A TOOL FOR COMPOSING SHORT MUSIC PIECES BY MEANS OF BREEDING

TATSUO UNEMI and EIICHI NAKADA

Department of Information Systems Science, Soka University
1-236 Tangi-machi, Hachiōji, Tokyo 192-8577, Japan

Abstract

This paper presents a design of support system for music composition based on Simulated Breeding, a type of interactive optimization technique of Evolutionary Computing. In our prototype system named Sbeat, each individual in the population is a short music section of sixteen beats including three parts, guitar, bass, and drums. The melody and rhythm are generated by a type of recursive algorithm from genetic information. By selecting favorite piece among scores displayed on the screen, the user listens to the sounds and decides which should be the parents to reproduce offspring in the next generation. Embedding some domain specific functions, we can build a useful tool to make it easier for a beginner to compose his/her favorite music pieces.

Keywords

Interactive evolutionary computing, simulated breeding, music composition, MIDI application.

1 Introduction

Interactive Evolutionary Computing (IEC) [13] is a promising technique to find better solutions in the domains for optimization by user's subjective criteria. Differently from ordinary methods of evolutionary computing, fitness values are not calculated automatically by the predefined evaluation function but are given by the user for each individual in some manner. In the method named *Simulated Breeding* [17], the user directly picks up his/her favorite individuals as the parents for next generation. This means the fitness values can take only 1 (selected) or 0 (not selected). It disables stochastic selection, but has an advantage to reduce the number of user's operations to assign the fitness values.

One of the most successful application fields of IEC is Computer Graphics (CG) art. We can find some number of valuable works by researchers and artists, such as K. Sims [11, 12], P. Todd and W. Lathum [14], S. Baluja [1], G. R. Greenfield [3], and so on. We can obtain the information about this type of tools from A. Rowbottom's survey [10].

Sounds and music are also attractive target domains for application of IEC technique because they are also strongly depending on subjective criteria for artistic production. One of the key issues for building a successful system in this domain is how the user check and select suitable individuals from the population. In the applications for graphic domain, it is relatively easy for the user to catch up the features of all individuals in the population because they can be observed in few seconds. The suitable size of population in graphic domain is from twenty to thirty. The unavoidable issue on the audio domain is that it needs enough duration for the user to measure how the individual is good or bad. This means the size of population must be restricted depending on the time duration the user needs to decide his/her evaluation for individuals, so we need to develop any method to reduce both duration and times of user's operations as much as possible. The style of Interactive Genetic Algorithm (IGA) requires the user to observe all of individuals in the population because he/she has to assign relative fitness value for all of individuals in each generation. From this reason, the style of interaction in Simulated Breeding is better for this domain.

J. A. Biles [2] has proposed an alternative method to solve this problem and implemented it in his system named GenJam which supports to create improvisational phrases in Jazz music. In GenJam, the user listens to endless phrases generated from individual genotypes by turns. The user push "G" or "B" key according to his/her good or bad feeling on the phrase. It is not necessary for the user to assign the fitness value explicitly nor to know the correspondence between the individual and the phrase.

One objective of this research is to investigate possible design of graphical user interface based not on the style of GenJam but on Simulated Breeding. A closely related system has been already developed as Sonomorph system by G. L. Nelson [7, 8], but it still needs to search alternative design of the system to jump up from laboratory to society because the interactive evolutionary part of Sonomorph and its interface design is still in the experimental level. The points that improves in the system presented here is not only on the user interface but also on the type of object tune. The other previous works treats only a simple single track, but this system can breed a piece containing

