A Hybrid System for Automatic Generation of Style-Specific Accompaniment

Ching-Hua Chuan* **and Elaine Chew**†
University of Southern California Viterbi School of Engineering
*Department of Computer Science and
†Epstein Department of Industrial and Systems Engineering
Integrated Media Systems Center, Los Angeles, CA
{chinghuc,echew}@usc.edu

Abstract

Creating distinctive harmonizations in an identifiable style may be one of the most difficult tasks for amateur song writers, a novel and acceptable melody being relatively easier to produce; and this difficulty may result in the abandonment of otherwise worthwhile projects. To model and assist in this creative process, we propose a hybrid system for generating style-specific accompaniment, which is capable of creating new harmonizations for melodies, with proper harmonic resolutions, in a style that is learned from only a few examples. In the proposed system, a chord tone determination module first learns, then determines, which notes in a given melody are likely chord tones. According to these chord tones, triads are assigned first to the bars with unambiguous solutions, and these triads serve as checkpoints. The system then constructs possible chord progressions using neo-Riemannian transforms between checkpoints, and represents the alternate paths in a tree structure. A Markov chain with learned probabilities for these neo-Riemanian transforms then generates the final chord progression. We select four songs by the British rock band, Radiohead, to evaluate the system. Three songs are used for training, and an accompaniment is generated for the held out melody. We present the results of two case studies. We find that the system generates chords closely related to the original, and the resulting chord transitions reinforce the phrase structure of the melody.

Keywords: Automatic Style-Specific Accompaniment, Chord Tone Determination, Neo-Riemannian Transforms, Markov Chains.

1 Motivation

In this paper, we describe an automatic style-specific accompaniment system that makes song writing accessible to both experts and novices. This work is inspired by the fact that many people without formal musical training can sing karaoke, some even quite well, but have difficulty crafting sophisticated chord arrangements in specific styles for the melodies which they sing with such ease. Without proper accompaniment, those creative melodies would have only a limited existence, and would probably soon be discarded. Thus, our solution to this problem is to create a system that would automatically generate accompaniment to a melodic composition in a specific style, given exemplar pieces.

Such a system must satisfy the following requirements. First, the system should be able to identify the features important to the style specified by the user, based on only a few examples. For music novices, it may be difficult for them to use musical terms to describe a style, but it is intuitive for them to ask for harmonization similar to some particular songs. Second, the system must be capable of creating new chords not present in the example pieces, but these chords should still be consistent with the specified style. Third, chord transitions and harmonic resolutions must follow the style of the examples. Last but not least, the accompaniment needs to support the phrase structure of the melody, for example, through the insertion of proper cadences at phrase endings.

In this paper, we propose a hybrid system for generating style-specific accompaniment. The system combines music theoretic knowledge and statistical learning, which has the advantage of being able to simultaneously maintain stylistic elements, such as chord tones and chord transitions, learned from the examples, and create new chords with the theoretically and structurally correct harmonic resolutions. Statistical learning allows the system to construct style-related rules for automatic accompaniment, however, the approach becomes problematic when there are only limited numbers of training examples. The specification of style is often best done with no more than a few examples, as large numbers can dilute the defining features. However, rules learned from only a few examples cannot be widely generalized. In particular, the generating of new chords sequences with appropriate transitions are especially difficult for a purely statistical learning system. Furthermore, without music knowledge, such as the use of cadences at phrase endings and of neo-Riemannian transforms that ensure parsimonious voice leading, a sequential and statistical approach has difficulty generating

appropriate chord progressions with proper structural emphases.

The system comprises of several modules employing different computational approaches. The system first determines the chord tones in the melody. The chord tone determination module applies machine learning techniques to choose the chord tones from the input melody based on the example pieces. The system uses seventeen attributes to represent the melody, including phrase structure information. Then, chords are prescribed at checkpoints in the melody where it has been determined that the harmony in that bar is unambiguous. Using these checkpoints as anchors, we use neo-Riemannian transformations to build chord progressions between between consecutive checkpoints, according to the chord transitions learned, and making sure to provide correct and smooth harmonic resolutions. Finally, we use Markov chains to generate the final chord progression. Section 2 presents the details of each of these system modules.

To demonstrate and evaluate the system, we train the system on three songs of the British rock band, Radiohead, and generate chord progressions for the fourth. The original accompaniment of the fourth piece serves as ground truth. Section 3 presents the results of the experiment and system evaluation.

