Realtime Generation of Harmonic Progressions Using Constrained Markov Selection

Arne Eigenfeldt[1] and Philippe Pasquier[2]

[1]School for the Contemporary Arts, [2]School of Interactive Arts and Technology,
Simon Fraser University, Canada
{eigenfel, pasquier}@sfu.ca

Abstract. We present a method for generating harmonic progressions using case-based analysis of existing material that employs a Markov model. Using a unique method for specifying desired harmonic complexity, tension between chord transitions, and a desired bass-line, the user specifies a 3 dimensional vector, which the realtime generative algorithm attempts to match during chord sequence generation. The proposed system thus offers a balance between user-requested material and coherence within the database.

1 Introduction

Generative systems have had a long history within computer music [1] and interactive realtime performance [2]. One standard model for such systems has been that of improvisation [3, 4], in which the software interacts with either a composer or performer. Such models have tended to restrict harmonic movement, by employing a static, modal harmony [5] or ignoring harmony altogether in favour of a free-jazz approach [6]. These restrictions are necessitated because harmony cannot, by its very nature, be improvised collectively: it requires a clear goal (although this goal can be achieved through a variety of progressions).

Several computer music systems have been developed that do allow the generation of harmony, although few are in use within realtime computer music, with the notable exception of Rowe [7]. Such systems have tended to be stylistically motivated, in that they attempt to reproduce specific progressions from within a defined stylistic period: for example, Baroque chorales [8].

As Pachet and Roy point out [9], harmonic language is style specific; as such, any system that relies upon specific rules will restrict itself stylistically, and thus limit its potential expressiveness. Furthermore, the same authors note that a harmonic language's rules tend to outline specific combinations and linear progressions that should be avoided, rather than followed.

Markov models offer a straight-forward method of deriving correct harmonic sequences based upon a specific corpus, since they are essentially quoting portions of the corpus itself. Furthermore, since the models are unaware of any rules themselves, they can be quickly adapted to essentially "change styles" by switching source data.

However, as Ames points out [10], while simple Markov models can reproduce the surface features of a corpus, they are poor at handling higher-level musical structures.

This research offers a method for the composer to specify high-level control structures, influencing the generative algorithm that chooses from the generated Markov transition tables. Using a unique method of specifying desired harmonic complexity, tension between chord transitions, and a bass line, the user can specify a three dimensional vector which the realtime generation algorithm attempts to match during sequence generation. As such, the proposed system offers a balance between user-requested material and coherence with the database.

2 Related Work

Harmonic analysis is a well-researched field, particularly in relation to recent advances in Music Information Retrieval (MIR). Harmonic generation, specifically realtime generation for compositional and/or improvisational systems, is less researched, or at least less documented. Composers of interactive music have written only marginally about their harmonic generation algorithms; those systems that are well documented [22, 23] tend to be non-creative systems that attempt to apply correct (stylistic) harmonic practices to a given melody. This may be a good exercise for music students or musicologists attempting to formulate stylistic rules, but one less useful for creative composers.

2.1 Harmonic Analysis

Theoretical models of tonality have existed for decades, if not centuries; one of the most influential in recent years being Lerdahl's Tonal Pitch Space [11]. Anglade and Dixon used Inductive Logic Programming to extract harmonic rules from a large database of existing songs [12]. Ogihara and Li used n-grams of chord sequences to construct profiles of jazz composers [13]. There has been significant research in chord recognition and automatic labeling (for reviews, see [14] and [15]). Similarity of chord sequences has been researched by Liu et. al using string matching [16], while both Pachet [17] and Steedman [18] used rewriting rules. Mauch [19] analysed the frequencies of chord classes within jazz standards. de Haas et. al [20] used a method called Tonal Pitch Step Distance, which is based upon Lerdahl's Tonal Pitch Space, to measure chord sequence similarities

2.2 Harmonic Generation

Methods of harmony generation have included n-gram statistical learning for learning musical grammars [21], as well as several using genetic algorithms [8, 22]. Chuan and Chew [23] created automatic style-specific accompaniment for given melodies, using musical style as the determining factor in type of harmonization. Whorley et al. [24] used Markov models for the generation of 4-part harmonization of "hidden" melodies.

