AN INTERACTIVE MUSIC COMPOSITION SYSTEM BASED ON AUTONOMOUS MAINTENANCE OF MUSICAL CONSISTENCY

Tetsuro Kitahara
Nihon University
kitahara@chs.nihon-u.ac.jp

Satoru Fukayama, Shigeki Sagayama
The University of Tokyo
{fukayama,sagayama}@hil.t.u-tokyo.ac.jp

Haruhiro Katayose, Noriko Nagata
Kwansei Gakuin University
{katayose,nagata}@kwansei.ac.jp

ABSTRACT

Various attempts at automatic music composition systems have been made, but they have not addressed the issue of how the user can edit a composed piece. In this paper, we propose a *human-in-the-loop* music composition system, in which the manual editing stage is integrated into the composition process. This system first generates a musical piece based on the lyric input by the user. Then, the user can edit the melody and/or chord progression. The advantage of this system is that once the user edits the melody or chord progression of the generated piece, the system can regenerate the remaining part so that this part musically matches the edited part. With this feature, users can create various melodies and arrangements and avoid the musical inconsistency between the melody and the chord progression. We confirmed that this feature facilitates the trial and error process of users who edit music.

1. INTRODUCTION

Automatic music composition (AMC) is an important task in sound and music computing, from both an academic and an industrial point of view. From an academic point of view, AMC involves constructing a computational model of human creative activities. From an industrial point of view, AMC provides a means for musically unskilled people to obtain original songs. Therefore, various researchers have developed AMC systems [1, 2, 3, 4, 5, 6].

There are two major approaches used in the existing AMC systems. The first is the fully automatic approach, in which AMC systems generate musical pieces based on the user's input, such as lyrics and styles [1, 2, 3, 4]. Because the main focus in those studies is the exploration of new models and/or algorithms for creating musically superior or novel melodies, they do not address the issue of what the system should do when the generated melody does not match that desired or expected by the user.

The second is a semi-automatic approach based on interactive evolutionary computation [5, 6]. The systems based on this approach run iterations of automatic generation of a musical piece and a user's evaluation of the generated piece. The merit of this approach is that it does not require the users to have musical skills because all they need to do is to judge whether the generated piece is good. In practice, however, this approach can impose an excessive burden on users because they have to repeatedly listen to and evaluate system-generated pieces (sometimes thousands of times). In addition, if they want to partly modify the generated piece, they cannot specify which part of the generated piece should be regenerated and how.

The common problem with these studies is that, even if the generated piece is different from what the user wants, the user cannot specify to the system what should differ in the generated piece and how, so that the system can regenerate it[1] . This is an important problem because it is almost impossible for AMC systems to generate pieces that perfectly match users' desires at the first attempt. When the generated piece is different from what the user wants, the most common solution is for the users to edit the piece themselves using commercial software such as music sequencers or digital audio workstations. It is, however, not easy for unskilled people to appropriately edit a generated piece of music using such software because musical pieces in general consist of multiple voices, each of which could produce inharmonic tones if inappropriately edited. We attribute this problem to the unidirectional nature of the composition process: from automatic generation to manual editing.

In this paper, we propose an AMC system, called *OrpheusBB*, in which the manual editing stage is integrated into the iterative composition process. OrpheusBB allows users to edit the melody and chord progression of a generated piece after the first automatic generation. Once the user edits part of the melody or chord progression, the system immediately regenerates the rest of the piece. By repeating such editing, users can elaborate upon the piece without considering the possibility that the melody and the chord progression may become musically inconsistent (typically inharmonic). This approach of iterative composition involving a manual editing stage is called the *human-in-the-loop* approach. The technical issue in achieving this system is how to estimate a melody and chord progression that are musically consistent with the edited part in real time. We call this *autonomous maintenance of musical consistency* (AMMC) and achieve it using a Bayesian network.

2. SYSTEM DESIGN

In the human-in-the-loop approach, music composition is regarded as an iterative process of automatic music gener-

[1] Roads also discussed human-system interactions in music composition from a similar point of view, stating that "totally automated composition programs demand little in the way of creativity" [7].

