite, a string quartet composed using a rule-based approach combined with stochastic properties (Hiller and Isaacson, 1957). Despite the fact that the piece was a huge step forward, it also showed that much work remained to be done as it lacked complex structural elements such as repetition, phrases and functional harmony.

*nailsoncunha@sti.ufpb.br
†anand@ci.ufpb.br, anandsubraman@gmail.com
‡dorien_herremans@sutd.edu.sg, dorien.herremans@gmail.com

The framework for computer-aided composition (CAC) proposed in this paper tackles some of the remaining challenges in the field of automatic composition: long-term structure and generating expressive music. Combinatorial optimization methods are used to generate guitar solos over an accompaniment that is based on the 12-bar blues chord progression. This problem is somewhat related to previous work on jazz improvisation, e.g. GenJam (Biles, 1994), the Impro-Visor system developed by Keller and Morrison (2007) and the blues melody generation system by Hall and Smith (1996), as these systems also generate a single line of music over an accompaniment. In this research, however, we specifically work with *guitar music* and allow the solos to have *expressive properties* such as bends, slides, hammer-on and pull-off, and that are constrained to contain turnaround *cadence constraints*. Only a handful of papers have attempted to use integer programming for music generation (see Section 2). The approach implemented in this paper is the first to successfully solve the guitar solo generation problem with *integer programming*. The resulting guitar solos were evaluated through an empirical experiment and were shown to be more enjoyed than random solos. We project that the framework developed in this paper could support composers in exploring unique compositions; provide more diverse and original material to guitarists; and potentially be a didactic aid to future composers.

1.1 Computational methods for music generation

Currently, existing research on automatic composition systems can roughly be categorized into three groups: machine learning models, rule-based systems and optimization approaches. The first category is formed by the *machine learning models*, and encompasses techniques such as Markov models and deep learning. An early example of this category is the Markov model built on a corpus of 39 simple nursery rhyme melodies by Pinkerton (1956). In this system, a random walk method was used to generate new melodies. Later, more complex probabilistic models have been developed that incorporate more musical features (e.g. pitch contour and rhythmic patterns), such as the system implemented by Roig et al. (2014). Recently, there has also been some preliminary research on using deep learning models for generating music (Boulanger-Lewandowski et al., 2012; Herremans and Chuan, 2017), however, finished compositions with high-level structure have yet to be generated.

One of the seminal works on the guitar solo generation problem is due to McVicar et al. (2014a). They developed AutoLeadGuitar, a system that generates lead guitar solos with rhythm, pitches and bends (an expressive performance feature), based on a statistical model learned from a corpus of existing solos. A complementary research project by McVicar et al. (2014b), called AutoRhythmGuitar, focuses on creating the rhythm guitar part, a type of guitar part which usually accompanies a lead guitar or other solo instrument. The authors conducted a qualitative analysis of the generated results to confirm that the tablature represented realistic musical compositions and were mostly possible to play. Other than the work by McVicar et al., we are not aware of any work that is guitar-specific.

A second category of music generation systems, inspired by the work of Hiller and Isaacson (1957), is that of *rule-based systems*. One example of a rule-based system is the model for generating counterpoint developed by Schottstaedt (1984). Counterpoint is a type of renaissance music that is strictly defined by rules in music theory. Systems that contain many rules are

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

often combined with an optimization technique, in order to find solutions that adhere to all of the rules. This brings us to the last category, which considers music composition as a *combinatorial optimization* problem. This approach is adopted in this research, as it offers many advantages, such as the ability to easily constrain aspects of the music such as long-term structure, and cadences (Herremans et al., 2015). Adding constraints to a music generation system based on machine learning techniques is notoriously difficult (Pachet and Roy, 2011), however, optimization techniques offer a viable method to tackle this problem. In this research, we chose to work with optimization techniques combined with knowledge of music theory, in order to impose a larger structure, i.e. 12-bar blues, and cadence constraints on the music. In Section 2 a brief overview is given of how optimization algorithm have typically been applied to music generation problems.

1.2 Guitar licks and 12-bar blues

A guitar solo can be broken down into small melodic elements called *licks*, which can be defined as group of notes, pre-planned phrases or runs that a guitar player can use in solos over a particular chord or chord progression (Fisher, 1995). In this research, we use licks as the building blocks to generate complete and coherent guitar solos. One of the advantages of working with licks as a fundamental unit is that they already include complex expressive features such as *bends*, *slides*, *hammer-ons* and *pull offs*.

The proposed lick-based approach enables us to easily pose constraints on structure (12-bar blues) and other musical components. The 12-bar blues structure is a popular chord progression based predominantly on the I-IV-V chords of a key (e.g. C – F – G in the key of C) with the following structure: I-I-I-I-IV-IV-I-I-V-IV-I-V) (Krenz, 2010). The dominant seventh chord, which consists of the root, major third, perfect fifth, and minor seventh, is often used instead of the major chord in blues music.

In this research, licks are used as the building blocks for the formalized guitar solo generation problem, which in essence consists of scheduling a sequence of licks. We use a branch-and-cut procedure to solve this problem to optimality.

1.3 Structure of the paper

After a brief survey of existing work on music and operational research in Section 2, we construct a dataset of short solo fragments containing expressive features, i.e., *licks* (Section 3). A set of transition preferences is then described which determines the cost of playing a lick j immediately after lick i. Based on this, an optimization problem is defined in Section 4 that consists of finding the optimal sequence of licks that fits into the 12-bar blues form, given a number of constraints. This problem is formulated as an integer programming problem. In contrast to existing research, we implemented a branch-and-cut method in Section 5 that is able to find an optimal solution to the guitar solo generation problem based on a given set of licks. The resulting framework produces good results, which are verified in an empirical experiment described in Section 6.

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

2 Music generation and operational research

Many techniques that have thus far been used to solve music generation as an optimization problem include metaheuristics that are based on genetic/evolutionary algorithms and local search-based approaches. The first genetic algorithm for automatic music composition was implemented by Horner and Goldberg (1991) and focused on the problem of thematic bridging. Their system transformed an initial fragment into a final (given) fragment over a specified duration, using a set of operators. The resulting piece consists of a concatenation of all patterns that are created during this process. Another interesting application of genetic algorithms is GenJam (*Genetic Jammer*), created by Biles (1994), which composes monophonic jazz solos given a fixed chord progression. Unlike traditional optimization algorithms, the fitness function is not rule-based but feedback from a human mentor is received for each population member (Biles, 2003). This creates a huge bottleneck, often referred to as the *human fitness bottleneck* (Tokui and Iba, 2000).

Metaheuristics applied to the music generation problem are not limited to genetic algorithms. In 2007, Geis and Middendorf (2007) were the first to apply ant colony optimization to harmonize a melody. Simulated annealing was used by Browne and Fox (2009) to arrange pre-written motives according to a pre-specified tension profile computed by a neural network. Herremans and Sörensen (2012, 2013) developed a variable neighbourhood search algorithm to generate both first and fifth species counterpoint. In these systems, the objective function penalizes a solution when the music theoretical rules of counterpoint are violated. In extensions of this work, the objective function was constructed based on a Markov model, which was trained on existing music (Herremans et al., 2015), and a tonal tension model (Herremans and Chew, 2017).

While the metaheuristic techniques described above attempt to find a *good* solution within a fast computing time, there have also been some attempts at finding *exact* solutions to music composition problems. Tanaka and Fujii (2015) consider a musical piece as a sequence of patterns of chords, rhythms and melodies. They offer a mathematical formulation which focuses on different integer programming constraints for each of the patterns that control global structure. The work is currently limited to a theoretical description and for an implementation of a solution method they refer to future work. Recently, Tanaka et al. (2016a,b) proposed a mathematical formulation of a problem posed by the twelve-tone serial composer, Milton Babbitt, known as the *"all-partition array"*. This problem consists of finding a rectangular area of pitch-class integers that can be partitioned into regions whereby each region represents a distinct integer partition of 12. This is a hard problem of which very few solutions exist. Tanaka et al. (2016a) found a new solution to the all-partition array problem by using constraint programming. This was done by splitting this problem into two sub-problems and then joining them to form a complete solution. For a more complete overview of existing music generation systems and techniques, the reader is referred to Fernández and Vico (2013); Herremans et al. (2017).