Similarly, Chan and Ventura [25] harmonize a given melody by allowing user input for parameters that governed the overall mood of the composition.

Several systems have used probabilistic models for chord generation, including Paiement et al. [26], whose system was used as an analysis engine for jazz harmony to determine stochastic properties of such harmony. This system is extended in Paiement [27], which uses a machine learning perspective that attempts to predict and generate music within arbitrary contexts given a training corpus, with specific emphasis on long-term dependencies. Allan and Williams [28] used a data set of chorale harmonisations composed by Bach to train a HMM, then used a probabilistic framework to create a harmonization system which learned from examples.

2.3 Differences from Previous Research

Our work differs from previous research in that it is not based in music information retrieval nor cognitive science, but in creative practice. Our particular approach has been informed by a number of heuristic choices stemming from the first author's expertise in composition.

As it is a creative system, our interest is not in modeling a specific musical style; thus, a rule-based system is not useful. Machine learning strategies offer great potential; however, their usefulness has thus far been limited to rather pedestrian activities of melody harmonization. Furthermore, they do not, at this time, offer the flexibility and speed required by realtime computer music. In fact, the realtime nature of our system is one of its distinguishing qualities, in that it can quickly change direction in performance based upon user control. Lastly, we offer a useful measure for harmonic complexity and voice-leading tension that can be used to define harmonic progressions outside of functional harmony. This research does not attempt to construct correct harmonic sequences within the context of functional harmony; it is a creative system based within the 'post-tonal' harmony found in certain 20th century musical styles.

3 Description

This system uses a case-based system [29] to generate Markov conditional probability distributions, using either first, second, or third-order chains. However, rather than allowing the generative algorithm to freely chose from the derived transitions, user specified vectors, suggesting bass-line movement, harmonic complexity, and voice-leading tension, are overlaid in order to stochastically choose from the best matching solutions. The system is written in MaxMSP.

3.1 Source Data

For the purposes of this research, the database consisted of chords derived from jazz standards by Miles Davis (4 tunes), Antonio Carlos Jobim (4 tunes), and Wayne Shorter (6 tunes), all taken from the Real Book [30]. 33 compositions by Pat Metheny

taken from the Pat Metheny Songbook [31], equally drawn from the tunes written in the 1970s, 80s, and 90s, were also used. Source data are standard MIDI files, consisting only of harmonic data at chord change locations (see Section 4).

3.2 Representation

The term set and chord is used interchangeably in this research. In strict terms, every chord is a set, but not every set is a chord. Chords usually refer to vertical collections of pitches that contain a root, 3rd, 5th, and possibly further extensions (i.e. sevenths, ninths) and their alterations (i.e. lowered ninths, raised elevenths, etc.); sets are any combination of unique pitch classes that need not contain specific relationships. Similarly, set-types are unique sets, or chords; for example, the set (0 4 7 11) is a major seventh chord.

Chords are represented as pitch classes [32], although not in normal or prime form. In pitch class theory, the minor triad (0 3 7) is the inversion of the major triad (0 4 7), and is thus considered identical in normal form (i.e. Forte 3-11); however, in tonal music, major and minor chords function very differently. For this reason, the decision was made not to use Forte's set theory representations; instead, the major triad is represented as (0 4 7), whereas the minor triad as (0 3 7).

Extensions beyond the octave are folded within the octave; therefore, the dominant ninth chord is represented as (0 2 4 7 10). Transpositions of chords are not considered unique; instead, bass movement, in pitch classes, between chords is acknowledged. Thus, the chords progression Cmaj7 to Fmaj7 is considered a movement between identical chords, but with a bass movement of +5.