1.1 Related Work

Automatic accompaniment as a harmonization problem has been studied for more than a decade. Natural Language Processing techniques, such as n-gram statistical learning, have been applied to the learning of musical grammars for harmonizing music in the style of the seventeenth century (Ponsford et al., 1998). Evolutionary techniques, such as Genetic Algorithms, have been proposed and implemented in the generating of four-part chorales (Phon-Amnuaisuk et al., 1999). Phon-Amnuaisuk and Wiggins (1999) compared the results between genetic algorithms and a rule-based system for solving a four-part harmonization problem, and found that the rule-based system performed better than the one employing genetic algorithms. Other techniques, such as Hidden Markov Models, have also been utilized in the harmonization of chorales (Allan and Williams, 2005).

Chord progressions, transitions, and resolutions in the harmonization of pop-rock music have also been studied by musicologists and music theorists. Instead of using traditional roman numerals, Kochavi (2002) and Capuzzo (2004) used the neo-Riemannian framework for analyzing the chord transitions in pop-rock music as transformations on the *tonnetz*, thus demonstrating a viable representation and robust grammar tool that accounts for tonal ambiguities caused by modal mixture in pop-rock music.

Most music computational methods adopt sequential processing in time, i.e., processing the note/bar that occurs earlier before processing the next note/bar. When sequential processing is not required, we have the advantage of selecting the process order in a way that best benefits the task at hand. In the proposed system, instead of computing sequentially, we assign chords first at the checkpoints, then determine the chord progressions in the bars between. Similar computational ideas can be found in Chew and Wu (2005), where the separating of pitches into different voices is done through the connecting of maximal voice contigs, where the solution is certain.

2 System Description

This section provides the details of the proposed hybrid system for automatic accompaniment. The system and data flow diagram is shown in Figure 1. The system consists of three major parts performing the following tasks: chord tone determination, triad construction and checkpoints setup, and chord progression generation. Chord tone determination, described in Section 2.1, chooses the chord tones from the melody. Based on the chord tones reported by the previous module, triads are first used to harmonizing the parts of the melody that contain notes identified strongly as triadic chord tones. The details for triad assignment are shown in Section 2.2.

The third part of the system, described in Section 2.3, is responsible for generating the chord progression for the entire melody. Based on the triads assigned in the previous module. All possible progressions between any two of these triadic checkpoints are generated by applying neo-Riemannian transforms. We use a tree structure to represent the possible paths, and a pruning algorithm to remove paths containing disallowed transitions. The final chord progression is created by considering the conditional probabilities of all possible paths as represented in a Markov chain.

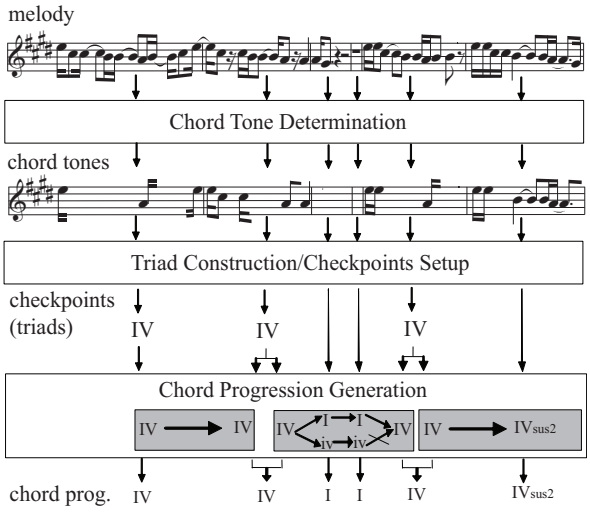

Figure 1: The data flow and system modules for automatic accompaniment

2.1 Chord Tone Determination

The chord tone determination module classifies notes in each bar into chord tones and non-chord tones. We separate this module from the next one, chord determination, for a few reasons: chord tone classification becomes simpler when one does not consider the relations between adjacent chords; this module also learns the melodic harmonization style of each bar; and, new chords (chords not in

the training examples) can still be constructed in a consistent style. For example, a sus4 chord can still be produced by adding the 4th note to the basic triad if the 4th note is reported as a chord tone, even if the sus4 chord does not appear in the training data.

We apply machine learning techniques to determine the chord tones. This module consists of a Support Vector Machine (SVM), which is responsible for selecting chord tones from the melody in each bar in order to learn the types of melodic harmonization decisions made typically by the particular band based on the examples provided. In the learning stage, each note in a bar is represented by the seventeen attributes described in Section 2.1.1; the ground truth is provided by the rhythm guitar chords in the original lead sheet. In the testing stage, the module classifies a note as a chord tone or non-chord tone based on the seventeen attributes of the melodic representation.

