Interactive Music Composition with Evolutionary Computation

Ying-ping Chen

NCLab Report No. NCL-TR-2007001
January 2007

Natural Computing Laboratory (NCLab)
Department of Computer Science
National Chiao Tung University
329 Engineering Building C
1001 Ta Hsueh Road
HsinChu City 300, TAIWAN
http://nclab.tw/

Interactive Music Composition with Evolutionary Computation

Ying-ping Chen
Department of Computer Science
National Chiao Tung University
HsinChu City 300, Taiwan
ypchen@cs.nctu.edu.tw

January 8, 2007

Abstract

This article presents an interactive music composition system which utilizes the black-box optimization model of evolutionary computation. The core CFE framework—Composition, Feedback, and Evolution—is presented and described. The music composition system produces short, manageable pieces of music by interacting with users. The essential features of the system include the capability of creating customized pieces of music based on the user preference and the facilities specifically designed for generating a large amount of music. In addition to the system structure and implementation, we also introduce the auxiliary functionalities of the system. Finally, several pieces of music composed by the described system are demonstrated as showcases. This work shows that it is feasible and promising for computers to automatically compose customized or personalized music.

1 Introduction

Music plays an important role in our daily life. It makes us sad, happy, and excited. One may wish to listen to "pleasant music", but the definition of pleasant music is quite different for different people. Hence, composing music that is loved by everyone is an extremely difficult task, if not impossible. Furthermore, nowadays we are surrounded by lots of electronic devices capable of playing music or generating sound, such as alarm clocks and cellular phones. These devices oftentimes can be customized to play the user-specified music. For example, we can observe that many people try to use different, distinguishable ringtones for their cellular phones. The purpose for us to do so is not merely to distinguish phone calls but also to establish self-identities by using the music or sound that can define us. As a result, customization for pleasant music is desirable for our modern life.

In addition to music customization, for some applications, a large amount of music pieces may be needed, such as the scene music of games and the background music of web pages. It would be fantastic if ordinary people can easily create music or sound on their own. Although there are lots of computer software which can help people to compose music, such a task of composition is still very hard for unskilled or untrained people. In order to resolve this situation, we are trying to make computers able to automatically create music for us instead of merely letting us put notes into tracks. For this purpose, we develop a system which creates pieces of music by interacting with users. The generated music can be used on cellular phones, alarm clocks, or other devices of which the music can be set or loaded by the user.

In particular, we design an interactive music composition system based on the techniques borrowed from two fields. One is evolutionary computation [1, 2, 3, 4, 5]. Based on the concepts

and models of evolutionary computation, we build the kernel optimization mechanism which can interact with the user and consider the scores given by the user as the objective values. The other is computer music. More specifically, we adopt the MIDI format, which is used in the system as the output format. If we create music in the MIDI format, we can guarantee that the created music can be played on computers, cellular phones, or other customizable devices.

The article is organized as follows. Section 2 briefly reviews the current state of creating music in the field of evolutionary computation. Section 3 describes the CFE framework, and section 4 presents the auxiliary functionalities for enhancing the system. The showcases are demonstrated in section 5, and the URLs for accessing these showcases are offered. Finally, section 6 concludes this article.

2 State of the Art

There have been several attempts to compose music with the techniques of evolutionary computation. In this section, we briefly review these proposed frameworks and broadly classify them by analyzing the three facets: the initialization, the grading method, and the goal to achieve.

Initialization. We can classify the frameworks based on the methods used to initialize the population in the evolutionary environment. There are several kinds of initialization procedures proposed in the literature:

- *Random initialization.* Random initialization [6] provides a relatively bad quality for the initial population but is limited by fewer restrictions than other methods are.
- *Complex function initialization.* Complex function initialization [6, 7] initializes the population through certain pre-designed rules and only produces individuals which satisfy the specified restrictions.
- *Song initialization.* Song initialization [8] creates the population by analyzing one or more available songs and by decomposing these songs into individuals.