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

3 Quantifying musical knowledge: licks and transition costs

Before going into detail on how the guitar solo generation problem is formalised, we will discuss the characteristics of the dataset of guitar licks used in this research, as well as the preferences adopted to estimate the transition costs between the licks.

3.1 Dataset of guitar licks

A blues licks dataset was built for this research based on licks presented in specialized music books (Fritts, 2012; Marshall, 1999). This dataset is available at `http://dorienherremans.com/guitar_licks_dataset`. The original licks have a duration ranging from 2 to 5 bars in 4/4 time and, except for the *turnaround* licks (see below). These original licks, from the textbooks, were broken down into several 1-bar licks when adding them to the dataset, so as to increase the level of diversity in the generated solos. All licks in the dataset are thus 1 bar long, except for the turnaround licks, which are 2 bars long. The resulting set of licks was converted to the MusicXML format, the open format standard for exchanging digital sheet music. The resulting blues licks dataset consists of 342 licks, all in the key of C.

There is no consensus in the literature when it comes to formally classifying blues licks. Except for some specific and well-established cases that systematically appear in some classification schemes such as the turnaround licks, each textbook usually follows their own convention of grouping the different types of licks. We propose a classification of the licks in our dataset into four groups. This classification, together with a number of simple and intuitive characteristics that govern blues guitar solos, were defined in collaboration with professional musicians (i.e. Zé Filho and Marcos Rosa) as well as by consulting specialized textbooks on how to play blues guitar (Fritts, 2012; Johnson, 2002; Schonbrun, 2003; Manzi, 2010; Pierce, 2015), in order to capture musical knowledge in the system. The occurrence characteristics of the types of licks in the dataset, as defined below, are described in Table 1.

Table 1: Occurrence of different types of licks in the dataset composed in this research.

	Regular	With pause			Repetition	Total
		Start	End	Both		
Fast blues	33	15	13	3	1	65
Moderate blues	93	46	53	5	19	216
Slow blues	14	18	9	5	3	49
Turnaround licks	10	2	0	0	0	12
Total	150	81	75	13	23	342

We discern four types of licks: turnaround licks; repetition licks; licks that begin or end with a pause; and regular licks.

a. Turnaround — licks that are played during the last two bars of the 12-bar cycle, more precisely, over the chords I and V, as illustrated in Figure 1. Their purpose is not only to finish the cycle itself, but also to serve as a transition to the next one. According to Johnson (2002), the 12-bar blues turnaround has the ability to simultaneously offer harmonic closure

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

and rebirth, and most of these licks are thus useful for both opening blues tunes as well as for ending them. The textbook by McCabe (2002) is entirely devoted to turnaround licks.

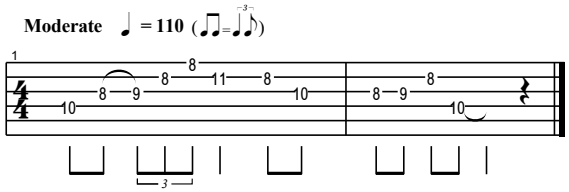
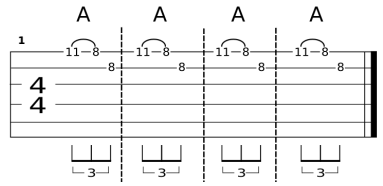
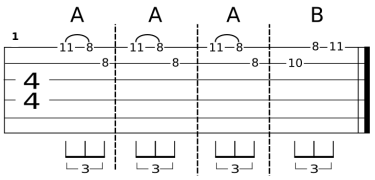


Figure 1: Example of turnaround lick from our dataset

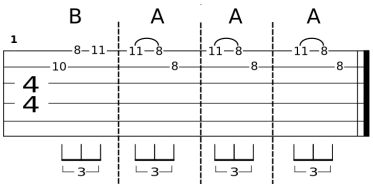
b. Repetition — licks that are based on the repetition of melodic figures (Scruton, 1999). Figure 2 shows the four cases of repetition licks that occur in the dataset used in this research. The vertical dashed lines in this figure subdivide the licks into four equal parts, each of them corresponding to a beat. In Figure 2a, all four parts are the same, resulting in an AAAA form. In Figure 2b and 2c the respective first or last fragment is different, thus forming an AAAB or BAAA form. Finally, Figure 2d shows the last case that occurs in the dataset, with an ABAB form.



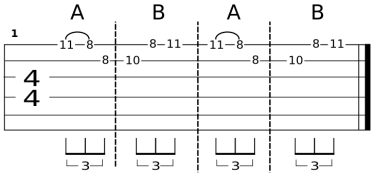
(a) Case 1



(b) Case 2



(c) Case 3



(d) Case 4

Figure 2: Example of the four types of repetition licks

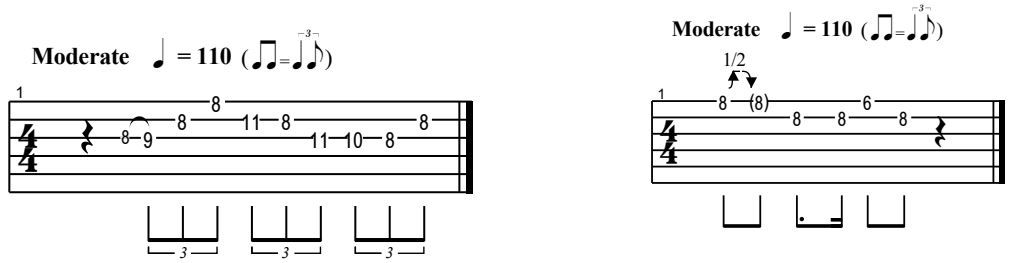
c. Licks that begin and/or end with a pause — These licks begin or end with a musical rest or silence in which no pitch plays either on the down beat or through the final beat of a bar. Figure 3 shows two examples of this type of lick.

d. Regular — all licks other than those described above.

3.2 Lick transition preferences

The above described classification system allows us to assign a transition cost between licks, which in turn will enable us to construct an objective function which evaluates the quality of

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.



(a) Lick that begins with a pause

(b) Lick that ends with a pause

Figure 3: Examples of licks that begin and/or end with a pause in our dataset

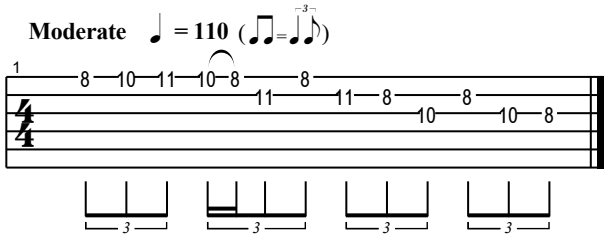


Figure 4: Example of a regular lick in our dataset

solos during the optimization process (see next section).

This transition cost is based on two types of preferences: *transition preferences* and *occurrence preferences*, as described in Tables 2 and 3 respectively. The former set of preferences takes into account the characteristics of a pair of licks (i, j) by assigning a penalty/bonus when lick j appears immediately after lick i in a solo, while the latter considers the particularities of the lick itself within the context of a full solo. In our convention a bonus is denoted by a negative cost and a penalty is represented by a positive cost. The bonus and penalty values described here are indicative values used by the system as default (and in the experiment), the user is free to change these to their own preference.

Table 2: Transition preferences between licks.