The output of the chord tone determination module is a list of notes that should be considered chord tones in each bar of the test piece. Given the output list, a few possible chords can be chosen as harmonization candidates.

2.1.1 Melody Representation

We use seventeen attributes to describe each melody note in a bar. The meanings of these attributes are shown in Table 1. We represent each pitch as a numeric pitch class, numbered from zero to eleven, and normalized so that the tonic is zero. The duration represents the length of the pitch class in beats. The next four attributes describe the scale relationships among the pitch classes in the bar, and include the presence of the upper and lower neighbors, the third, and the fifth. We only consider the intervals of thirds and fifths because neo-Reimannian operations are based on triads, even though they have been extended to include seventh and ninth chords transitions (see Kochavi (2002), Capuzzo (2004).)

Table 1: Attributes for melodic representation

Attribute	**Meaning**
Pitch class (pc)	numeric pc, tonic normalized to zero
Duration	note duration (in beats)
Upper neighbor	pc one scale step above present?
Lower neighbor	pc one scale step below present?
Third	pc three scale steps above present?
Fifth	pc perfect fifth above present?
Metric strength 1	note on metric strength level 1?
Metric strength 2	note on metric strength level 2?
Metric strength 3	note on metric strength level 3?
Metric strength 4	note on metric strength level 4?
Metric beat 1	note on beat 1?
Metric beat 2	note on beat 2?
Metric beat 3	note on beat 3?
Metric beat 4	note on beat 4?
Phrase Position	start, middle, end or bridge
Number of pc's	total number of pc's in bar
Odd/Even bar	note in odd or even bar in phrase?

The next eight attributes relate to metric information. The metric strength attribute shows the metric strength (frequency of onsets on that position in the bar, reminiscent of the inner metric calculations described in Volk (2002) and Chew et al. (2005)) of the pulse on which the note resides. The metric beat attribute records the metric position in the bar according to the time signature. The phrase position in the training data is manually annotated as: start, middle, end, or bridge (an interlude segment between two phrases.) The last two attributes provide information on the number of pitch classes in the bar, and the whether the bar count is odd or even within the phrase.

2.2 Triad construction and checkpoints setup

With the list of chord tones in each bar, we assign triads to harmonize each bar. Triads are the basic building blocks for more elaborate chords. In triad assignment, we appoint chord tones with attributes that strongly support their being members of a triad, such as chord tones having their Third present in the same bar, or chord tones having their Third and Fifth in the same bar. If a chord tone can be harmonized by a major as well as a minor triad, chord selection is determined based on the conditional probabilities calculated during the learning stage.

By using the selected chord tone checkpoints, we first determine the chords for the bars with strong evidences for harmony choice independently of the bars with less evidence for harmonicity. A cadence is a typical example of such a checkpoint. These checkpoints act as the stable points for starting the harmonization process. A wrong chord at a checkpoint dramatically reduces the quality of the song's accompaniment. For instance, incorrect harmonies at a cadential checkpoint can easily result in auditory discomfort.

The setting up of checkpoints divides the harmonization task into smaller sections of chord series generation. Instead of finding a chord progression path for the entire melody, we generate a suitable path of chord progression between each pair of the adjacent checkpoints. This setup not only makes the computation efficient, but also enables us to break the sequential order for processing music. For example, cadential checkpoints help to ensure proper chord resolutions at phrase endings.

2.3 Chord progression generation

This section describes the assignment of chords at the chord tone checkpoints, and between checkpoints.

2.3.1 Chord candidate selection

When the choice for triad assignment is clear, we set the accompaniment chord for that bar to this triad. If the chord tones cannot be harmonized by any one triad, an extended seventh chord will be considered. If the chord tones cannot be fully covered by any triads nor seventh chords, then a compound chord, consisting of pitches from a cluster of chords, will be constructed in order to cover as many of the chord tones in the bar as possible.

A compound chord is constructed based on distance (number of neo-Riemannian transforms) in the chord space shown in Figure 2. For example, the chord pair (I, vi) is a cluster of two chords with a neo-Riemannian

distance of one. The cluster having the shortest total inter-chord distance is chosen as the compound chord, and the function of this chord is determined later based on the context; the determination of context will be described in Section 2.3.2. If there are multiple clusters with the same shortest distance that cover all the chord tones in a bar, then all possibilities are kept as potential candidates.