Grading method. One of the essential components to create music is the way to judge or grade the music generated by the computer program. In the literature, we can find the following methods for grading music pieces:

- *Real audience.* One way is to judge the music with the real audience through either real-time judging [4, 9] or non-real-time judging [5]. Because lots of runs may needed in the evolutionary process, such a grading method may easily tire the audience.
- *Neural network.* Neural network modules can be trained to evaluate the generated music [5]. However, it takes tremendous time to train the neural network, and the judgment quality offered by a trained neural network is also hard to determine.
- *Artificial fitness functions.* Some frameworks utilize certain fitness functions [6, 7] to automatically grade the generated music. In these approaches, constructing an appropriate fitness function is hard and critical.

Goal to achieve. According to the different goals of the music creation frameworks proposed in the literature, we can have the following categories:

- *Theme of music* To evolve the theme of music, a sequence of notes [4, 8] or a sequence of functions, such as $\sin(\cdot)$ and $\cos(\cdot)$ [7], is adopted as the genotype of the music.

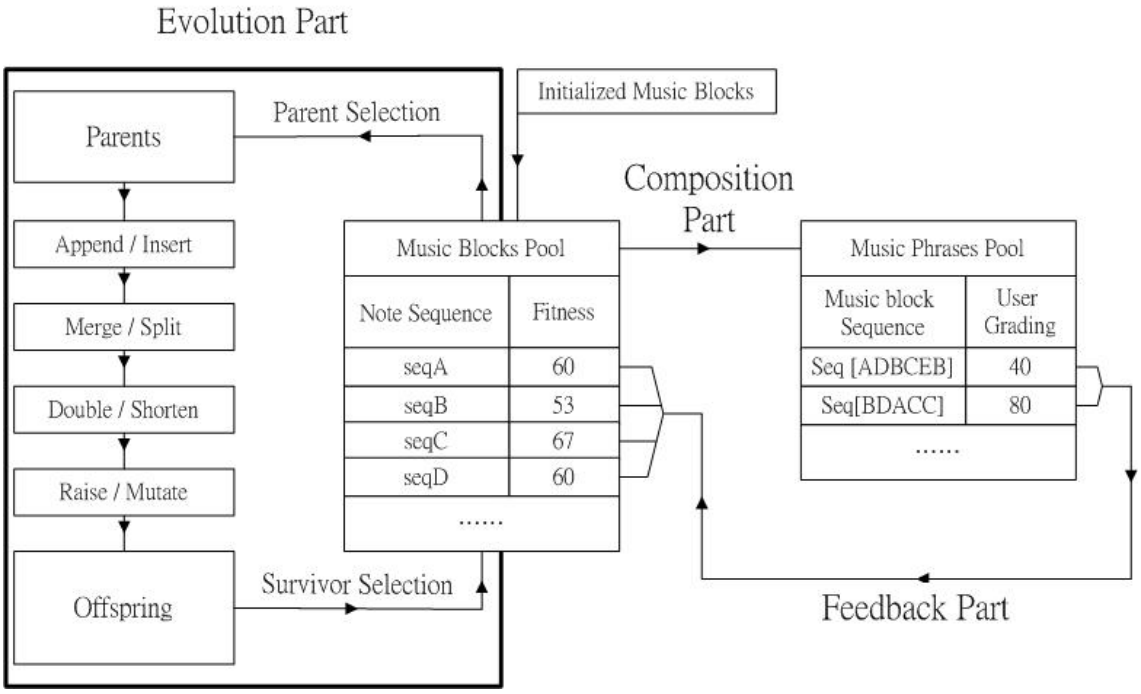

Figure 1: The structure of the CFE framework.

- *Tempo of music* To evolve the tempo of music, a sequence of tempo numbers [9] is adopted as the genotype of the music.

According to the three aforementioned facets, the differences of the present work from those previously proposed frameworks include: (1) For the initialization mechanism, the CFE framework initializes its population with a procedure in between random initialization and complex function initialization. The CFE framework randomly generates pieces of music as individuals probably with limited help of music theory, as described in section 4. (2) For the grading method, because the objective of the CFE framework is to generate personalized music, the real audience composed of only one single person is asked to evaluate the created music instead of using a real audience of many people or other computational techniques. (3) For the design goal, the proposed framework aims at creating short pieces of music instead of creating complete songs, which are usually the goal of previous studies.