Preference	Transition	Type	Cost
T1	From a repetition lick to any other lick	bonus	-50
T2	From a lick that does not end with pause to any other that starts with a pause ≤ 1 beat	bonus	-15
T3	From a lick that ends with a pause ≤ 1 beat to any other that does start with a pause	bonus	-15
T4	From any lick i to any lick j where the first note of j is adjacent and in the same octave of the last note of j in the major/minor pentatonic scale and (e.g., i = C5 and j = D5)	bonus	-15
T5	From any lick i to any lick j where the first note of j is equal to the last note of i	bonus	-15
T6	From any lick that ends with a pause > 1 beat to any other lick that starts with a pause > 1 beat	penalty	+25
T7	From any lick that ends with a pause > 2 beats to any other lick	penalty	+25
T8	From any lick to any other one that starts with a pause > 2 beats	penalty	+25

The transition preferences from Table 2 are motivated as follows. Preference (T1) assigns

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

a bonus for any transition involving repetition licks, as our interviews with expert musicians revealed that these can be used in virtually any moment of the solo (except at the end of the 12-bar cycle). This is confirmed by Manzi (2010), who recommends the repetition of melodic fragments in a solo. Moreover, as pointed out by Fritts (2012), repeated figures are often effective at building and sustaining important moments of a solo, sometimes highlighting an instant of great intensity in the music, and they are widely used in blues and similar musical genres (e.g., variations of *rock*). Preferences (T2) and (T3) assign a bonus to short pauses between the transitions in order to favour a change of context throughout the melody, and to avoid an overload of notes in the solo as suggested by Manzi (2010). Furthermore, when discussing aspects of musicality in a blues context, Schonbrun (2003) highlights the importance of pauses with an analogy to blues singers who have to pause in order to take a breath. He emphasizes that the great blues and rock players use a *vocal-like approach* when playing their solos in an attempt to emulate the human voice and its improvisatory style. Preferences (T4) and (T5) aim at favouring the *smooth voice leading* idea (Willmott, 1994) using the pentatonic scale, which is one of the most popular scales and offers a good framework for improvising blues solos (Fritts, 2012). According to Schonbrun (2003), the sound of blues in fact started with singers who sang pentatonic melodies. The practice of voice leading is often considered as making music sound more 'pleasing' (Huron, 2016), which goes towards explaining its broader use in practice (Aldwell et al., 2010). Finally, as suggested by experts in the interviews we conducted, preferences (T6), (T7) and (T8) aim to avoid long pauses in the solo as they may create an unintended tension. It must be pointed out that not all long pauses are bad, as they can also provide excitement in anticipation of the next lick to be played (Manzi, 2010). Therefore, one could opt to impose a bonus instead of a penalty or just simply ignore this preference by neither penalizing nor providing a bonus for long pauses between a pair of licks. In the default preference settings, we chose to assign a penalty value.

Table 3: Occurrence preferences of a solo within the context of a full solo.

Preference	**Description**	**Quantity**
O1	Number of turnaround licks	$= 1$
O2	Number of repetition licks	$= 1$
O3	Number of licks that begin and/or end with a pause	≤ 3

The description of the second type of preference is given in Table 3. The allowed occurrence of turnaround licks is limited to one, as this type of lick always has to occur once at the end of the solo. Secondly, the number of repetition licks and licks that begin/end with a pause is limited to promote the diversification of the solo by stimulating the use of regular licks. The musical knowledge captured by the above scheme of preferences is implemented in the framework as constraints. The transition preferences can be interpreted as soft constraints, and they need not be satisfied, but at the same time they are directly responsible for the quality of the solo. The occurrence preferences, on the other hand, can be seen as hard constraints, meaning that they have to be satisfied for any generated solo. The respective default cost and occurrence constraints of each of the preferences, used later for the experiment in Section 6.2.1, are depicted in Tables 2 and 3. These values were defined after performing extensive interviews

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

with professional musicians and listening tests in different settings. They are customizable input parameters that will be used to formalize the objective function and constraints.

The following section describes how the problem of guitar solo generation can be seen as an optimization problem.

4 Guitar solo generation as a combinatorial optimization problem

Based on the values reported in Table 2, a cost matrix $\mathbf{p}$ can be built, which contains the transition costs between a given set of licks. To build this matrix, all values from $\mathbf{p}$ are initialized with a cost of 100. In the next step, each pair (i, j) of licks from the set is examined. If the transition from i to j falls under one of the preferences defined in the previous section, the corresponding entry p_{ij} of the matrix is updated by adding the associated penalty or by subtracting the bonus. The resulting cost matrix is given to the optimization model as input data.

The optimization problem associated with the guitar solo generation can be defined over a graph $G = (L, A)$, where L is the set of licks (vertices), including the dummy licks 0 and $n + 1$ that are used for modelling purposes, and A is the set of arcs connecting two licks, such that $A = \{(i, j) \mid i \neq j\}$. Also, let $L' = L \setminus \{0, n + 1\}$. We define T and R as the set of turnaround and repetition licks, respectively, and P as the set of licks that begin and/or end with a pause. Let p_{ij} be the transition cost between $i \in L$ and $j \in L$, and c_i be the duration of $i \in L$ expressed in number of bars. The objective is to find an ordered subsequence of licks that minimizes the sum of transition costs, while respecting the following constraints: (i) each lick can be used at most once; (ii) the solo must start and end at the dummy licks 0 and $n + 1$, respectively; (iii) the solo must have exactly b bars; (iv) there must be at most r repetition licks; and (v) there must be at most s licks that begin and/or end with pause.

Figure 5 shows a graphical representation of a sequence of licks as a solution of the guitar solo generation problem. The example depicts the 12-bar solo starting and ending at the dummy licks 0 and $n + 1$, respectively. It uses licks 4, 16, 23, 25, 7, 27, 13, 11, 26 and 14, in this order, to form the solo respresented in Figure 7. The numbers next to the arcs indicate the transition penalties. For clarity, only the arcs that are part of the solution are shown.

Let x_{ij}, $(i, j) \in A$, be a binary variable that assumes value 1 if the arc $(i, j) \in A$ is in the solution, and 0 otherwise, and let y_i, $i \in L'$, be a binary variable that assumes value 1 if the lick $i \in L$ is in the solution, and 0 otherwise. The integer programming based mathematical formulation can be written as follows.

$$\min \sum_{(i,j)\in A} p_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{j\in L\setminus\{n+1\}} x_{ij} = y_i \qquad \forall i \in L' \tag{2}$$

$$\sum_{i\in L\setminus 0} x_{ij} = y_j \qquad \forall j \in L' \tag{3}$$

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

$$\sum_{i \in L'} c_i y_i = b \tag{4}$$

$$\sum_{i \in R} y_i \leq r \tag{5}$$

$$\sum_{i \in P} y_i \leq s \tag{6}$$

$$\sum_{j \in L' \setminus T} x_{0,j} = \sum_{i \in T} x_{i,n+1} = 1 \tag{7}$$

$$x_{ij} + x_{ji} \leq 1 \qquad \forall (i,j) \in A,\ i < j \tag{8}$$

$$\sum_{i \in S} \sum_{j \in \overline{S}} x_{ij} \geq 1 \qquad \forall S \subseteq L',\ 2 < |S| < b \tag{9}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{10}$$

$$y_i \in \{0,1\} \qquad \forall i \in L' \tag{11}$$

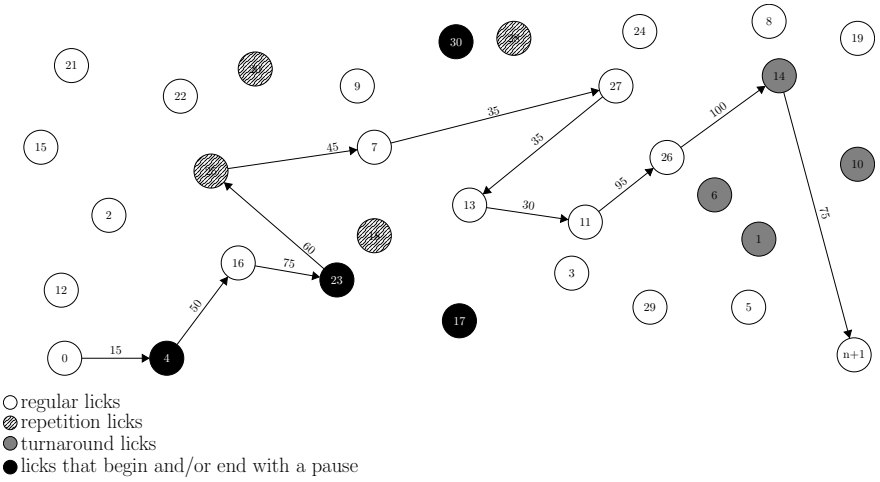


Figure 5: Example of a graphical representation of a sequence of licks as a solution of the guitar solo generation problem