The new chord construction mentioned above extends the system's vocabulary of chords so that it is capable of harmonizing the melody using new chords not in the learning examples. The system can also generate chords that are neither triads nor seventh chords, but are chords that can be frequently found in popular music, if the chord tones contain sufficient information. For instance, the Fsus4 chord, consisting of the pitches {F, B♭, C}, can be covered by the cluster (F, B♭), i.e., (IV, ♭VII) in the key of C major, and is functionally regarded as a dominant in chord transforms. The extra note, B♭, will not sound incongruous, as long as it is resolved correctly, according to neo-Riemannian transformations.

2.3.2 Neo-Riemannian transforms

Neo-Riemannian transforms have been used by music theorists to analyze harmonic patterns and voice-leading in pop-rock music in the recent decades (Kochavi, 2002; Capuzzo, 2004). There are four fundamental operations in neo-Riemannian transforms for describing the chord progressions: I (Identity, same chord), L (Leading-tone exchange), P (Parallel), and R (Relative), as shown in Figure 2. Although the biggest benefit provided by neo-Riemannian transformations is modal music analysis, we still represent chords in terms of roman numerals in this paper for the following reasons: melodies sung by people are, for the most part, still tonal music; and, the roman numerals normalize all pieces by key, thus reducing the number of pieces required in the learning stage.

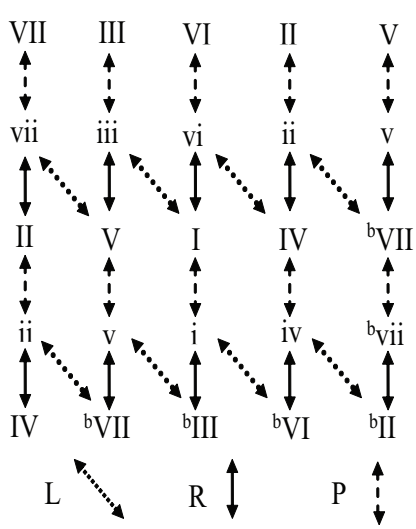

Figure 2: Neo-Riemannian transforms in the chord space.

2.3.3 Tree construction and pruning

We use a tree structure to represent the possible chord progressions between two checkpoints. In the tree, each node represents a chord that contains all the chord tones generated for that bar. A child node represents a chord that results from a learned neo-Riemannian transform to the next bar. The height of the tree equals the number of bars between and including the two checkpoints.

To construct a valid tree for chord progressions between two checkpoints, three constraints must be satisfied. The first two are local constraints, while the third one is a global constraint: (1) the chord selected in each bar should contain all the reported chord tones; (2) a mapping between two adjacent chords must be a valid (learned) neo-Riemannian transform; and, (3) the root node chord must be the first checkpoint, and a leaf node chord at the bottom of the tree must be the second checkpoint.

If a branch cannot continue grow to the second checkpoint, then the branch would not produce a plausible progression. We apply a pruning algorithm for removing those stunted branches in order to make the chord progression calculations more efficient. The pruning algorithm works as follows: if a node cannot establish a valid link to any of the nodes in the next level, it is considered a dead end, and it will report this information to its parent. If a node, n, receives a "dead end" message from all of its children, then n becomes a dead end too. The pruning process continues backtrack until it reaches either a node containing a live child or the root.

An example of a tree structure for chord progressions is shown in Figure 2.3.3. The roman numeral at a node shows the chord candidate for the bar at that level, based on the reported chord tones for that bar. The circled nodes are the checkpoints, which are harmonized by the I chord in this example. Links between nodes (shown with arrows) are created based on valid neo-Riemannian transforms; the particular neo-Riemannian operations invoked are shown in italics. The dashed arrows represent pruning actions when a node cannot continue to grow to the next level. In this example, a total of three valid chord progressions are found: {I, I, V, V, I}, {I, I, iii, iii, I}, and {I, I, iii, vi, I}.

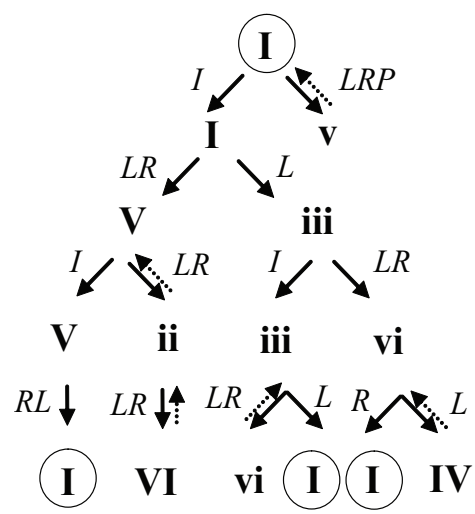

Figure 3: Example tree structure for chord progression.