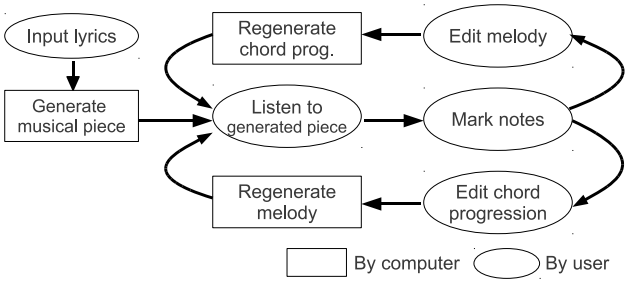


Figure 1. Iterative process in OrpheusBB.

ation and manual music editing (Figure 1). This approach was inspired by the collaboration between a creator and his assistants. A creator (such as a musician) sometimes creates his works in collaboration with assistants. In this case, the creator determines the key concept of the work and creates the important parts alone, but may leave the remaining parts to the assistants. Then, the assistants create the remaining parts using their professional knowledge and experience so that they are consistent with the creator-made parts. The ideal goal of the human-in-the-loop approach is to develop such a sophisticated music composition assistant.

Human-in-the-loop AMC systems should satisfy the following three requirements:

1. *Direct edit*

   If users want to edit a specific part of the melody and/or chord progression, they should be allowed to directly edit the part that they want to edit.

2. *Priority on users' editing*

   Users' editing should have priority. In other words, a mechanism should be provided to avoid overwriting the part edited by the user.

3. *Autonomous maintenance of musical consistency*

   If the user's editing causes musical inconsistency, the system should correct the remaining part to maintain musical consistency. For example, if a user's melody editing introduces a disharmony with the chord progression, the system should correct the chord progression.

OrpheusBB first generates a musical piece based on the lyrics input by the user. After listening to the generated piece, the user can edit any note in the melody and/or any chord in the chord progression (satisfying Requirement 1). When the user edits the melody, the system immediately updates the chord progression to best match the edited melody (satisfying Requirement 3). Similarly, the melody is updated if the chord progression is edited. The user can mark certain manually edited or automatically (re)generated notes if they like those notes. The marked notes will not be overwritten by the update caused by subsequent editing, even if the marked and edited notes cause musical inconsistency (satisfying Requirement 2).

2.1 Initial Input as the Seed of Composition

The initial input of the user,as a seed of the composition, is an important issue because there are an infinite number of musically appropriate melodies. This initial input should efficiently constrain the variety of melodies available but should not require musical skill for the input. If the target music is limited to Japanese songs, lyrics can be used as the initial input, because Japanese is a pitch-accent language [4]. Given a Japanese sentence, its natural prosody in the standard Japanese pronunciation style is uniquely determined, and the pitch motion in the prosody can be used as a constraint for creating a melody. In fact, making a melody following the prosody of the lyrics is well known as a method for Japanese song composition. For this reason, lyrics are adopted as initial inputs in OrpheusBB.

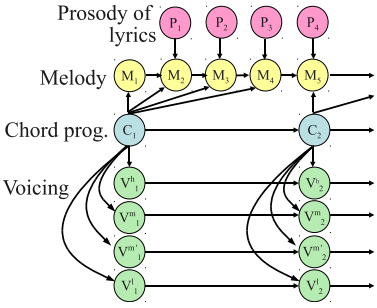
2.2 Achieving Autonomous Maintenance of Musical Consistency

The main technical issue in achieving the above-mentioned three requirements is how to achieve autonomous maintenance of musical consistency (AMMC). The key point in achieving AMMC is determining what type of knowledge representation framework should be used for describing the mutual dependency of the melodies and chord progressions. The knowledge representation framework to be used here should provide a mechanism for inferring the optimal values for the remaining part when part of the framework is updated (a new value is observed). One possible framework for such inference may be the truth maintenance system (TMS) [8]. It is, however, unsuitable for music because TMSs are based on logical reasoning. Another available inference framework is the Bayesian network. A Bayesian network is a probabilistic graphical model in which the dependencies between random events are represented via a directed acyclic graph. Because the dependency between melodies and chord progression can be probabilistically described (e.g., the C♯ note is rarely used in melodies under the C chord), it is suitable for domains, such as music, for which appropriateness cannot be logically determined.