Proceedings of the 2001 IEEE Systems, Man, and Cybernetics Conference

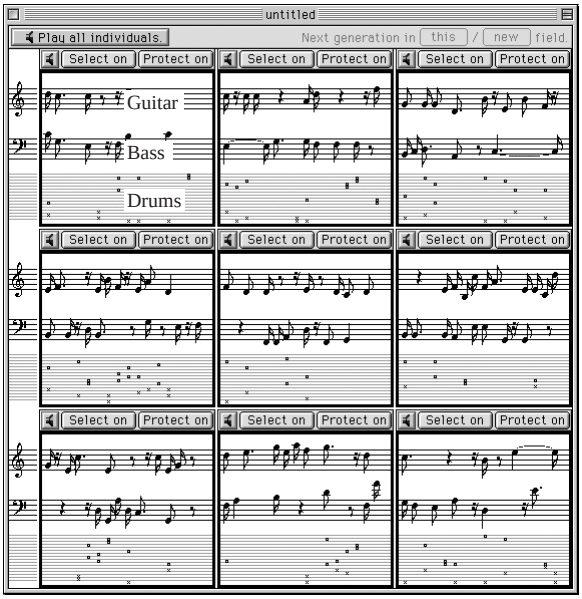

Figure 1: A typical window of Sbeat containing nine initial pieces.

Table 1: Sixteen instruments selected from GM drum kit.

n	*m*	instrument name
0	36	Bass Drum 1
1	41	Lo Floor Tom
2	43	Hi Floor Tom
3	45	Lo Tom Tom
4	47	Low Mid Tom Tom
5	48	Hi Mid Tom Tom
6	50	Hi Tom Tom
7	37	Side Stick
8	38	Acoustic Snare
9	42	Closed Hi Hat
10	44	Pedal Hi Hat
11	46	Open Hi Hat
12	49	Crash Cymbal 1
13	51	Ride Cymbal 1
14	55	Splash Cymbal
15	53	Ride Bell

n = decimal number of genetic code.
m = GM drum kit number.

multi-parts.

Another key issue to lead the application to be successful is the method of morphology, that is, the algorithm to produce the phenotype from the genotype. Considering the structure of score of man-made music, we employ a type of recursive algorithm as described later.

The following sections present our design of prototype system named Sbeat by describing the type of music piece corresponding to each individual, the structure of genotype, the development process, and the design of graphical user interface for breeding and for making it more practical for beginners.

2 Phenotype

The phenotype, the object of selection, is a short music piece including three parts of sixteen beats. The user evaluates each piece by seeing the score displayed on the screen and listening to the sounds generated by General MIDI (GM) [5] device or software attached to the computer. Figure 1 shows a typical window of Sbeat containing nine pieces in subwindows arranged in three by three grids. Each subwindow shows the score of three parts, guitar, bass, and drums, from top to bottom. The guitar part is played as combination of the note in the score and the note lower by two steps from it. The default instrument is Acoustic Nylon Guitar, and the user can change it to other types prepared in GM sound resource, Acoustic Steel Guitar, Electric Jazz Guitar, Electric Clean Guitar, Overdriven Guitar, and Distortion Guitar. The bass part is played in one octave below the displayed score. The default instrument is Acoustic Wood Bass. The alternatives are Electric Bass Fingered, Electric Bass Picked, Fretless Bass, Slap Bass, and Synth Bass. The drum part is played using sixteen instruments of usual drum set selected from GM drum kit shown in Table 1. Two of them can be simultaneously played in the same beat position.

3 Genotype

We designed a genome in a form of two dimensional array of bytes that contains information of each beat for each part. The unit is a byte of which structure is shown in Figure 2. The significant part of four bits are for rhythm, and the rest four bits are for note. The rhythm part is interpreted as follows.

(1) If the most significant bit is 1 then the previous note continues.

(2) If the left three bits are 011_2 then it rests, that is, silent.

(3) Otherwise, it plays something according to the note part.