The objective function (1) minimizes the transition costs between the licks. Constraints (2) and (3) state, respectively, that exactly one arc must leave and arrive at a lick that has been selected to be in the solo, thus ensuring the continuity of the solo and that each lick occurs at most one time. Constraint (4) imposes that the total duration of the licks (in bars) used in the solo must be equal to b. Constraint (5) limits the solo to have at most r repetition licks. Constraint (6) guarantees that the solo has at most s licks that begin and/or end with pause. Constraint (7) ensures that the solo starts with a lick other than a turnaround, and ends with a turnaround lick. Constraints (8) and (9) are the so called *subtour elimination constraints*, which in practice prevent the occurrence of multiple and disconnected solos. This class of constraints is well known in the travelling salesman problem (TSP) literature (Applegate et al., 2007). Because there are an exponential number of constraints (9), i.e., one for each possible subset S in set L', they are added on demand in a cutting plane fashion (Chen et al., 2009). Note that the number of constraints can be limited by only considering those subsets that are within the interval $2 < |S| < b$. This is because the number of licks in the solo will never exceed the

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

value of b and thus will never be subtours larger than b, and the particular case where $|S| = 2$ is already dealt by constraints (8). Finally, constraints (10) and (11) define the domain of the decision variables. The average CPU time for solving this optimization problem with CPLEX on a PC Intel core 2 quad with 2.00 GHz and 4 GB of RAM was 4 seconds.

When constraints (5) and (6) are relaxed, the resulting problem can be seen as a variant of the well-known *elementary shortest path problem with resource constraints* (ESPPRC) (Feillet et al., 2004; Irnich and Desaulniers, 2005), which is $\mathcal{NP}$-hard (Dror, 1994). In this class of problems, a *resource* can be associated with time, capacity and so on (Feillet et al., 2004). An *elementary path* corresponds to a path in which all vertices are pairwise distinct (Irnich and Desaulniers, 2005). In our case, the resource is related to the length of the solo expressed in number of bars. For a more detailed description of the ESPPRC problem, the reader is referred to (Feillet et al., 2004; Irnich and Desaulniers, 2005). The next section describes how this optimization problem is integrated in the guitar solo generation framework that is developed in this research.

5 Guitar solo CAC framework

The proposed framework, whose outline is shown in Figure 6, is composed of seven steps, that can be described as follows. Step 1 is responsible for receiving all of the input information, such as the licks database, the number of bars of the solo, the tempo of the song and the size of the instance (i.e., the number of licks to be provided to the optimization model), to name a few. In order to ensure a certain level of diversity, Step 2 selects a subset of licks at random according to the input data and parameter values provided in the previous step. The cost matrix is built from this subset during Step 3. The optimized solo is then generated in Step 4 by solving the integer programming formulation described in Section 4. In Step 5, an optional post-processing function is performed in order to adjust or remove undesirable tensions between notes. The generated output is exported in the MusicXML format in Step 6. Finally, the guitar solo is displayed in tablature form in Step 7. In the next subsections, we will discuss the main modules of the framework: preprocessing of the input data, optimization, and post-processing.

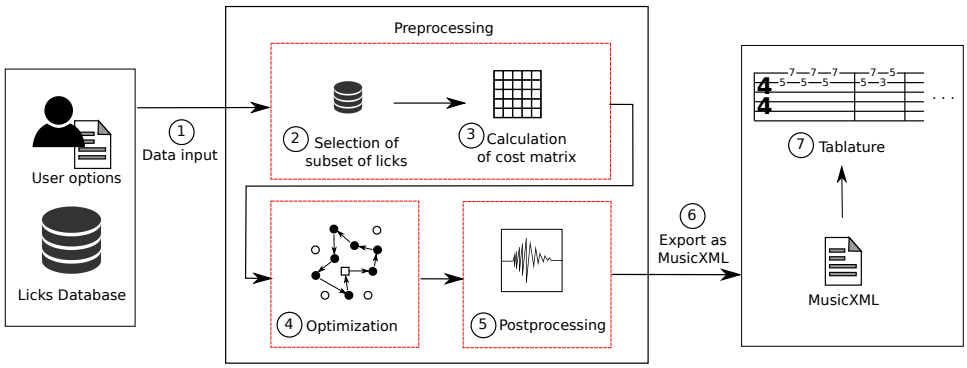


Figure 6: Guitar solo CAC framework

The overarching framework was implemented in *Python* 2.7 and the *lxml* library version 3.4 was used to manipulate texts and files in the XML format.

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

5.1 Preprocessing

The preprocessing module includes Steps 2 and 3 from Figure 6. Firstly, the subset of licks is selected by considering not only the size of the instances, but also the tempo, namely, 60 BPM (*slow blues*), 100 BPM (*moderate blues*) or 150 BPM (*fast blues*). Each of the licks from the database has a recommended tempo, as suggested in Fritts (2012); Marshall (1999). Therefore, if the user has specified the desired tempo of the solo as 100 BPM, then the subset will consist of random licks taken from the original database with this same tempo. Based on this subset of licks, the cost matrix $\mathbf{p}$ is calculated (Step 3) as described in Section 3 and sent to the optimization model together with the input information from Step 1.

5.2 Optimization

The optimization module receives the instance of the problem, which is composed of the information provided in Step 1, as well as the selected subset of licks and cost matrix defined in Section 3. It should be noted that the scheme of preferences and their respective penalty/bonus costs can be modified by the user. For example, a user may be indifferent to long pauses, in which case the associated preference can be removed by setting the cost of the corresponding penalty to zero. Alternatively, if the user enjoys long pauses, a negative value can be assigned so as to provide a bonus to favour this characteristic. The resulting optimization problem is then solved by means of a *branch-and-cut* algorithm implemented over the mathematical formulation described in Section 4. The best performing mixed integer programming (MIP) solvers typically implement this type of approach, which consists of embedding *cutting plane*-based procedures (those that add valid inequalities on demand) into a *branch-and-bound* algorithm. The latter is a well-known implicit enumeration method that is based on the divide and conquer strategy (Lawler and Wood, 1966). Most MIP solvers not only make use of default cutting plane schemes, but also allow the users to implement their own routines. In our case, we add the subtour inequalities (9) on demand as cutting planes using a standard min-cut based separation procedure (Stoer and Wagner, 1994), which in turn searches for the most violated inequality of a potential infeasible solution. The reader is referred to Chen et al. (2009) for more details about the classical algorithms for solving MIP problems.

The developed branch-and-cut algorithm was coded in C++ and CPLEX 12.6 was used as MIP solver. The resulting optimized sequence of licks is fed into the post-processing module.