In this study, we describe the dependencies between the prosody of lyrics, a melody, a chord progression, and chord voicings (each is called a *layer*) in a dynamic Bayesian network (DBN), as shown in Figure 2. This network represents two types of dependencies: the sequential dependency within each layer and the simultaneous dependency between different layers. The network should ideally have arcs between the melody nodes and the voicing nodes [9], but these are omitted to reduce the computational cost. To use this network, the system has to know the temporal correspondence between the nodes of the melody layer and those of the chord progression layer. We therefore put the following restrictions on the melody and chord progression:

- The rhythm of the melody is determined in the initial generation stage and cannot be modified in the subsequent manual editing stage.
- The chords change every two beats (half measure).

Once music knowledge has been represented in this



P_i	Prosody (pitch motion) of the i-th mora from the previous mora {up, down, flat}
M_i	i-th note in the melody {C, C♯, D, · · ·, B}
C_i	i-th chord {C, · · ·, B}×{maj, min}
V_i^x	Voicing for the i-th chord {C, · · ·, B}

Figure 2. Dynamic Bayesian network used in our system.

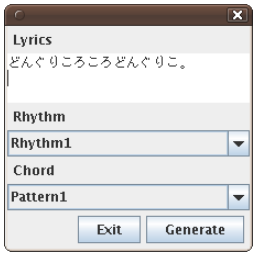


Figure 3. Initial input window.

DBN, AMMC is reduced to the problem of inferring the most likely values for the remaining nodes after updating the value for the node corresponding to the edited note/chord. Marking a note is reduced to giving a probability of 1.0 to the current value for the node corresponding to the marked note. AMMC is therefore achieved through the following steps:

1. If the user marks a note, the probability of the current value for the corresponding node is set to 1.0.
2. If the user edits a note or chord, the value for the corresponding node is updated, and then, the probability inference is determined.
3. After determining the probability inference, the music data (melody, chord progression, and chord voicings) are updated to the values with the highest likelihood, and the editing window is updated for further editing.

3. IMPLEMENTATION

We implemented OrpheusBB based on the design described above.

3.1 Graphical User Interface

3.1.1 Initial Input Window (Figure 3)

Once OrpheusBB is launched, the initial input window appears. The user inputs Japanese lyrics with both *kanji* and *kana*. If necessary, the user can change the chord progression and/or the rhythm pattern of the melody. Once the "Generate" button is pressed, the system generates a melody that has the same pitch motion as the prosody of the input lyrics, and proceeds to the music editing stage.

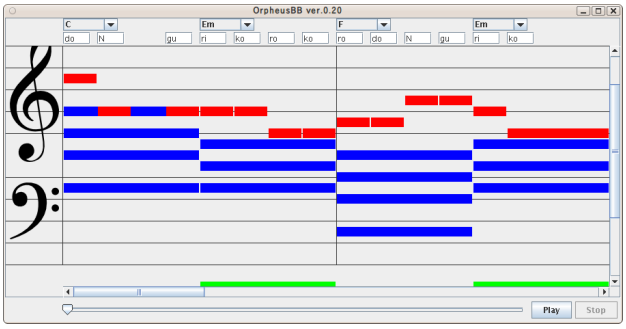


Figure 4. Music editing window.

3.1.2 Music Editing Window (Figure 4)

Once it has generated a musical piece, OrpheusBB displays a pianoroll-like window, where the user can edit the melody, chord progression, and chord voicings in the generated piece. If one or more components are edited by the user, the remaining elements are immediately updated based on the afore-mentioned AMMC function to avoid producing inharmonic tones in the edited melody or chord progression. Specifically, the chord progression and its voicings are updated when the melody is edited, and the melody and chord voicings are updated when chords are changed.