It may be possible that no branches can be found that connect the two checkpoints, due perhaps to errors in

chord tone determination or to a sparse learning set. In this situation, the generation of chord progression will be split into two sub-problems. At each split, we allow an arbitrary *non*-Reimannian transition. A possible heuristic for selecting a split point considers the number of types of neo-Riemannian transforms learned in bar transitions. A bar transition having more possible chord transforms is more likely to allow more flexible (possibly new) harmonizations (transitions), and is thus a good candidate for a split point. The worst case splitting occurs when the chords selected in each bar cover the chord tones, but no neo-Reimannian transform exist to transition between the bars.

2.3.4 Markov Chains

After constructing the tree, we can readily extract all successful paths between the checkpoints. Each of these paths can be considered a Markov chain, and the likelihood of that path can be calculated from the conditional probabilities in the Markov chain.

Assume we have a path with n chords, $\{C_1, \ldots, C_n\}$, where each chord is indexed by its bar number. The probability that this chord progression occurs can be expressed as:

$$\begin{aligned} & P(C_1, \ldots, C_n) \\ & \quad = \quad P(C_1)P(C_2|C_1)\ldots(C_n|C_{n-1}) \\ & \quad = \quad P(C_1)P(NRO_{1,2})\ldots(NRO_{n-1,n}), \end{aligned} \tag{1}$$

where $NRO_{i-1,i}$ is the neo-Riemannian operation between chord C_{i-1} and C_i. Equation 1 accounts for the probability of the chord progression, but it ignores the phrase information for each bar. In order to generate a chord progression that better reflects the phrase structure of the melody, we modify Equation 1 to include the phrase position information of each bar:

$$\begin{aligned} & P(C_1, \ldots, C_n|B_1, \ldots, B_n) \\ & \quad = \quad P(C_1|B_1)P(C_2|C_1, B_1, B_2) \\ & \qquad \ldots P(C_n|C_{n-1}, B_{n-1}, B_n) \\ & \quad = \quad P(C_1|B_1)P(NRO_{1,2}|B_1, B_2) \\ & \qquad \ldots P(NRO_{n-1,n}|B_{n-1}, B_n) \end{aligned} \tag{2}$$

where B_i is the phrase position for bar i, which falls into one of four possible categories: start (S), middle (M), end (E), and bridge (B), as described in Table 1. For example, $P(LR|S, M)$, the probability that neo-Riemannian operations LR occurs from a starting bar to a bar in the middle of the phrase, can be calculated from the examples as follows:

$$P(LR|S, M) = P(LR, S \rightarrow M)/P(S \rightarrow M) \tag{3}$$

The first term, $P(C_1|B_1)$, in Equation 2 is redundant if C_1 is a checkpoint at the first bar. When C_1 does not occur at the first bar, we may have multiple choices for the chord there, and the term $P(C_1|B_1)$ does affect the resulting progression. If the conditional probability of all chord candidates are zero, due to limited learning pieces, we substitute the term $P(C_1|B_1)$ with $P(C_1)$ instead.

3 Evaluations and Results

In this paper we test our system on music by the British rock band, Radiohead. The experiment design and results are detailed in the following sections.

3.1 Experiment design

We consider four songs by Radiohead: *Creep*, *High and Dry*, *Fake Plastic Trees*, and *Airbag* for our accompaniment generating experiment. We consider these four songs similar, not in terms of particular musical features such as melody, chord progressions, and rhythmic patterns, but according to general properties, for example, all four songs were published in their first three albums, they are each relatively slower than other songs in the respective albums, and each song shows a clear acoustic rhythm guitar accompaniment. However, we did consider one musical fact: all four songs are in major keys; we considered this fact important because strategies of chord progressions would be very different in major vs. minor keys.

We obtained the lead sheets for the songs from the website *http://www.gprotab.net*. For each score, we extracted the two main tracks, melody and rhythm guitar, and removed all other instruments. We manually verified the chords, selecting the main chord for each bar, and discarding ornamental chords and other musical elaborations for the purpose of training and evaluation. Phrase information for each bar and key information are also added to the annotation. The repeats in each song are truncated to simplify the process.