3 The CFE Framework

In this section, we describe the CFE framework in detail to demonstrate that for untrained people, creating personalized music by themselves is feasible and practical.

3.1 Overview

The CFE framework contains three major parts: Composition, Feedback, and Evolution. The structure of the system implemented in the present work is shown in Figure 1. In this framework, we try to find the best way to compose music rather than the "best" melody. To be more accurate, the individuals in the evolutionary environment are no longer complete songs but some musical elements or guidelines. The Composition part uses these musical elements and

guidelines to construct new melodies. The composed melodies then wait for the user's response such as making a grade or telling good or bad. After the system receives the information, the Feedback part distributes these feedbacks among the musical elements and guidelines for evaluating how fit these building blocks are. For discovering better methods, the methodology of evolutionary computation is adopted such that new elements are born into the population.

The three parts can be used separately. Therefore, once the user is satisfied with the composed music, no more work is necessary when he or she needs more pieces of music because Composition can be conducted alone. Since Composition and Evolution are isolated, for making use of the domain knowledge, such as the constraints, indications, and implications in the music theory, it is easier to embed such knowledge into Composition than to interfere with the regular operations of the evolutionary algorithm.

3.2 Composition

In the present work, the type of music which we focus on is the theme music of short, specific lengths, say, 8 or 16 measures. These music pieces are named *music phrases* in the framework. Inspired by some pop music that some subsequences appear in a song frequently and repeatedly, we take a layered approach to find out the potentially good sequences of notes. Our system deals with the short theme music by using two levels of hierarchy. The music phrase consists of short, variable-length sequences of notes, called *music blocks*. Composition picks the favored music blocks and fills in the incomplete music phrases until the specified length is reached.

3.3 Feedback

The design of the Feedback part provides the interface for users to give their responses to the system. We simply let users listen to the music phrase composed by Composition and let them grade it in the range from 0 to 100. It is not too complicated for users because the grading is episodic such that users do not have to listen to the music nervously for the need to make real-time responses like applauding. Once the grading is made, the score is distributed to all the music blocks contained in that phrase. Thus, the fitness value of a music block is determined by the average grade of all the music phrases in which the particular music block occurs. The key idea of this design is that good music blocks make good music.

3.4 Evolution

The Evolution part, seeking for the fittest music blocks, plays an essential role in the music composition system. We employ an evolutionary algorithm similar to a typical genetic algorithm, because music blocks can be easily and intuitively represented with a sequence of numbers.

The flow of the employed evolutionary algorithm works in the following way. Firstly, we initialize the population of which the individuals are music blocks containing only one single note with identical fitness values. Then, parent selection chooses one or two music blocks according to the fitness values. The common selection operators, such as tournament selection, can be used for this purpose. The selected parents will go through certain operations, such as appending and inserting, to generate the offspring. Finally, survivor selection decides which music blocks stays in the population and which to be replaced by the newly generated music blocks. In our implementation, we use a fixed population size and remove the music blocks of the lowest fitness. In the following sections, we will introduce the operations designed for dealing with music blocks, including appending, inserting, merging, splitting, doubling, shortening, mutating, and raising.

3.4.1 Append and Insert

The Append operation concatenates two music blocks. The Insert operation, however, puts one music block into the other at a random position to search for better combinations of the two building blocks, as shown in Figure 2.

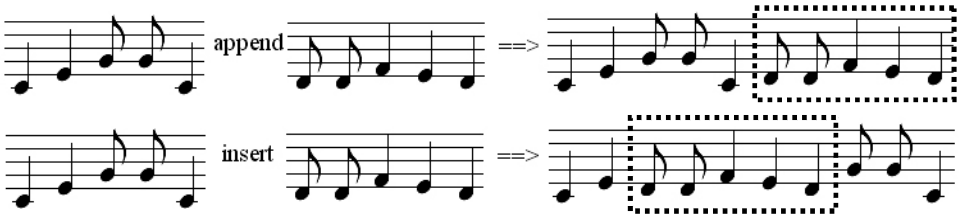

Figure 2: Operations: Append and Insert.

3.4.2 Merge and Split

The Merge operation chooses two adjacent notes in a music block and merges them into a single note with the pitch of one note and the combined tempo length of the two notes. In contrast, the Split operation selects one note and splits it into two of the same pitch and half the length of the original tempo. These two operations adjust the music block configuration locally, as shown in Figure 3.