5.3 Post-processing

In the post-processing module, the solo generated during Step 4 is further modified to avoid unintended dissonances or tension. In 12-bar blues, the harmony is often based on dominant seventh chords, which are those containing a *tritone* (a harmonic interval consisting of three adjacent whole tones) between the major third and the minor seventh (Sadie and Tyrrell, 2001). Such chords generally offer more freedom in an improvisation setting, as they have a characteristic tension due to dissonance that is responsible for an active sound and a strong gravitational movement towards the tonic (Savidge and Vradenburg, 1994). Yet, certain pitches can still cause too much tension depending on their duration and the particular context of

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

the solo. Therefore, the post-processing phase was optionally included in the framework as an attempt to reduce the possible occurrences of such notes.

The module implements two simple procedures based on advice from music experts: (i) if there exists a note in the lick corresponding to the major seventh of the chord that is being played, and if the duration of the note is greater than or equal to 1 beat, the procedure raises the pitch by a semitone so that it becomes the tonic of the chord; (ii) if there exists a note corresponding to the minor second of the chord, the procedure lowers its pitch by a semitone to the tonic of the chord.

An example of a generated solo is displayed in Figure 7. In order to properly evaluate the output of the algorithm, an extensive listening experiment was conducted, which is described in the next section.

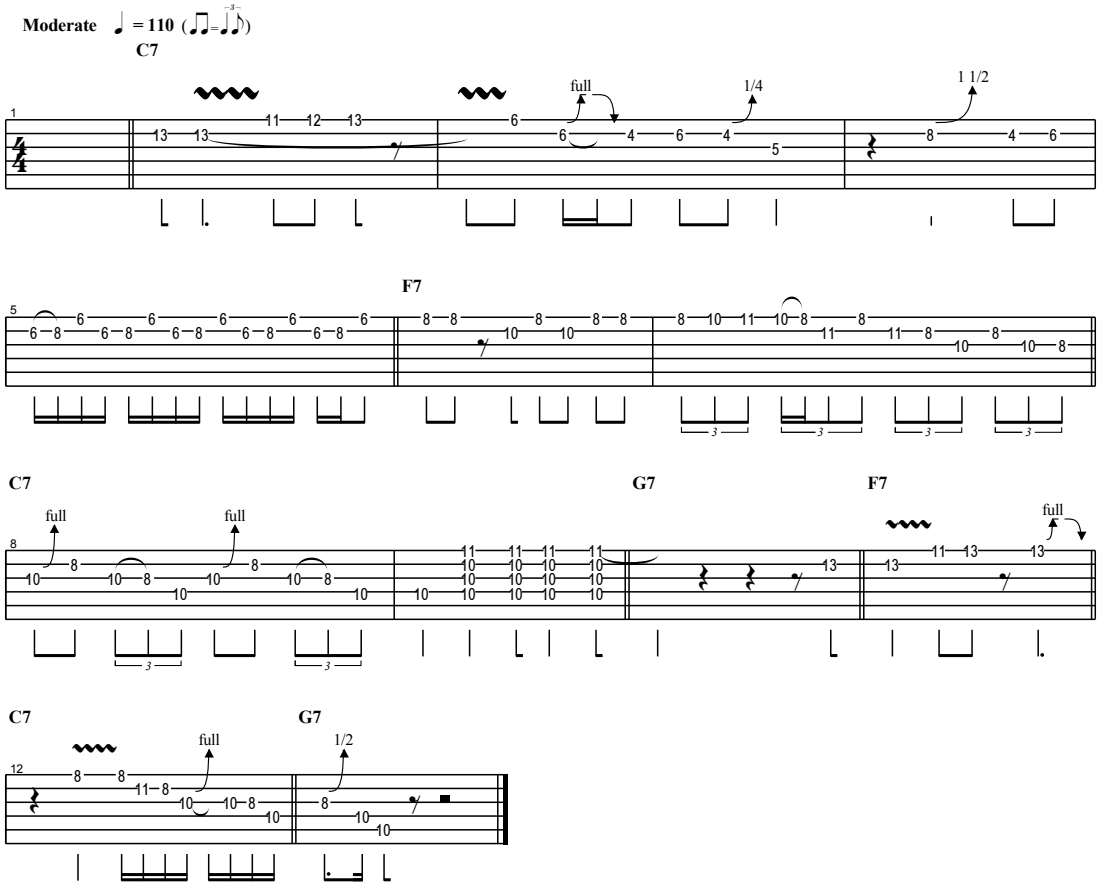


Figure 7: Output example of an optimized solo with a 12-bar blues structure and expressive features (e.g. a bend in bar 2 and vibrato in bar 10). The types of licks occurring in this solo are depicted in Figure 5.

6 Evaluation of the generated solos

As the aim of this research is to create a framework that generates guitar solos, we conducted an extensive listener experiment, as suggested by Agres et al. (2016), to evaluate the musical output. In the following subsections, the experimental setup and results are discussed.

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

6.1 Experimental setup

Different types of listening experiments can be found in the literature on CAC systems. For example, Hall and Smith (1996), who developed a method for automatically composing blues melodies, submitted their results to listener tests. Their results indicate that listeners could not differentiate a composition generated by their system from those created by humans. Bäckman (2009), who suggested an evolutionary-based approach for generating jazz harmonies, submitted their results to a live listening test, in which a jazz band performed the generated harmonies. The objective was to observe its performance while playing harmonies that were created using non-standard chords progressions when compared to standard jazz progressions.

Research in the field of computational creativity, such as (Pearce and Wiggins, 2007) and (Wiggins et al., 2009), is concerned with evaluating the *creative* nature of their systems, hence experiments typically include a comparison with existing music. An experiment that compares existing music with generated music faces some important challenges. The most important challenge being that listeners might recognize the existing music and therefore have a memory/liking effect to it, as per the 'Mere-exposure effect (Zajonc, 1968). Other techniques of evaluating creativity of music generation systems, such as measuring affective responses, are reviewed by Agres et al. (2016). It remains hard to objectively evaluate the output of music generation systems (Pearce and Wiggins, 2001), and in many cases there is no listening experiment at all (Conklin and Witten, 1995; Matić, 2010; Davismoon and Eccles, 2010; Herremans and Sörensen, 2013).

The objective of this paper is to create a mathematical formulation of the guitar solo generation problem and show that it can be solved with an integer programming approach based on licks. We do not claim that our system is creative, but that it can find an optimal solution given a number of user preferences for transition and occurrence of licks. Given the fact that we use licks as "building-blocks", it is essential to show that a uniformly randomised selection of licks does not sound as good as the optimized licks, which is what we will do in this section. In order to evaluate the quality of the generated solos on a higher level, we also asked participants to rate if they found the licks to be good enough to be used in a real-life performance.

In this work we are interested in verifying whether the guitar solos generated by our algorithm are more enjoyable than a (uniformly) random combination of licks. Secondly, we want to evaluate if listeners find the generated licks suitable for inclusion in a live performance. In order to test these two hypothesis, we designed an online experiment in which participants listened to both generated and random solos, and indicated their level of enjoyment on a 7-point Likert scale ranging from 1 (not at all) to 7 (very much), together with indicating if the solo could be used in a song or live performance.

Stimuli generation The framework described above, with the default preference settings described in Section 3, was used to generate 15 optimized solos. An additional 12 random solos were generated by arbitrarily selecting licks. A single random subset of licks containing 160 licks, an amount that is scalable enough for the branch-and-cut algorithm, was derived from the main dataset (of 342 licks). This subset was fed to the optimization algorithm in order to generate the optimized solos, and it was used to randomly select the licks that form the random

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

solos.