The user can mark notes in the melody and/or chord voicings to prevent those notes from being changed by AMMC in subsequent editing against the user's will (Figure 5 Left). In the typical situation for this note marking function, the number of marked notes gradually increases with every iteration of the manual editing and automatic regeneration process. As the number of marked notes increases, the musical piece is expected to become closer to the user's desires. Thus, the user elaborates upon the piece through the iterative action of (1) editing notes/chords, (2) listening to the edited and automatically updated notes, and (3) marking them if desired.

The chord candidates are represented by chord names, such as *C* or *Dm*; however, the selection of an appropriate chord is not easy for people who are not familiar with this notation. We therefore implemented a function for assisting such people in selecting chords, where the likelihoods of the chord candidates are represented in grayscale (Figure 5 Right). If a chord has a high likelihood under the current context (the melody and neighboring chords), the chord name appears in dark gray (or almost black) in the dropdown list. If a chord has a low likelihood, on the other hand, the chord name appears in light gray (or almost white). Using this function, the user can select a chord that is expected to match the current melody from the dark gray candidates. In addition, the user can select a light gray chord to change the impression because the light gray chords are expected to have largely different impressions. When a light gray chord is selected, the melody is regenerated in most cases.

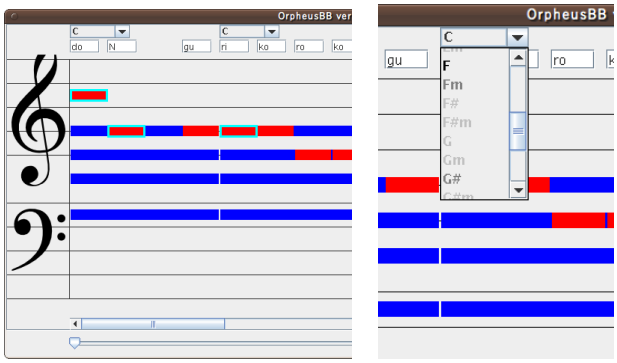


Figure 5. Left: marked notes. Right: a dropdown list for selecting a chord where the chords with high likelihoods are displayed in dark gray and the chords with low likelihoods are displayed in light gray.

3.2 Generation of a Melody from Lyrics

Given a set of lyrics, a melody is generated using the method adopted in Orpheus [4]. The input lyrics are first analyzed by the front-end module of a Japanese text-to-speech engine to identify the pronunciation (*yomi*) and prosody (pitch motion) of the lyrics. Then, the rhythm of the melody is determined. Under the constraint that the melody must have the same number of notes as the number of morae of the lyrics, the system generates the rhythm by dividing (e.g., from one quarter note to two eighth notes) or merging (e.g., from two quarter notes to one half note) notes in the rhythm pattern selected by the user as needed. The notes in the pattern that should be preferentially divided or merged are defined by a tree structure called a *rhythm tree* [4].

After constructing the rhythm, the pitch (note number) of each note in the melody is determined so that the pitch motion matches the prosody of the lyrics and is also musically appropriate. Based on the emission probability of each pitch and the transition probability from each pitch to each pitch, which are manually defined in advance, the note sequence that has the highest probability is searched for using the dynamic programming search method [4].

3.3 Autonomous Maintenance of Musical Consistency

The autonomous maintenance of musical consistency (AMMC) is achieved based on the DBN described in Section 2. Because implementation of the DBN shown in Figure 2 is not practical due to the computational cost and the amount of training data required, we simplify the DBN as follows:

- When the melody is edited, the system infers only the chord nodes, considering all melody nodes to be given. In this case, the DBN is equivalent to a hidden Markov model, so the Viterbi algorithm can be used to infer the chord nodes.
- When a chord is changed, the system infers only the melody nodes, considering all chord and lyric nodes to be given. In this case, inference of the melody nodes is equivalent to an optimal path search performed in Orpheus for melody generation.