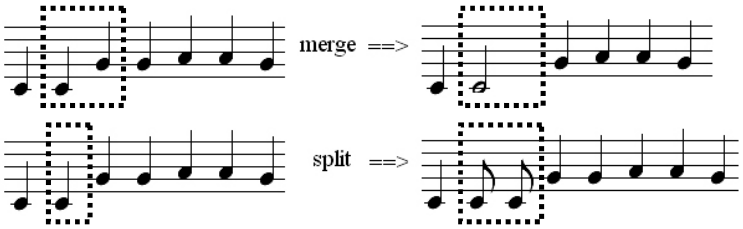

Figure 3: Operations: Merge and Split.

3.4.3 Double and Shorten

These operations operate on the tempo of notes. The Double operation uniformly doubles the tempo length of all the notes in a music block, and the Shorten operation makes the tempo length half, as shown in Figure 4.

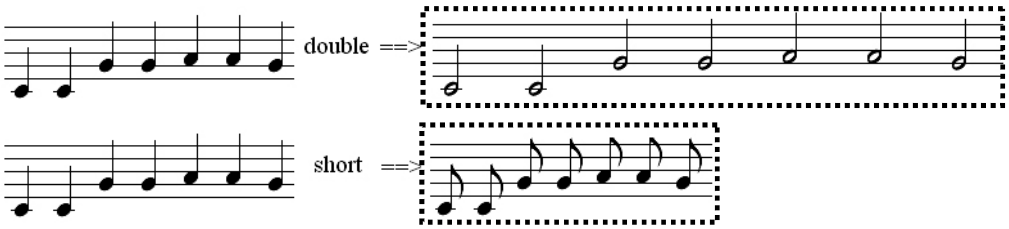

Figure 4: Operations: Double and Shorten.

3.4.4 Mutate and Raise

Different from the Double and Shorten operations, the Mutate and Raise operations only act on the pitch of notes. The note rises or falls in pitch. The Raise operation applies these changes uniformly to all the notes in a music block, while the Mutate operation only works on a randomly chosen note, as shown in Figure 5.

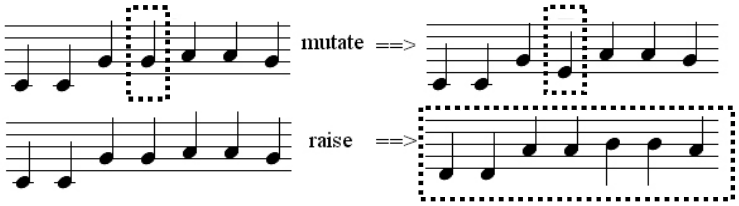

Figure 5: Operations: Mutate and Raise.

4 Auxiliary Functionalities

We implement a reference system based on the described CFE framework to automatically compose and customize music. By grading the music, users express their satisfactory degrees and train the evolutionary environment. After having a test drive, we find that the system needs to be enhanced for two reasons.

First, we should make the grading runs as few as possible. Our system is unlike common evolutionary computing applications which utilize programmed fitness functions. Our individuals are graded by the user. We have to take the human limitations and restrictions into consideration. Users may be tired with a large number of grading runs. As a consequence, we have to reduce the number of grading events.

Moreover, we would like to improve the capability of music composition. As the music composition in the real world, every type of music, such as jazz, blues, and the like, has its own composition rules, styles, and guidelines. For this purpose, we embed some elements of the music theory into the system. In the following sections, we describe the enhancements to help the system compose music.

4.1 Reduce the Grading Runs

In order to reduce the grading runs, we design the following two mechanisms:

Block to block fitness table The block to block fitness table is an $N \times N$ table, where N is an integer parameter, say 20. Considering the overhead, this table is unable to record all the fitness values of relations of each music block pair. Instead, the table records only the fitness values of block pairs which have a top-N fitness value in the music block pool. The table is also used to force two music blocks to be concatenated into one new music block if the fitness of their relation is higher than some specified threshold.