This implies the probability assignment in which continuation is 50%, rest is 12.5%, and play is 37.5%. We added restriction that prevents for continuing at the first beat position in every eight beats to produce rhythms

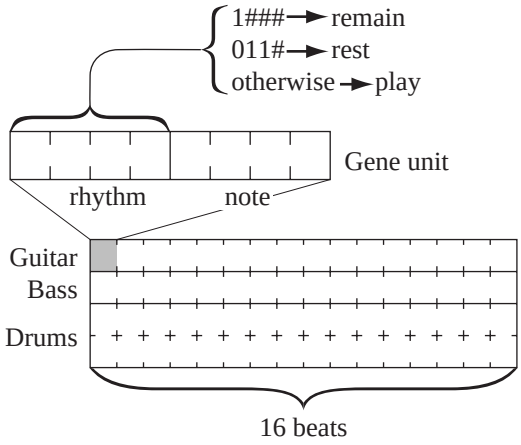

Figure 2: Structure of genotype.

```
fill_notes(g, w) begin
    x := (w + 1) / 2;
    if x ≤ 1 then k[0] := (g[0] & 0111_2) + 4;
    else fill_notes(g, x);
    i := x;
    while i < w do begin
        s := k[i − x] + delta(g[i]);
        k[i] := min(max(s, 0), 15);
        i := i + 1
    end
end
```

Figure 3: Recursive algorithm to generate the score from genotype. $k[i]$ is the $i+1$st integer for the basic melody sequence. The function **delta**(x) returns an integer in $[-2,2]$ based on the note part of the value of argument x. *This figure was corrected in Octover 16, 2001.*

with relatively stable feeling. This means that rest is 12.5% and playing is 87.5% at these positions.

For the guitar and bass parts, genetic information is constructed as one byte par one beat. For the drum part, two bytes are assigned for each beat to select maximally two instruments described in the previous section. Upper four bits of the second byte are ignored in this case.

The genetic operations currently implemented are mutation and crossover. They are in the ordinary style in simple Genetic Algorithm. Mutation is done by flipping each bit in a constant probability, mutation rate μ. The current setting is μ = 5%. Crossover operation is always done if the user select more than one parents from the population by the style of one point crossover dividing genome into two parts. The cutting position is randomly chosen from 15 beat boundaries. It means any different parts of the same timing are never separated by crossover operation. This is the current implementation in Sbeat, but it could be better to decide the cutting position independently among parts to bring more diversity in the offspring.

4 Development

The development process that maps genotype to phenotype is important to produce better candidates in both initial population and mutants because a trivial random algorithm usually cannot generate any acceptable solution as a seed for evolution.

One useful bias is the probability to appear some features in phenotype. We can know some statistical characteristics over feasible solutions among all of possible solutions in some domain. The design of genotype described above is also based on this sort of consideration.

Another point we should consider is the structure of phenotype from the view point to measure similarity between candidates. Similarly to another domain, a music piece is also has some number of features, such as chord, melody, speed, rhythm, and so on. It means that it has multi-dimensional similarity space. It isn't a necessary condition but is better to make the genotype to include a similar structure to the phenotype, because the usual mutants should somewhat resemble the ancestor.

We introduce a type recursive algorithm to develop a basic melody from a genotype. It is useful to introduce similarity between the data structure of genotype and the listener's feeling. The different point from the researches on fractal music composition is that the expected effect of the algorithm is not on the generated score itself but the difference among mutants. From this point of purpose, we designed the algorithm as shown in Figure 3 to fill out all of sixteen beats positions by integers from 0 to 15. The integer for ith beat is calculated using ith gene unit and the integer value of $(i-w/2)$th beat, where w is the greatest value of 2^j less than i, when total number of beats is also 2^n. Here, j and n are integers. This algorithm works even when the total number of beats is not 2^n by dividing the sequence into the first half and the latter half. Another method can be to divide it by the least divisible prime number.

Each integer for beat is not interpreted as a note on the twelve pitches on equal tempered scale but two octaves plus one step of minor scale from A3 to B5.