The 15 optimized solos were generated as follows. Firstly, the subset of licks, together with the input information, was given to the optimization algorithm. The developed branch-and-cut algorithm is then run to find an optimized solo that will form the first stimulus for the experiment. Next, to generate the second stimulus, a randomly chosen lick of the first stimulus was removed from the subset, so as to prevent the same solo from being generated again. The optimization procedure was then repeated to generate the second stimulus. This procedure was repeated until the 15 distinct stimuli were generated. Finally, in order to prevent biased evaluations, all solos were rendered together with the same 12-bar blues harmony (using dominant seventh chords) and ended with the same turnaround lick.

The random solos were generated by selecting licks uniformly random. The only constraint that was kept is that the solo needs to end in the same turnaround lick as the optimized solos. No improvements were necessary during post-processing.

Participants A set of 173 participants were recruited for the experiment. Based on their musical expertise, the participants can be classified into three groups: (1) Beginners — 75 participants without elementary musical background, including those that are starting to learn how to play an instrument; (2) Intermediates — 51 participants with some theoretical or practical musical background, including amateur musicians. (3) Professionals — 47 participants with extensive musical experience, including professional musicians, teachers, producers, etc.

Procedure An online interface was custom made to conduct the listening experiment. Each participant was presented with 3 optimized solos and 3 random solos, each lasting 30 seconds, in an arbitrary order, yet he/she was not informed about their nature. For each solo, there is a standard accompaniment by bass and drums. After listening to a solo, the participants were asked to indicate their level of enjoyment on a 7-point Likert scale ranging from 1 (not at all) to 7 (very much), and if they thought that the solo could be used in a song or live performance. He/she was also asked to indicate if they thought that the solo could be used in a song or live performance. All stimuli used in this listening experiment are available online at `http://dorienherremans.com/guitarsolos`. In the next section the results of the experiment are discussed in detail.

6.2 Results and discussion

This section presents an extensive analysis of the results of the listening study. The enjoyment ratings from the above described experiment are analysed, followed by a discussion of how the participants rated the usability of a solo in a song or live performance.

6.2.1 Enjoyment ratings

Visualization of participant ratings Figure 8 shows a comparison between the ratings obtained by the two types of solos per proficiency level of the participants. Detailed results are reported in Table 4. We can see that the percentage of individuals that rated the random solos in the range from "Not at all" to "Moderate" tends to increase with the expert level of

the participants. Moreover, only 13.5% of the professional musicians rated the optimized solos between "Not at all" and "Moderate". In addition, while none of them completely dislike the optimized solos, 5.67% of them did not enjoy the random solos at all. When considering all solos that did not even reach a "Moderate" level of enjoyment for such musicians, this number increases to 34%. On the other hand, the percentage of beginners, intermediate and professional musicians that rated the optimized solos in the range from 5 to 7 ("Very much") in the Likert scale was 65.77%, 62.74% and 60.99%, respectively. As for the random solos, the percentages were 54.66%, 45.1%, 43.27%, respectively, which in turn are still surprisingly substantial. This may be explained by the fact that the random solos are still rendered over a 12-bar blues chord sequence, which may seem familiar to listeners. Secondly, while the connections between licks are not optimized, the short individual licks might still sound good, causing short moments of enjoyment that lack a larger structure.

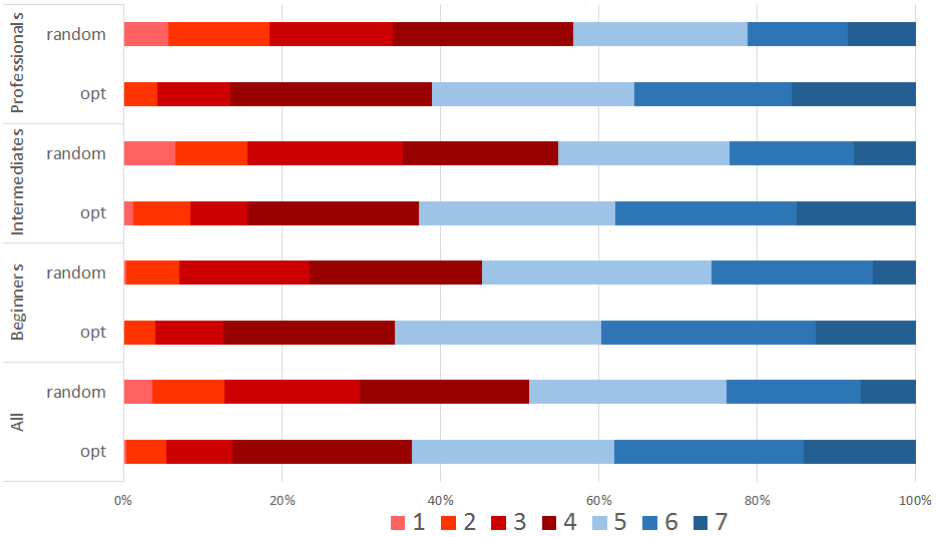


Figure 8: Enjoyment ratings resulting from the experiment

Table 4: Percentage of users that evaluated the solo according to the different categories of the Likert scale

	All		**Beginners**		**Intermediates**		**Professionals**	
	Opt	Random	Opt	Random	Opt	Random	Opt	Random
1 (Not at all)	0.39%	3.66%	0.00%	0.44%	1.31%	6.54%	0.00%	5.67%
2	5.04%	9.06%	4.05%	6.67%	7.19%	9.15%	4.26%	12.77%
3	8.33%	17.15%	8.56%	16.44%	7.19%	19.61%	9.22%	15.60%
4 (Moderate)	22.67%	21.39%	21.62%	21.78%	21.57%	19.61%	25.53%	22.70%
5	25.58%	24.86%	26.13%	28.89%	24.84%	21.57%	25.53%	21.99%
6	23.84%	16.96%	27.03%	20.44%	22.88%	15.69%	19.86%	12.77%
7 (Very much)	14.15%	6.94%	12.61%	5.33%	15.03%	7.84%	15.60%	8.51%

Statistical analysis In order to further analyse the obtained results, we have performed a statistical significance test between the mean ratings of the optimized and random solos using

the *R package for statistical computing* (R Core Team, 2013). Prior to that, we have first conducted three distinct normality tests (Shapiro and Wilk, 1965; Anderson and Darling, 1954; Lilliefors, 1967) over the data that is composed of the difference between the average ratings of the optimized and random solos for each of the 173 participants. The results of each of these tests confirm that the data is not normally distributed, with a p-value of 0.03186, 0.00228 and 0.00006 respectively.

Therefore, we apply a non-parametric significance test, more precisely, the *Wilcoxon signed-rank test* (Woolson, 2008), in order to verify whether there is a statistical difference between the mean ratings of the optimized and random solos. The p-values are for all, beginner, intermediate and professional musicians, are all < 0.0001, which confirm that the optimized solos can be considered more enjoyable than the random ones, regardless of the musician category. This can be visually observed in the box-plots depicted in Figure 9.