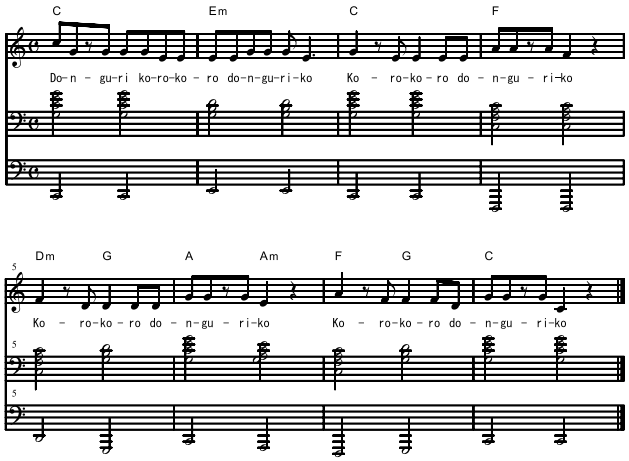


Figure 6. The generated piece given the lyrics "*Donguri korokoro donguriko*." The latter part "*korokoro donguriko*" was repeated three times because the input lyrics are too short for a eight-measure piece.

- If a chord is changed by the user or system, only the voicing of the changed chord is inferred.

The conditional probabilities for all nodes are experimentally defined based on conventional music theory.

4. EXAMPLE OF USE

In this section, we present an example of the composition of a musical piece using this system as a proof of concept. In particular, we will confirm here that:

(1) chords are automatically changed when notes in the melody are changed to notes producing disharmony with the current chord progression,

(2) the melody is automatically regenerated when edited chords produce disharmony with the current melody,

(3) the user can prevent the system from overwriting the notes that they like by marking such notes, and

(4) the coloring of the chord names in the chord selection lists helps the user select chords.

A user executed music generation with the lyrics "*Donguri korokoro donguriko*", obtaining the piece shown in Figure 6. He then changed "E–E–G–G" in the melody in the second measure to "F–F–A–A", which results in disharmony with the current chord "Em". Then, the chord was automatically changed to "F", which matches the edited melody (Figure 7). The neighboring chords were also changed. He marked the edited and automatically updated notes to prevent those notes from being overwritten.

Next, he changed the chord progression of the last two measures ("F–G–C–C") to "F–A♭–G–C" to change the impression of the ending. He selected "A♭" because this chord may give listeners the feeling of oddness as it is rarely used in this context, according to the color (light gray) of the chord name in the dropdown list. Accordingly, the melody was regenerated (Figure 8), and he marked those notes.

The unmarked notes of the first two measures in the melody were, however, automatically changed. This is be-

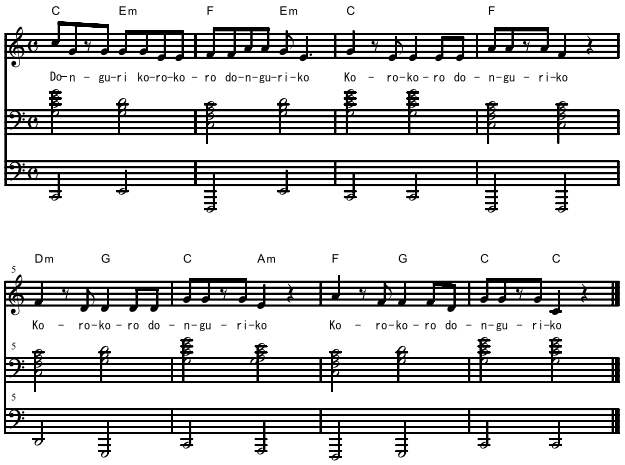


Figure 7. The result of changing "E–E–G–G" in the second measure of Figure 6 to "F–F–A–A". The corresponding chord was automatically changed to "F".

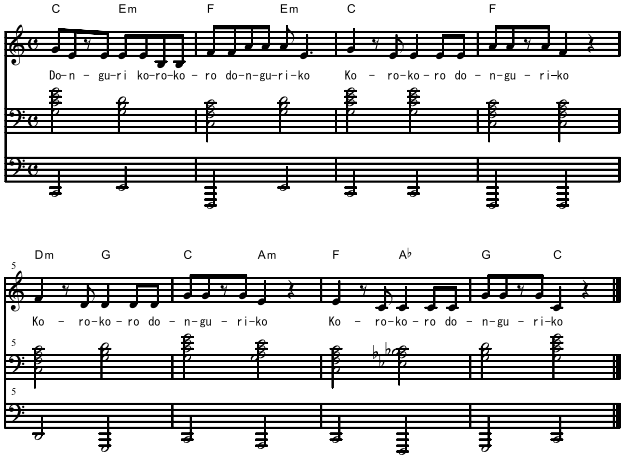


Figure 8. The result of changing the last two-measure chord progression in Figure 7 to "F–A♭–G–C". The melody was automatically regenerated.