The score of guitar part is generated by combining the rhythm information with the basic melody described above. If the rhythm part indicates continuation or rest then the note information in the basic

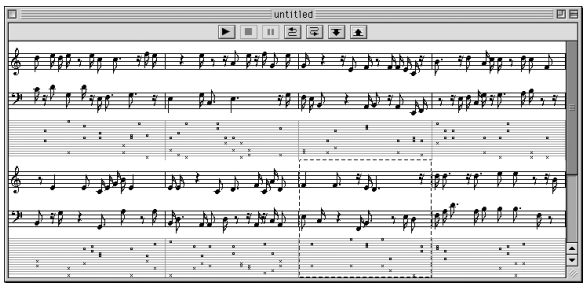

Figure 4: Integration window.

Figure 5: Part option dialog.

melody at the beat position is ignored. For the bass and drum parts, the user can select whether it refers to rhythm information from independent data or data used for guitar part. A type of synchronized score is generated if all of parts use unique information for the rhythm. The note for bass part is calculated by adding a bias to the basic melody according to the genetic information.

This method for development is used for both drawing the score in the subwindow and preparing the tune sequence data to play by the computer[1] .

5 Graphical User Interface

One of the important parameters we should determine is the number of individuals simultaneously shown on the screen. As shown in Figure 1, the population size is nine here. This number was determined through some preliminary experience on several different numbers of individuals. Twelve might be possible, but more than sixteen is seems to be redundant by the following reason. The user cannot memorize many pieces to compare them each other because it takes long time to check individuals in acoustic domain. On the contrast from the case of CG, it is difficult to compare them by listening to them simultaneously.

The basic idea to play sound by clicking the subwindow of individual is quite same as Sonomorph, GA Music [6], and IEC application for tuning a hearing

[1] Current implementation is using Tune Player Functions of QuickTime Music Architecture on MacOS.

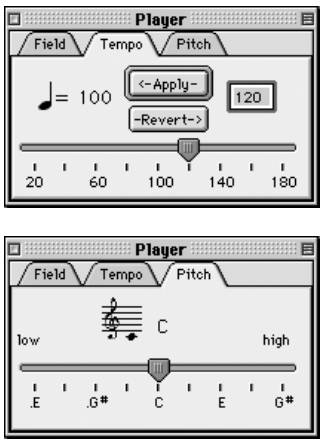

Figure 6: Player option dialog.

aid by M. Ohsaki [9]. The early version of Sonomorph and other systems use simple push buttons to play tune and to assign the fitness value or to select it as a parent for each individual. They don't provide any information about the sound visually. The newer version of Sonomorph shows simple dashes that corresponds to the score. As an improvement, Sbeat shows scores for each individual on the screen so that the user can imagine the sound before listening to it. It is easy to read the score if the user is an expert for music, but the beginner can also learn to catch up the outline of sound through the usage of this system.

The user can choose one of two play back modes, repeating one tune and playing all tunes. The former mode repeatedly plays the selected tune until clicking it again or clicking another individual. The latter mode plays tunes for all of individuals in the population consecutively. It is convenient to find the best candidate from the population.

The other important features of Sbeat are multi-field user interface and integration window. The former one was proposed in [16] for CG application, which enhances the diversity of production just similarly to island model of evolutionary theory. The user can breed some different populations using several field windows independently, and then make some individuals to migrate to the other fields by *drag and drop* operation. The integration window as shown in Figure 4 is to collect individual sections into a longer musical piece. The user can copy a score of individual into any section in an integration window, and edit them with a simple editing functions, copying and pasting, deletion, and shifting left and right.

The user can also change the optional attributes for each part by a dialog window shown in Figure 5.