Chords with alternate bass notes (Cm/F) or inversions (Cm/G) are considered unique; thus, Cm/F is represented as (0 2 7 10), and Cm/G is represented as (0 5 8).

Chords are represented within chord vectors as indices into an array of recognized pitch class sets. Currently, this array contains 93 unique chords; for example, the minor seventh chord (0 3 7 10) is the first element in this array and is considered set type 1, while the major seventh (0 4 7 11) is the eleventh element, and is considered set-type 11. When combined with the bass note movement between chords, transitions can be defined as a two-element vector: for example, the pair (2 11) (-2 1) represent a major seventh build on D, followed by a minor seventh chord two semitones lower.

4. Analysis

The database requires an initial analysis, executed prior to performance, to be done on specially prepared MIDI files. These files contain only harmonic data at points of harmonic change, with user-defined markers (controller data) specifying phrase beginning and ending points. Individual files are written for each tune in the database, consisting of the sequential chords (see Section 4.1) and the relative duration of chord types (see Section 4.2). The generation of the Markov transition tables occurs at performance time, as these are dependent upon a user-selected corpus from the larger database (see Section 4.3).

4.1 Harmonic Data

Within the MIDI file, chords are written in root position, with a separate staff containing the bass line (see Fig. 1). This is done for human analysis of the original notation file, since the chord analysis algorithm can identify chord other than root position. No analysis is done on voice-leading, since voice-leading is a performance, rather than a compositional, decision within improvised music; as such, a voice-leading algorithm was created for performance, that controls registral spacing and individual pitch transitions.

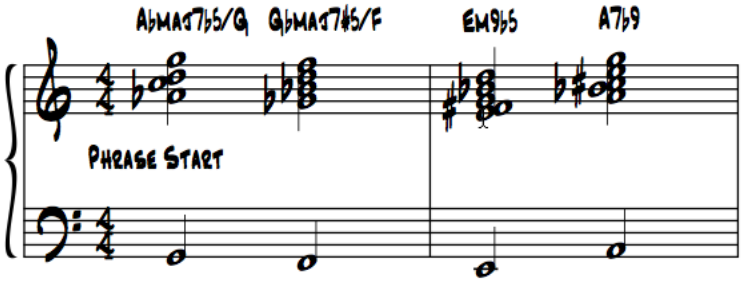

Figure 1. Example notation for analysis

The four chords found in Fig. 1, are represented in Table 1.

Table 1. Different representations of the four chords from Fig. 1. Only the third column is stored in the individual data files.

Chord name	MIDI notes	Stored values	Set Type
AbMaj7 b5 / G	55 56 60 62 67	7 8 12 14 19	31
Gbmaj7#5 / F	53 54 58 62 65	5 6 10 14 17	75
Em9b5	52 54 55 58 62	4 6 7 10 14	76
A7b9	57 58 61 64 67	9 10 13 16 19	25

4.2 Duration Data

A mean duration for each chord is calculated in order to give harmonic duration context for generation. Thus, a separate file is created for each composition in the database that contains the mean harmonic rhythm of the composition, and each individual chord's relative ratio to this mean. For example, if the harmonic rhythm of the composition consisted entirely of half notes, the average duration would be 2.0, or two beats. Each chord type in the composition would then receive a ratio of 1.0.

The use of ratios to an overall harmonic rhythm is used, instead of discrete timings, since it was felt that a chord's relative duration within a composition is more important than its duration in comparison to other compositions. For example, chords that function as (dissonant) passing chords tend to have shorter durations than stable chords anchoring a tonality, and will thus produce smaller ratios.

4.3 Probability Table Generation

The user can select individual compositions from the database as the specific corpus for chord generation. From this corpus, the `initial-chord` array is generated – consisting of the first chord of each phrase – and the `final-chordpair` array: the last two chords in each phrase. First, second, and third-order transition probabilities are then calculated for every chord transition, and compiled separately. The tables store root movement and set type as a pair; thus, using only the four chords from Fig. 1, the first-order table is shown in Table 2.