Adaptive evolution For each grading event, our system can change the number of evolution rounds according to the diversity of the newly given scores. For example, the following two conditions with three scores:

- Condition 1: 30, 40, 90;

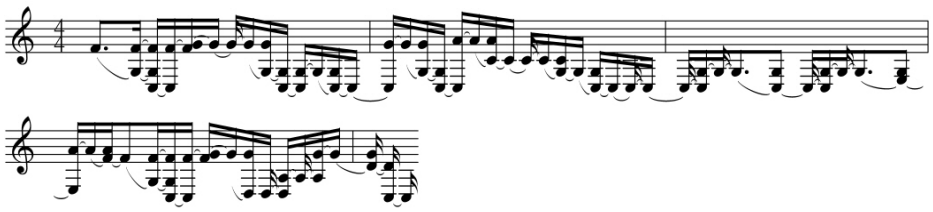

Figure 6: Listen to this showcase at `http://nclab.tw/SM/2007/01/MIDI/2-01.mid`

- Condition 2: 95, 85, 90.

For these two conditions, although their third scores are both 90, the third score in condition 1 is very different from the other two scores. The grade diversity in condition 1 is greater than that in condition 2. We assume that in condition 1, the third score reveals more information of the user preference. Hence, the system executes more evolutionary iterations for condition 1 than it does for condition 2.

4.2 Improve Music Composition

In order to improve music composition, we integrate the basics and elements of the music theory into the system. Our system can refer to the theoretical elements and compose music according to certain standards and/or common sense. In the present work , we employ only the fundamental elements and do not confine the variety of music styles.

Default note to note fitness table In the system, there is a note to note fitness table. It records the fitness of relations of each note pair. During the system initialization, we set the pre-defined fitness into the note to note fitness table. We expect the default fitness table to help compose not-too-bad music at the early stage.

Music block repeat A sequence of notes repeating in the whole song often occurs, such as that in "Happy Birthday" and in "Twinkle Twinkle Little Star". We implement the Composition part to provide this feature. Thus, there are two options with different probabilities for choosing a music block to compose an unfinished music phrase:

- Select a new block which is in the music block pool but not in this unfinished phrase;
- Select an old block which appears in this unfinished phrase.

By doing so, the repeat of music blocks can be controlled by the probability parameter.

5 Showcases

Figures 6 to 13 demonstrate several showcases created by the system discussed in the article. These showcases indicate not only that the proposed framework can accomplish the goal for ordinary users to create music but also that the implemented system can create music of various types, styles, and lengths. If interested, the showcases, as MIDI files, can be accessed through the provided URLs. For more showcases, please visit

`http://nclab.tw/SM/2007/01`

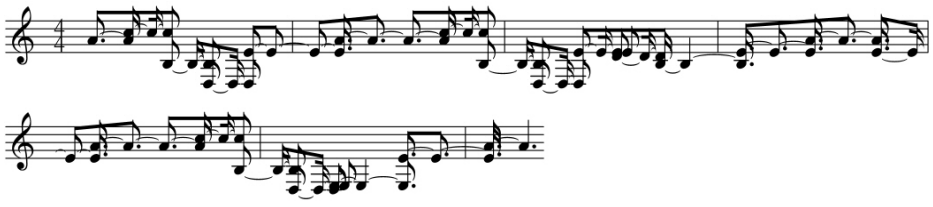

Figure 7: Listen to this showcase at `http://nclab.tw/SM/2007/01/MIDI/2-02.mid`

6 Summary and Conclusions

We started with describing the motivation and the goal of this work. Inspired by previous studies in the literature and compelled by the need of having personalized music, we proposed the CFE framework. We presented the implementation of the system and the functionalities that we used to enhance the system. Finally, we provided several showcases to demonstrate that the proposed system can indeed accomplish its assigned task.

Our work shows that it is feasible and promising for computers to automatically compose customized or personalized music. Although the system currently acts only on short pieces of music, the design may be extended to compose longer music pieces, such as complete songs. The created music can be used in many applications, such as games, cellular phones, background music of web pages, and the like. With this system, everyone effectively has a private music composer at their service.

Acknowledgments

The work was partially sponsored by the National Science Council of Taiwan under grants NSC-95-2221-E-009-092 and NSC-95-2627-B-009-001 as well as by the MOE ATU Program. The author is grateful to the National Center for High-performance Computing for computer time and facilities.