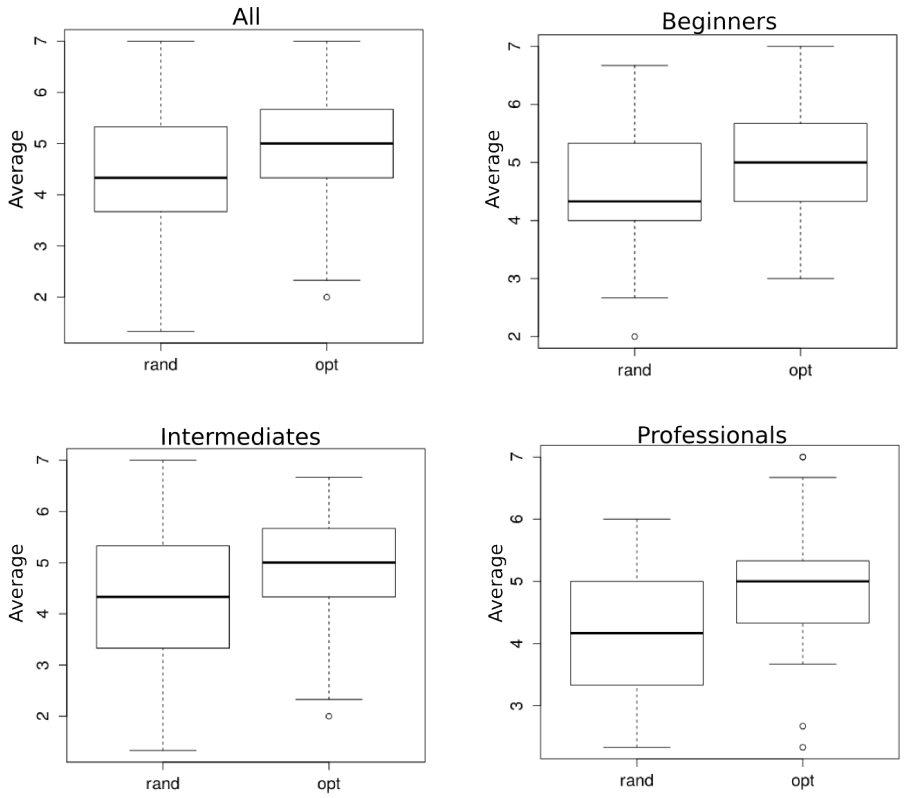


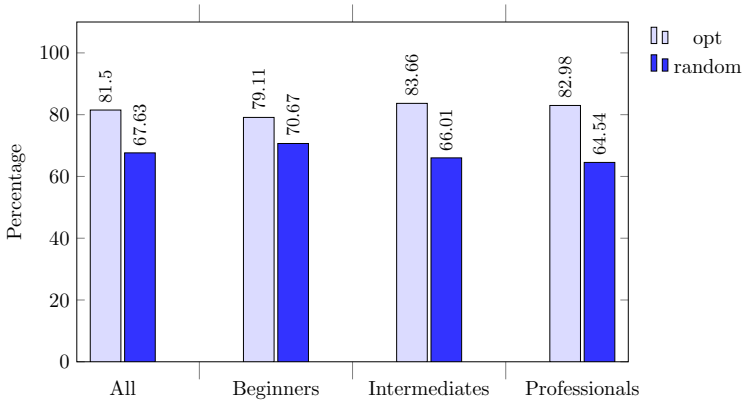
Figure 9: Box-plots of the Wilcoxon signed-rank test showing the enjoyment rankings

6.2.2 Usage in a song or live performance

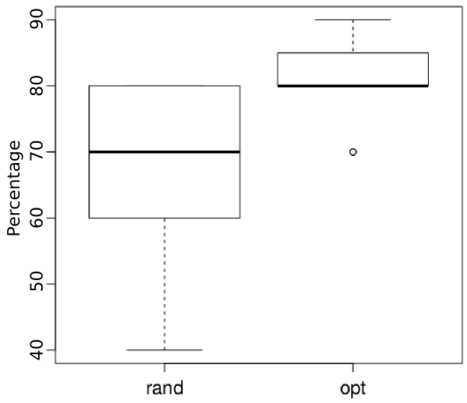
Visualisation of results Figure 10a shows the percentage of times that the optimized and random solos were considered by the participants to be worth using in a song or in a live performance. The results suggest that the optimized solos appear to have more appeal to the participants than the random ones. Note that the percentage difference between the two kinds of solos is more prominent for the professional (18.44%) and intermediate (17.65%) musicians, when compared to the beginners (8.44%), which indicates that professional musicians are better at recognizing optimized solos from random ones.

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

Statistical analysis The same non-parametric test from the previous section was used to verify if there is a statistical difference between the mean values of the optimized and random solos, when it comes to the percentage of times that they were considered to be worthy of being used in a song or in a live performance. The p-value obtained after applying the Wilcoxon signed-rank test is 0.0027, thus illustrating that there is a statistical difference for a 0.01 significance level. The box-plot of the data associated with this analysis is depicted in Figure 10b and clearly shows that a higher percentage of optimized solos were selected for use in a song or live performance than random solos.



(a) Percentage of times that the optimized and random solos were considered by the participants to be worth using in a song or in a live performance



(b) Box-plot of the Wilcoxon signed-rank test showing the percentage of times that the solos of each type were considered to be worth using in a song or live performance

Figure 10: Analysis of results about usage in a song or live performance

The results of this statistical analysis support that the framework developed in this paper is able to successfully generate 12-bar blues guitar solos. The resulting solos are rated as having a higher enjoyability than random solos by users and are preferred over the random solos in the context of a live performance or song.

7 Conclusions

We have developed a computer-aided composition system that generates guitar solos based on an exact optimization algorithm that encodes musical knowledge. The newly produced solos include complex expressive features such as bends and hammer-ons and are generated over a 12-bar blues structure.

The task of generating guitar solos was defined as an optimization problem that consists of finding an optimal sequence of a subset of small melodic fragments (i.e., licks), which are provided by the user. A dataset of licks was built especially for this research, based on a collection of musical textbooks. In cooperation with expert musicians we derived a set of transition preferences which allowed us to specify the objective function. A branch-and-cut algorithm was implemented to find an exact solution to this problem. To the best of our knowledge, this research is one of the first attempts to use this particular type of methodology in the field of computer-aided algorithmic composition. The resulting solos are automatically

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

exported to MusicXML, an open standard for music notation, which allows for an easy display of the solo in tablature format.

In order to evaluate the output of the algorithm, a listening experiment was set up with 173 participants (beginners, intermediates and professional musicians). The results showed that the optimized solos were significantly more enjoyed than those generated as a random sequence of licks, regardless of the category of musicians that evaluated them. This indicates that the proposed framework succeeded in generating new and enjoyable guitar solos that contain expressive features. The approach used in this paper allows for the easy inclusion of constraints, such as turnaround cadence constraints, which is known to be a current challenge in the field of automatic music generation.

In future research, more constraints could be included, and transition preferences might be learned from existing corpora of music. This will enable us to easily expand the framework to other styles of music, so that it becomes even more robust. We also intend to develop an approach for generating the building blocks of this system, i.e., the licks, automatically. This could be realized by potentially using other operational research and statistical methods such as Markov chains or machine learning algorithms, respectively. Due to the positive evaluation of the output of the framework in its current state, we believe that it may already be of use to composers searching for novel (copyright free) ideas; as an educational tool for future composers; and for guitarists in search of original material.

Acknowledgements

We would like to thank Thiago Gouveia and the professional guitar players Zé Filho and Marcos Rosa for their valuable contributions in this research. We also thank the anonymous reviewers for their constructive comments that helped to improve the quality of this paper. This project has received funding from the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brazil, under grant No. 305223/2015-1, and from the European Union's Horizon 2020 research and innovation programme under grant agreement No 658914.

References

Agres, K., Forth, J., and Wiggins, G. (2016). Evaluation of musical creativity and musical metacreation systems. *ACM – Computers in Entertainment*, (in press).

Aldwell, E., Schachter, C., and Cadwallader, A. (2010). *Harmony and voice leading*. Schirmer, 4 edition.

Anderson, T. W. and Darling, D. A. (1954). A test of goodness of fit. *Journal of the American statistical association*, 49(268):765–769.

Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2007). *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA.

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

Bäckman, K. (2009). Automatic jazz harmony evolution. In *Proceedings of the 6th Sound and Music Computing Conference (SMC)*, pages 349–354, Casa da Música, Porto, Portugal.

Biles, J. A. (1994). Genjam: A genetic algorithm for generating jazz solos. In *International Computer Music Conference: International Computer Music Association*, pages 131–141, San Francisco, US.

Biles, J. A. (2003). Genjam in perspective: A tentative taxonomy for GA music and art systems. *Leonardo*, 36(1):43–45.

Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. (2012). Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1159–1166, New York, US. ACM.

Browne, T. M. and Fox, C. (2009). Global expectation-violation as fitness function in evolutionary composition. In *Proceedings of EvoWorkshops 2009: Workshops on Applications of Evolutionary Computation*, pages 538–546, Tübingen, Germany. Springer.

Chen, D., Batson, R. G., and Dang, Y. (2009). *Applied Integer Programming*. John Wiley & Sons, Inc.

Conklin, D. and Witten, I. H. (1995). Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73.

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

Dror, M. (1994). Note on the complexity of the shortest path models for column generation in vrptw. *Operations Research*, 42(5):977–978.

Eck, D. (2016). Welcome to magenta! `https://magenta.tensorflow.org/welcome-to-magenta`. Accessed: 2017-06-15.

Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229.

Fernández, J. D. and Vico, F. (2013). AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48:513–582.

Fisher, J. (1995). *Intermediate Jazz Guitar: The Complete Jazz Guitar Method*. Alfred Music, United States.

Fritts, L. (2012). *2000 Blues: Licks that rock!* Centerstream Publishing.

Geis, M. and Middendorf, M. (2007). An ant colony optimizer for melody creation with baroque harmony. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 461–468, Singapore.

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

Hall, M. A. and Smith, L. (1996). A computer model of blues music and its evaluation. *The Journal of the Acoustical Society of America*, 100(2):1163–1167.

Herremans, D. and Chew, E. (2017). Morpheus: generating structured music with constrained patterns and tension. *IEEE Transactions on Affective Computing*, PP.

Herremans, D., Ching-Hua, C., and Elaine, C. (2017). A functional taxonomy of music generation systems. *ACM Computing Surveys*, 50:30.

Herremans, D. and Chuan, C.-H. (2017). Modeling musical context with word2vec. In *Proceedings of the First Workshop on Deep Learning and Music, joint with IJCNN*, volume 1, Anchorage, US.

Herremans, D. and Sörensen, K. (2012). Composing first species counterpoint with a variable neighbourhood search algorithm. *Journal of Mathematics and the Arts*, 6(4):169–189.

Herremans, D. and Sörensen, K. (2013). Composing fifth species counterpoint music with a variable neighborhood search algorithm. *Expert Systems with Applications*, 40(16):6427 – 6437.

Herremans, D., Weisser, S., Sörensen, K., and Conklin, D. (2015). Generating structured music for bagana using quality metrics based on Markov models. *Expert Systems with Applications*, 42(21):7424–7435.

Hiller, L. and Isaacson, L. (1957). *Illiac Suite*. Score, Theodore Presser Co, New York, USA.

Horner, A. and Goldberg, D. E. (1991). Genetic algorithms and computer-assisted music composition. *Urbana*, 51(61801):437–441.

Huron, D. (2016). *Voice leading: the science behind a musical art)*. The MIT press.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. *Column generation*, 6730:33–65.

Johnson, R. (2002). *How to play guitar Blues: the basics & beyond: Lessons & tips from the great guitar players (Guitar Player musician's library)*. Backbeat Books.

Keller, R. M. and Morrison, D. R. (2007). A grammatical approach to automatic improvisation. In *Proceedings, Fourth Sound and Music Conference*, pages 330–336, Lefkada, Greece.

Krenz, S. (2010). *Learn and master Blues guitar*. Legacy Learning Systems.

Lawler, E. L. and Wood, D. E. (1966). Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719.

Lilliefors, H. W. (1967). On the Kolmogorov-Smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, 62(318):399–402.

Manzi, L. (2010). *Complete acoustic Blues guitar method complete edition*. Alfred Music.

Marshall, W. (1999). *101 Must-Know Blues Licks*. Hal Leonard.

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

Matić, D. (2010). A genetic algorithm for composing music. *Yugoslav Journal of Operations Research*, 20(1):157–177.

McCabe, L. (2002). *101 Blues guitar turnaround licks*. Mel Bay Pubns.

McVicar, M., Fukayama, S., and Goto, M. (2014a). Autoleadguitar: Automatic generation of guitar solo phrases in the tablature space. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pages 201–213.

McVicar, M., Fukayama, S., and Goto, M. (2014b). Autorhythmguitar: Computer-aided composition for rhythm guitar in the tab space. In *40th International Computer Music Conference (ICMC) joint with the 11th Sound and Music Computing conference (SMC)*, Athens, Greece.

Pachet, F. and Roy, P. (2011). Markov constraints: steerable generation of markov sequences. *Constraints*, 16(2):148–172.

Pearce, M. and Wiggins, G. (2001). Towards a framework for the evaluation of machine compositions. In *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 22–32.

Pearce, M. T. and Wiggins, G. A. (2007). Evaluating cognitive models of musical composition. In *Proceedings of the 4th international joint workshop on computational creativity*, pages 73–80. Goldsmiths, University of London.

Pierce, D. (2015). *200 Blues licks for guitar in 3D*. Play music publishing.

Pinkerton, R. C. (1956). Information theory and melody. *Scientific American*, 194(2):77–86.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Roig, C., Tardón, L. J., Barbancho, I., and Barbancho, A. M. (2014). Automatic melody composition based on a probabilistic model of music style and harmonic rules. *Knowledge-Based Systems*, 71:419–434.

Sadie, S. and Tyrrell, J. (2001). *The New Grove Dictionary of Music and Musicians*. Oxford University Press, Oxford.

Savidge, W. M. and Vradenburg, R. L. (1994). *Scales Over Chords Gtr*. Music Sales America 3rd edition.

Schonbrun, M. (2003). *The everything rock & Blues guitar book: from chords to scales and licks to tricks, all you need to play like the greats*. Everything, 2 edition.

Schottstaedt, B. (1984). *Automatic species counterpoint*. Number 19. CCRMA, Dept. of Music, Stanford University.

Scruton, R. (1999). *The aesthetics of music*. Oxford University Press.

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.

Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611.

Stoer, M. and Wagner, F. (1994). *A simple min cut algorithm*, pages 141–147. Springer, Berlin, Heidelberg.

Tanaka, T., Bemman, B., and Meredith, D. (2016a). Constraint programming formulation of the problem of generating milton babbitt's all-partition arrays. In *Proceedings of the 22nd International Conference on Principles and Practice of Constraint Programming*, Toulouse, France.

Tanaka, T., Bemman, B., and Meredith, D. (2016b). Integer programming formulation of the problem of generating milton babbitt's all-partition arrays. In *Proceedings of the 17th International Conference on Music Information Retrieval (ISMIR)*, New York, US. International Society for Music Information Retrieval.

Tanaka, T. and Fujii, K. (2015). Describing global musical structures by integer programming on musical patterns. In *Mathematics and Computation in Music*, pages 52–63. Springer.

Tokui, N. and Iba, H. (2000). Music composition with interactive evolutionary computation. In *Proceedings of the Third International Conference on Generative Art*, volume 17:2, pages 215–226, Milan, Italy.

Wiggins, G., Pearce, M. T., and Müllensiefen, D. (2009). Computational modelling of music cognition and musical creativity. In Dean, R. T., editor, *The Oxford Handbook of Computer Music*, pages 383–420. Oxford University Press, Oxford.

Willmott, B. (1994). *Complete book of harmony theory and voicing.* Mel Bay Publications, Inc.

Woolson, R. F. (2008). Wilcoxon signed-rank test. *Wiley Encyclopedia of Clinical Trials.*

Zajonc, R. B. (1968). Attitudinal effects of mere exposure. *Journal of personality and social psychology*, 9(2p2):1.

Preprint accepted for publication in Journal of the Operational Research Society, Published online on 22 Nov., 2017, DOI: 10.1080/01605682.2017.1390528.