Table 2. First-order transition table for chords from Fig. 1.

Initial Set	Bass Movement + Set	Occurrences
(0 31)	(-2 75)	1
(0 75)	(-1 76)	1
(0 76)	(5 25)	1
(0 25)	-	1

The third-order transition table for the four chords from Figure 1 contains only one entry (root movements are relative to the first chord), illustrated in Table 3.

Table 3. Third-order transition table for chords from Fig. 1.

Index	Bass Movement + Set	Occurrences
(0 31) (-2 75) (-3 76)	(2 25)	1

After analysing four Miles Davis compositions, the 3 transition tables are illustrated statistically in Table 4.

Table 4. Transition tables for four Miles Davis compositions.

	First-order	Second-order	Third-order
# of chains	14	52	64
# of transitions	170	179	184
# of unique transitions	54	79	89

Since there are only 14 unique set types in these four compositions, there are only 14 first-order chains; however, these chords appear in 64 different 4-chord combinations, thus there are 64 third-order chains. Variety in the generated progressions depends strongly upon the size of the database.

The nature of the user-selected corpus will also influence the generation. Obviously, variety in generation depends on the number of potential transitions. If a corpus is heavily redundant, there will be limited variety in output. On the other hand, selecting a corpus from two composers of very different styles will result in a small intersection of transitions, especially within the higher-order transitions. In such a case, the generated progressions will tend to consist of material from one composer *or* another, with any transitions between the two occurring only when a chain from the intersection of the databases is stochastically selected.

5. Chord Progression Generation

Once the transition tables have been generated for the specific corpus, harmonic progressions can be generated using a mixture of stochastic choice and user request. An initial chord is selected from the `initial-chord` array; given this initial context, the complete harmonic progression is then generated by selecting from the available continuations.

5.1 User Defined Phrase Vectors

Selections are influenced by user-defined vectors for bass line, complexity, and tension, over a user-provided phrase length (see Fig. 2).

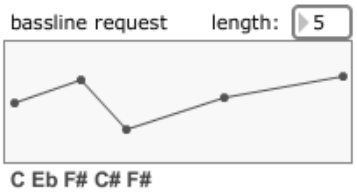

Figure 2. User defined bass line vector.

Given a first chord, the available symbols in the transition table are compared to the user-defined vector. Available symbols are scored based upon the distance between their values and the user-defined vector.

Distance vectors are created for bass-line, complexity and tension (see Section 5.2). The three vectors are each scaled by a user-defined function for each feature (i.e. bass line 0.7, complexity 0.4, tension 0.1): this allows the user to balance the request versus generating coherence with the corpus. The scaled vectors are then summed, and a roulette-wheel selection is made from the highest 5% of these scores. This selection method ensures that, given the same request, a variety of harmonic progressions can result – a desirable attribute in generative systems.

5.2 Harmonic Complexity and Transition Tension

Every set-type has a pre-calculated harmonic complexity, which is a distance function between the pitches of the set and those of a major triad and octave (0 4 7 12). A vector is created of the smallest distance of each note of the set to each note of the triad. Within this vector, each instance of pitchclass 1 (a semitone) is multiplied by 0.3, and each instance of pitchclass 2 (a whole tone) is multiplied by 0.1. Since all possible pitches within the octave are within 2 pitchclasses of one of the notes of the major triad and octave, sets that contain more notes will be considered more dissonant (since they contain more pitchclass differences of 1 and 2 between their pitches and the triad), than smaller sets.

These scores are summed to create the set's harmonic complexity. See Tables 5 and 6 for example ratings of the most consonant and most dissonant set types.

Table 5. Harmonic complexity ratings for the most consonant sets within the database.