References

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989, ISBN: 0201157675.

[2] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA, USA: The MIT Press, 1992, ISBN: 0262111705.

[3] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, ser. Natural Computing Series. Springer, 2003, ISBN: 3-540-40184-9.

[4] J. A. Biles, "GenJam: Evolution of a jazz improviser," in *Creative evolutionary systems.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 165–187, ISBN: 1-55860-673-4.

[5] H.-C. Chang, "Applying genetic algorithms to music composition: Modeling and simulation of art systems," Master's thesis, National Chiao Tung University, Hsinchu, Taiwan, 2002.

[6] P. Laine and M. Kuuskankare, "Genetic algorithms in musical style oriented generation," in *Proceedings of the First IEEE Conference on Evolutionary Computation (ICEC-94)*, vol. 2, 1994, pp. 858–862.

[7] M. Marques, V. Oliveira, S. Vieira, and A. C. Rosa, "Music composition using genetic evolutionary algorithms," in *Proceedings of the 2000 Congress on Evolutionary Computation (CEC-2000)*, vol. 1, 2000, pp. 714–719.

[8] R. A. McIntyre, "Bach in a box: the evolution of four part baroque harmony using the genetic algorithm," in *Proceedings of the First IEEE Conference on Evolutionary Computation (ICEC-94)*, vol. 2, 1994, pp. 852–857.

[9] A. Pazos, A. Santos del Riego, J. Dorado, and J. J. Romero Caldalda, "Genetic music compositor," in *Proceedings of the 1999 Congress on Evolutionary Computation (CEC-99)*, vol. 2, 1999, pp. 885–890.