cause, in the current implementation, any unmarked notes in the melody may be updated when a chord is changed. Of the updated melody, he changed the notes "B" in the first measure and "A" in the second measure to "C" and "G", respectively. Accordingly, the second chord in the first measure was automatically changed from "Em" to "Am". The chords in the second measure were not changed because they did not cause disharmony with the changed melody.

Finally, he changed the last "G" note in the last measure to "A". The voicing notes for the corresponding chord were not changed because these notes were marked. The completde piece is shown in Figure 9.

Thus we confirmed that the four aims mentioned in the beginning of this section were achieved.

5. DISCUSSION

5.1 Results of Trial Use

Through the trial use reported in the previous section, we confirmed that users can compose musical pieces based

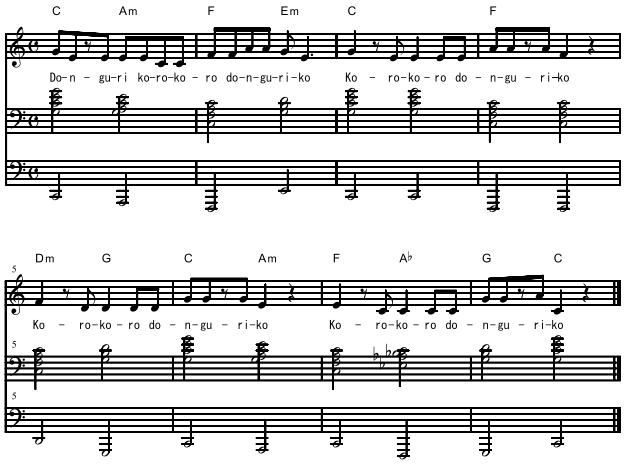


Figure 9. The complete musical piece.

on our composition model, where a musical piece gradually becomes closer to the user's desires by repeating the four steps of (1) listening to the system-generated piece, (2) marking the notes that the user likes, (3) editing the notes or chords that the user does not like, and (4) having the system update the rest according to the user's editing.

In addition, the coloring of the chord names in the chord selection lists was effective at assisting the user in selecting chords. It is well known that the use of non-diatonic chords is effective in making an impressive chord progression, but it is not easy to use these chords with a conventional music sequencer if the user does not have a knowledge of harmony theory. With the chord coloring of our system, users can simply select a chord from the light gray chord names. Because the melody is automatically regenerated, the users do not have to consider the mismatch between the melody and chord progression when selecting a chord.

However, the following areas for improvement were revealed in the trial:

- **Conditional probability table (CPT)**

  To make the system behavior clearly understandable, we adopted an extreme CPT: the probabilities of non-diatonic chords and the conditional probabilities of non-chord tones are very close to zero. For this reason, all melody notes under the A♭ chord were C. The exploration of more appropriate CPTs is an important future issue. The training of CPTs from existing music data should also be investigated.

- **Over-updating**

  Whereas AMMC worked well overall, some notes were automatically changed, even though they did not produce inharmonic tones. Because this phenomenon may negatively affect the user's trial and error process, it should be avoided by, for example, establishing a threshold for overwriting: if the likelihood for the current value is higher than the threshold, the current value should not be overwritten to the most likely value, even if it is not the most likely value.

- **Undo function**

  The user wanted to revert his editing several times but

unable to do so because the current implementation does not have an undo function. Implementing an undo function to facilitate further users' trial and error will be important in our future work.

5.2 Directability

The recent development of machine learning technologies and music corpora has facilitate great advances in the automation of music generation such as music composition, music arrangement, and musical performance rendering. Needless to say, such automation technologies are important in achieving an environment that enables musically unskilled people to create music. Most existing studies, however, have neglected the issue of allowing users to modify the automatically generated content.