The number of appearances of chords along with their phrase position $\in$ {S, M, E, B} of the four songs are shown in Figure 4a through 4d, where chords are represented as roman numerals, i.e. their function within the key. The phrase position M is furthered specified as being an odd or even bar, {Mo, Me}. Notice that the choices of chords and their distributions are very different from one song to another.

To test our system, we held out one song as the test melody, and used the remaining three training examples. For training, the melodies (represented as described in Section 2.1.1) as well as the chords are given to the chord tone determination module for the learning of the harmonization strategy. The conditional probabilities of chord appearances and neo-Riemannian transform are also calculated from the training examples at this stage. For testing, only the melody is given to the chord tone determination module. Based on the reported chord tones, a chord progression is generated according to the process described in Section 2.3.

3.2 Case Study 1: Creep

First, we choose *Creep* as the test song, and trained the system using the other three. Using the original accompaniment for *Creep* as ground truth, the overall correct rate is 81.48% in this 54-note sample. The statistics on the chord tone determination, such as true positive rate (TP), false positive rate (FP), precision (Prec.), recall (Rec.), and F-measure (F), are shown in Figure 5. Notice that the

false positive rate is much lower than the true positive rate. Thus, by generating accompaniment according to the reported chord tones, the harmonization process should still result in the original chords.

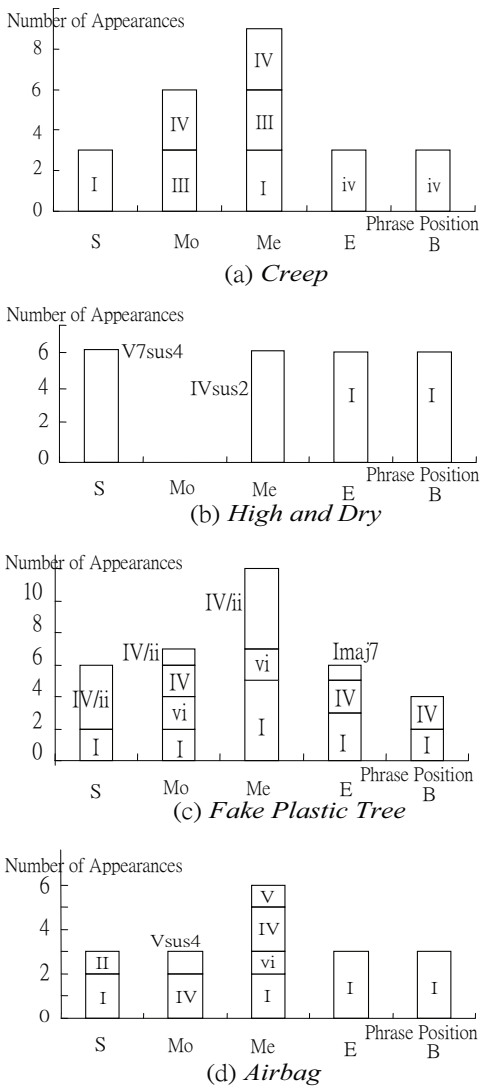

Figure 4: Chord distributions in the original songs

Figure 6 shows the generated chords as well as the original harmonizations for *Creep*. In the 26 bars, 9 of the generated chords are identical to the original. Most of these chords occur at the start of phrases beginning with a I chord, or in the middle of a phrase with IV chord. In the remaining bars, 5 generated chords are the parallel major/minor of the original ones, and 3 generated chords are related by a fifth to the original ones.

With regard to chord tone determination, there are only two false positives, pitch A in bars 1 and 13. In bar 1, the reported chord tones are G, A, which can be harmonized by the cluster of chords IV and the ii of IV in the chord space shown in Figure 2. This generated compound chord, IV+2, still has the function of IV, and is successfully resolved by the LR operation to the I chord in the next bar. In bar 13, the reported chord tones are B, A, harmonized by the iii+2 chord, which happen to be a

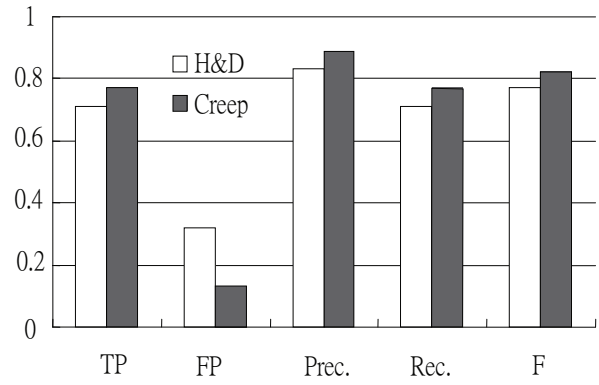

Figure 5: Chord tone determination statistics – true positive (TP), false positive (FP), precision (Prec.), recall (Rec.), and F-measure (F) – for *Creep* and *High and Dry*.

seventh chord in this case.