Consonant sets	Chord name	Harmonic Complexity
0 7	no 3	0.0
0 4 7	major triad	0.0
0 7 10	7 no 3	0.1
0 2 7	sus2	0.1
0 4 7 9	add6	0.1

Table 6. Harmonic complexity ratings for the most dissonant sets within the database.

Dissonant Sets	Chord name	Harmonic Complexity
0 3 5 6 8 9	13b9 / third	1.3
0 1 3 5 8	maj9 / seventh	1.2
0 3 6 8 11	7#9 / third	1.2
0 1 3 5 7 8	maj9#11 / seventh	1.2
0 1 3 6 10	m7 / sixth	1.0

The tension rating, *tr*, compares the intervals between adjacent sets, dividing *c*, the number of common tones between the two sets by *l*, the length of the second set:

$$tr(s_1, s_2) = 1 - \frac{c(s_1, s_2)}{1(s_2)}.$$

5.3 Generated Harmonic Progressions

Each phrase has a user-specified suggested length in number of chords. During sequence generation, once the generated length reaches 75% of this value, the algorithm begins testing if the last two chords generated are in the `final-chordpair` array. If the test returns true, the phrase generation algorithm exits.

The use of user-defined vectors influences the selection from the Markov transition tables, but there is no guarantee that the actual generated progression will match the user vector, due to the available values within the tables and the roulette-wheel selection from those values. For example, Fig. 3 displays a user-defined bass line, and the resulting third-order generated bass line, using a four-song database containing 108 chains and a requested phrase length of five chords. A larger database will result in a closer approximation, due to the potentially greater available choices. Lastly, the request may not, and need not, be in the style of the corpus; the result will be a stochastically chosen correction of the request given the actual corpus.

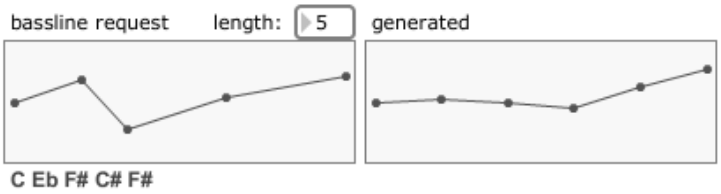

Figure 3. A user defined vector, left, and generated bass line, right, given a 4-song database.

Harmonic rhythm (chord duration) during performance is a ratio to the performance tempo, since every chord in the database acquires a mean ratio to that chord's duration within each composition in which it appears. Thus, relative duration will be consistent, allowing realtime harmonic rhythm to be adjustable, yet independent of the pulse.

6. Conclusions

We have presented a realtime chord sequence generation system that employs a unique user influence over variable-order Markov transition tables. The algorithm described here can be used as a compositional assistant, or embedded within a realtime generative system [33]. In such cases, a large part of the musical success of the system resides in the voice-leading algorithm, which is not described here. This algorithm finds the closest distance between adjacent chord tones, taking into account different chord sizes and octave displacements.

7. Acknowledgements

This research was funded by a grant from the Canada Council for the Arts and the Natural Sciences and Engineering Research Council of Canada.

8. References

[1] J. Chadabe. *Electric Sound: The Past and Promise of Electronic Music*. Upper Saddle River, New Jersey: Prentice Hall, 1997.

[2] J. Chadabe. Interactive composing: an overview. *Computer Music Journal* 8(1), 1984.

[3] T. Winkler. Strategies for Interaction: Computer Music, Performance, and Multimedia. *Connecticut College Symposium on Arts and Technology*, 1995.

[4] G. Garnett. The Aesthetics of Interactive Computer Music. *Computer Music Journal* 25(1), 2001.

[5] J. Chadabe. Solo: A Specific Example of Realtime Performance. *Computer Music - Report on an International Project. Canadian Commission for UNESCO*, 1980.

[6] G. Lewis. Too Many Notes: Computers, Complexity and Culture in Voyager. *Leonardo Music Journal,* 10, 2000.

[7] R. Rowe**.** *Machine Musicianship*. Cambridge, MIT Press, 2001.