Automation technologies should be developed by considering how the technologies are (or should be) used by users. From this point of view, we recently introduced the concept of *directability* [10], which indicates the controllability of content at an appropriate abstraction level. With conventional tools, such as music sequencers, creators have to directly edit all of the components of the content (e.g., all of the notes in the music). The goal of directability is to achieve intuitive editing by editing a structure-level representation.

Although users edit musical content at the note level in OrpheusBB, the editing is immediately reflected in the music structure described in a DBN. OrpheusBB can therefore be considered to be a directable user interface for editing music.

5.3 Use for Further Advanced Arrangement

AMMC would be more effective if it were applied to more advanced arrangements. When a walking bass line is used in the bass part, appropriate passing notes are carefully determined to smoothly connect chord to chord. If a chord is changed, the passing notes around the chord (especially before the chord) in the bass line should also be appropriately changed. By adding the bass line layer to our DBN, we can achieve autonomous maintenance of the walking bass line: once a chord is changed, the passing notes around the chord in the bass line are automatically modified. In actual performance situations, the detailed arrangement, such as passing notes in a bass line, is often left to each player, whereas the overall arrangement, such as the chord progression, is determined by the arranger. Similarly, users can focus on the overall arrangement, being freed from the detailed arrangement, by using AMMC.

6. CONCLUSION

Automatic music composition (AMC) systems, rather than simply generating a musical piece, should provide an environment that enables users to elaborate upon music by repeated manual editing with the assistance of AMC technology. Based on this belief, we developed a human-in-the-loop AMC system, *OrpheusBB*. The main advantage of this system is that, when the melody or chord progression is edited by the user, it can automatically regenerate the remaining part to maintain musical consistency between the edited part and the remaining part. We call this feature *autonomous maintenance of musical consistency*, and achieved it by using probabilistic inference based on a dynamic Bayesian network.

We have some future plan. First, we will conduct quantitative evaluations to more thoroughly explore the effectiveness of this system. Second, we plan to improve the graphical user interface for editing music data. The current user interface is not easily used by people who are unfamiliar with notewise editing in a pianoroll display. We are therefore currently developing a user interface that enables both notewise and non-notewise editing. Third, we plan to extend our DBN to achieve a more advanced arrangement, as discussed in Section 5.3.

Acknowledgments

This research was partially supported by CREST, JST, Japan. OrpheusBB was implemented in collaboration with Mr. Naoyuki Totani and Mr. Ryosuke Tokuami (Kwansei Gakuin University).

7. REFERENCES

[1] L. Hiller and L. Isaacson, "Musical composition with a high-speed digital computer," *Journal of Audio Engineering Society*, 1958.

[2] C. Ames and M. Domino, "Cybernetic composer: An overview," in *Understanding Music with AI*, M. Balaban, K. Ebcioglu, and O. Laske, Eds. AAAI Press, 1992, pp. 186–205.

[3] D. Cope, *Computers and Musical Style*. Oxford University Press, 1991.

[4] S. Fukayama, K. Nakatsuma, S. Sako, T. Nishimoto, and S. Sagayama, "Automatic song composition from the lyrics exploiting prosody of the japanese language," in *Proc. Sound and Music Computing*, 2010.

[5] D. Ando, P. Dahlstedt, M. G. Nordaxhl, and H. Iba, "Computer aided composition by means of interactive gp," in *ICMC 2006*, 2006, pp. 254–257.

[6] J. A. Biles, "Genjam: A genetic algorithm for generating jazz solos," in *Proc. ICMC*, 1994.

[7] C. Roads, *The Computer Music Tutorial*. MIT Press, 1996.

[8] J. Doyle, "A truth maintenance system," *Artificial Intelligence*, vol. 12, no. 3, pp. 251–272, 1979.

[9] T. Kitahara, M. Katsura, H. Katayose, and N. Nagata, "Computational model for automatic chord voicing based on bayesian network," in *Proc. ICMPC 2008*, 2008, pp. 395–398.

[10] M. Hashida and H. Katayose:, "Mixtract: A directable musical expression system," in *Proc. ACII 2009*, 2009.