The original harmonization contains only four chords, repeated periodically in every phrase. The chord B (III) and Cm (iv) are missing in the generated chords. Instead, their parallel major/minor are chosen. We can explain the result by examining the chord transitions in the training songs. In Figures 4b, 4c, and 4d, no chords are related by parallel major/minor in each song. Therefore, no parallel (P) neo-Riemannian operations are learned at the training stage. Although the chord iii and IV are more commonly used than III and iv, and no non-scale pitches such as D♯ and E♭ appear in the melody, the generated chords here may lack some of piquant harmonies of the original. For the bars labeled as cadences (bars 8, 16, 24, and 26), the chords (G, C, G, G) are generated instead of (Cm, Cm, Cm, G) as in the original.

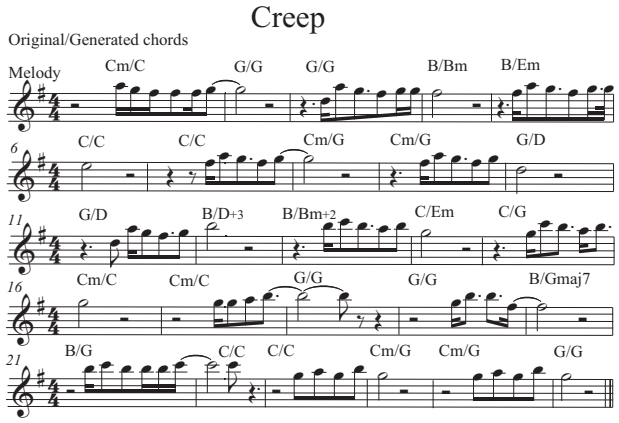

Figure 6: Original/Generated chords for *Creep*.

3.3 Case Study 2: High and Dry

We choose *High and Dry* as our second test piece. The overall correct rate, when comparing to the original lead sheet, is 70.49% on this 61-note sample. The statistics in

Figure 5 show that chord tone determination fared better in *Creep* than in *High and Dry*, especially for the false positive rates.

Three types of false positives are found in the chord tone determination. The first type occurs in bars 2 and 10, where the pitch C♯ is wrongly reported as a chord tone. This results in the choice of the chord A instead of Asus2. The second type happens in bar 22, where the pitch G♯ is reported falsely as a chord tone. However, the resulting compound chord E+4 includes all the pitches in Asus2. The third type occurs in bars 16 and 18. The wrongly reported chord tone F♯ results in a humdrum but straightforward chord progression from bars 16 through 18.

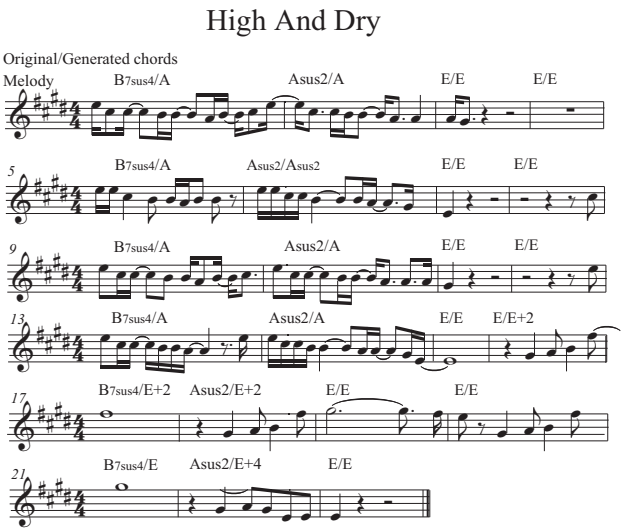

Figure 7: Original/Generated chords for *High and Dry*.

Figure 7 shows the generated chords as well as the original harmonization of the song *High and Dry*. In the 23 bars, 11 of the generated chords are identical to the original. An additional other 6 of generated chords either show the same basic function (bars 2, 10, 14, and 16) or cover the pitches in the original chords (bars 18 and 22). The original harmonization shows regular four-bar chord pattern: B7sus4 → A_{sus2} → E → E. A similar structure can be observed in the generated chord progressions: A → A → E → E. The chord E (I) is generated for all the bars labeled as cadences (bars 3, 7, 11, 15, and 23), as in the original song.