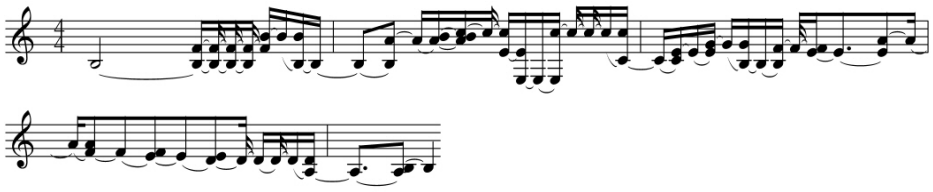

Figure 8: Listen to this showcase at `http://nclab.tw/SM/2007/01/MIDI/2-03.mid`

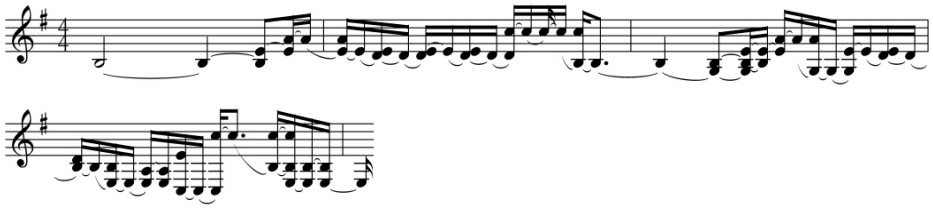

Figure 9: Listen to this showcase at `http://nclab.tw/SM/2007/01/MIDI/2-04.mid`

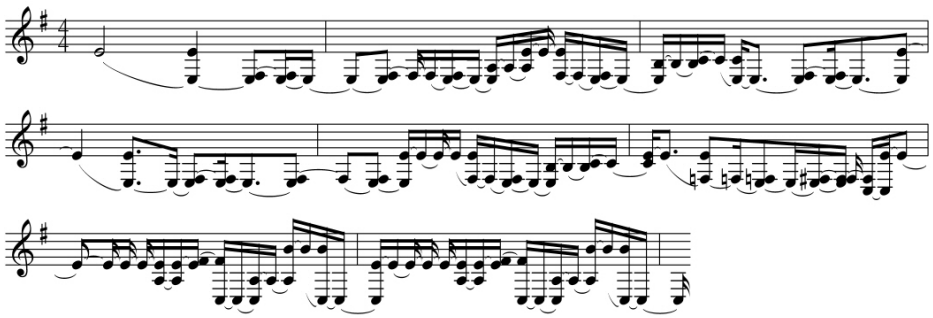

Figure 10: Listen to this showcase at `http://nclab.tw/SM/2007/01/MIDI/3-01.mid`

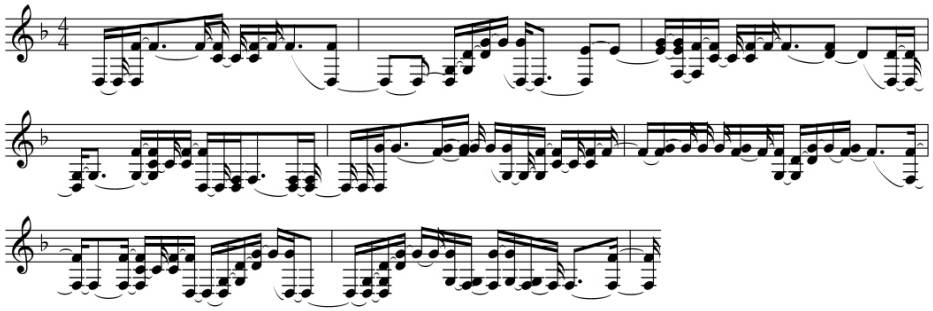

Figure 11: Listen to this showcase at `http://nclab.tw/SM/2007/01/MIDI/3-02.mid`

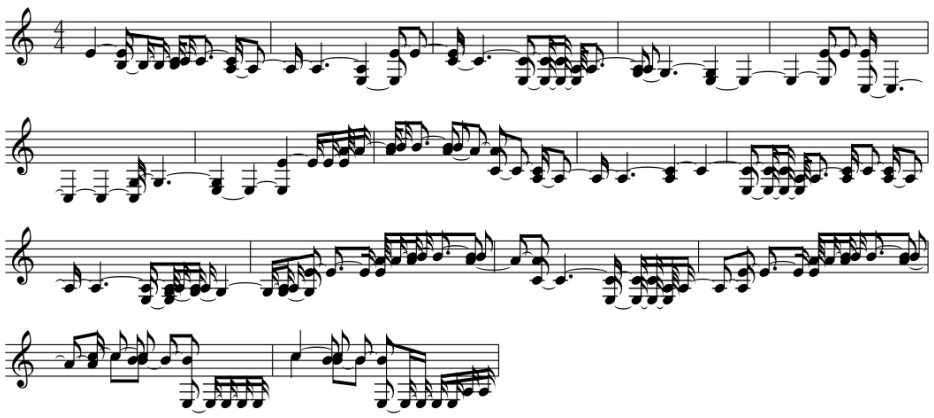

Figure 12: Listen to this showcase at `http://nclab.tw/SM/2007/01/MIDI/4-01.mid`

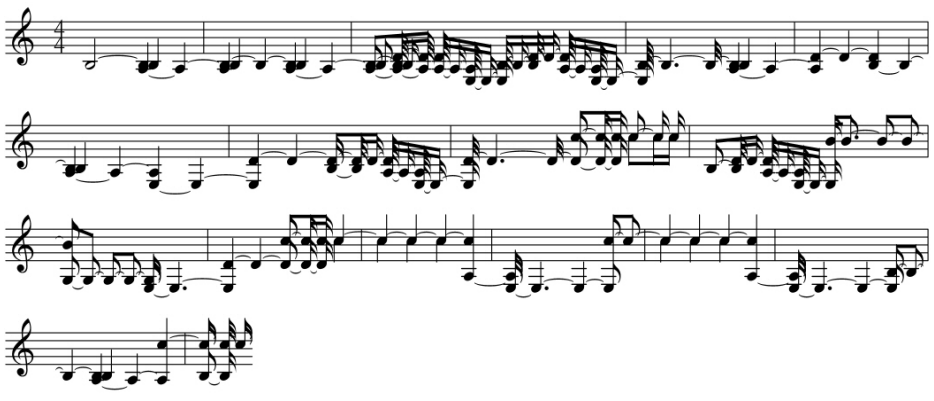

Figure 13: Listen to this showcase at `http://nclab.tw/SM/2007/01/MIDI/4-02.mid`