[8] R.A. McIntyre. Bach in a Box: The Evolution of Four-Part Baroque Harmony Using the Genetic Algorithm. *IEEE Conference on Evolutionary Computation*, 1994.
[9] F. Pachet, P. Roy. Musical harmonization with constraints: A Survey. *Constraints*, 6(1), 2001.
[10] C. Ames. The Markov Process as a Compositional Model: A Survey and Tutorial. *Leonardo* 22(2), 1989.
[11] F. Lerdahl. *Tonal Pitch Space*. Oxford University Press, 2001.
[12] A. Anglade, S. Dixon. Characterisation of Harmony with Inductive Logic Programming. *International Society of Music Information Retrival (ISMIR)*, 2008.
[13] M. Ogihara, T. Li. N-Gram Chord Profiles for Composer Style Representation. *ISMIR*, 2008.
[14] D. Temperley. *The Cognition of Basic Musical Structures*. Cambridge, MA, MIT Press, 2001.
[15] B. Pardo, W.P. Birmingham. Algorithms for Chordal Analysis. *Computer Music Journal,* 26(2), 2002.
[16] C.C. Liu, J. L. Hsu, and A.L.P. Chen. An Approximate String Matching Algorithm for Content-Based Music Data Retrieval. *International Computer Music Conference*, 1999.
[17] F. Pachet. Computer Analysis of Jazz Chord Sequences. Is Solar a Blues? *Readings in Music and Artificial Intelligence*, 1997.
[18] M.J. Steedman. A Generative Grammar for Jazz Chord Sequences. *Music Perception*, 2(1), 1984.
[19] M. Mauch, S. Dixon, C. Harte, M. Casey, and B. Fields. Discovering Chord Idioms Through Beatles And Real Book Songs. *ISMIR*, 2007.
[20] B. de Haas, R. C. Veltkamp, F. Wiering. Tonal Pitch Step Distance: A Similarity Measure for Chord Progressions. *ISMIR*, 2008.
[21] D. Ponsford, G. Wiggins, C. Mellish. Statistical learning of harmonic movement. *Journal of New Music Research* 28(2), 1998.
[22] S. Phon-Amnuaisuk, G. Wiggins. The four-part harmonization problem: A comparison between genetic algorithms and a rule-based system. *Proceedings of Society for the Study of Artificial Intelligence and Simulation of Behaviour Convention*, Edinburgh, Scotland, 1999.
[23] C. Chuan, E. Chew. A hybrid system for automatic generation of style-specific accompaniment. *International Joint Workshop on Computational Creativity (IJWCC),* 2007.
[24] R. Whorley, G. Wiggins, M. Pearce. Systematic Evaluation and Improvement of Statistical Models of Harmony, *IJWCC*, 2007.
[25] H. Chan, D. Ventura. Automatic Composition of Themed Mood Pieces, *IJWCC,* 2008
[26] J.Paiement, D. Eck, S. Bengio. A probabilistic model for chord progressions. *ISMIR*, 2005.
[27] J.Paiement. *Probabilistic Models for Music*, Phd Thesis, Université de Montréal, 2008.
[28] M. Allan, C. Williams. Harmonising Chorales by Probabilistic Inference. *Advances in neural information processing systems,* 17, 2004.
[29] L. Spector, A. Alpern. Criticism, Culture, and the Automatic Generation of Artworks. *National Conference on Artificial Intelligence*, 1994.
[30] *The Real Book volume 1*, Berklee College of Music, 1971.
[31] P. Metheny, *Pat Metheny Songbook*, Hal Leonard Corporation, New York, 2000.
[32] A. Forte. *The Structure of Atonal Music*, Yale University Press, New Heaven, 1973.
[33] A. Eigenfeldt, P. Pasquier. A Realtime Generative Music System using Autonomous Melody, Harmony, and Rhythm Agents. *12th Generative Art Conference GA2009*, 2009.