4 Conclusions and Discussion

In this paper, we proposed and demonstrated a hybrid approach to the building of an automatic style-specific accompaniment system. The system aims to make song writing accessible to novices and experts alike. Implementation details of song writing such as chord assignment and arrangement often requires years of musical training and practice, and may prevent less experienced song writers from focusing on the expression of their musical ideas. Our objective is to design a system that can not only provide proper chord progressions to melodies created by novices, but also present new harmonization ideas to experts. To this end, we have proposed a system that models the harmonization process in a sequence of logical steps, and generates chords with proper resolutions. The system is designed not only to allow users to concentrate on higher level decisions and to focus on creative ideas, it also serves as a systematic model for the process of generating accompaniment.

In the two test examples, we demonstrated the system's ability to create new chords, while maintaining the proper chord transition as in the provided examples, in accordance to the phrase structure of the melody. We find that the system generates chords closely related to the original, and the resulting chord transitions meet our expectations based on the melodic phrase structure. We also observed a few challenges in the chord generating process. For example, the melody of *Creep* does not contain strong cues for harmonization, and the system harmonized the same melody in different bars using different chords. Sometimes, one may choose to include chord tones that do not appear in the melody to improve voice-leading, or for ease of fingering on the instrument (e.g. guitar tablature); this are aspects which are currently not captured in the system. Neo-Riemannian operations are based on triads, and are not flexible for generating chords such as IVsus2 with exact pitches. Finally, symmetric phrase structures, reflected for example by regularly repeated chord patterns, are difficult to generate when using only local bar-by-bar analysis. Future systems could incorporate such higher level structures.

Acknowledgements

This research has been funded by the Integrated Media Systems Center, a National Science Foundation (NSF) Engineering Research Center, Cooperative Agreement No. EEC-9529152, a University of Southern California Women in Science and Engineering (WiSE) Digital Dissertation Fellowship, and by the NSF grant No.0347988. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors, and do not necessarily reflect the views of NSF or WiSE.

References

Allan, M. and Williams, C. K. I. (2005). Harmonising chorales by probabilistic inference. In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems: Proceedings of the Neural Information Processing Systems Conference 2004, Vancouver, B.C.*, volume 17, pages 25–32. MIT Press, Cambridge, MA.

Capuzzo, G. (2004). Neo-riemannian theory and the analysis of pop-rock music. *Music Theory Spectrum*, 26(2):177–199.

Chew, E., Volk, A., and Lee, C.-Y. (2005). Dance music classification using inner metric analysis: a computational approach and case study using 101 latin american dances and national anthems. In Golden, B., Raghavan, S., and Wasil, E., editors, *The Next Wave in Computing, Optimization and Decision Technologies: Proceedings of the 9th INFORMS Computer Society Con-*

ference, volume 29 of *Operations Research/Computer Science Interfaces*, pages 355–370. Springer, Annapolis, MD, US.

Chew, E. and Wu, X. (2005). Separating voices in polyphonic music: A contig mapping approach. In Wiil, U. K., editor, *Computer Music Modeling and Retrieval: Second International Symposium, CMMR 2004, Esbjerg, Denmark, May 26-29, 2004, Revised Papers*, volume 3310 of *Lecture Notes in Computer Science*, pages 1–20. Springer-Verlag, Berlin, Germany.

Kochavi, J. (2002). *Contextually Defined Musical Transformations*. PhD thesis, State University of New York at Buffalo, Buffalo, New York.

Phon-Amnuaisuk, S., Tuwson, A., and Wiggins, G. (1999). Evolving music harmonisation. In Dobnikar, A., Steele, N. C., Pearson, D. W., and Albrecht, R. F., editors, *Artificial Neural Nets and Genetic Algorithms: Proceedings of Fourth International Conference in Portoroz, Slovenia*. Springer, Vienna, New York.

Phon-Amnuaisuk, S. and Wiggins, G. (1999). The four-part harmonisation problem: A comparison between genetic algorithms and a rule-based system. In *Proceedings of Society for the Study of Artificial Intelligence and Simulation of Behaviour Convention*, Edinburgh, Scotland.

Ponsford, D., Wiggins, G., and Mellish, C. (1998). Statistical learning of harmonic movement. *Journal of New Music Research*, 28(2):150–177.

Volk, A. (2002). A model of metric coherence. In *Proceedings of the 2nd Conference on Understanding and Creating Music*, Caserta, Italy.