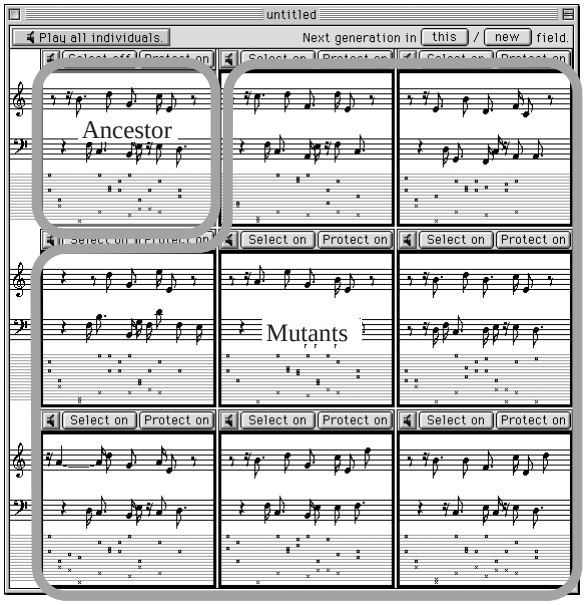

Figure 7: Individuals after mutation. The individual at upper left corner is the ancestor of mutants.

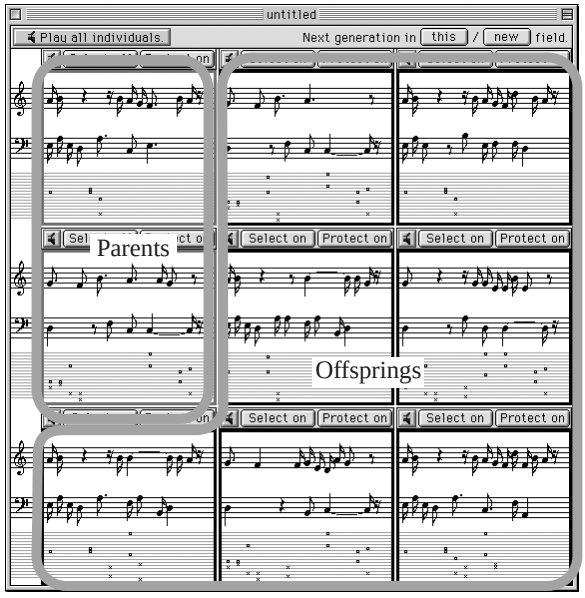

Figure 8: Individuals after crossover. The individuals at upper left corner and middle left position are the parents.

Checking off the button in the column labeled with "Play" stops the sound of the part. Checking on the button in the column labeled with "Protect" prohibits mutation in the corresponding part of genome. These two attributes allow the user to breed any part independently. Buttons labeled with "Sync" are for control of dependency on the rhythm with guitar part as described in the previous section. A pop-up menu appears to change the length of the part when the user pushes the button in the column labeled with "Length." The candidates are "full," "1/2" and "1/4." A shorter score is iterated by twice or four times to fill in the sixteen beats when it is 1/2 or 1/4. The next column labeled with "Instrument" includes pop-up menus to select the type of sound for each part. Additionally to the alternatives described in section 2, the user can select arbitrary instrument among all of the GM sound resources. The right most column is to control some types of effects, pan, volume, octave shift, reverb, chorus and celeste.

Another dialog window is available to change the tempo and pitch. The upper part of Figure 6 shows the dialog window to set the tempo within the range from 20 to 180 quarter notes par minutes. The lower one is to shift the pitch of all of the parts by half a step within one octave and five steps.

6 Examples of Breeding

Breeding process begins with an initial population of which genetic information is randomly generated. The user would reset all of individual by replacing random genome again, if the user could not find an acceptable score in the population. It might be necessary to reset them several times because the size of population is relatively small. Before seeking the feasible initial ancestors, the user may change the settings of the tempo and the part options using dialog windows described in the previous section.

Selecting an individual as a parent, the other individuals can be replaced into the mutants. Figure 7 shows an example of population just after mutation. The individual displayed at the upper left subwindow is the parent and the others are its mutants. The recursive algorithm described in Section 4 guarantees the resemblance of mutants to the parent.

If the user found more than one different types of favorite individuals, offsprings can be combinations of them spawning through crossover operation. As described before, it is done in a style of one point crossover, but the phenotype produced from the new genotype is not a simple concatenation of the first half of a parent and the latter half of another parent. Some length of the front part is inherited from a parent however the rest part of score is produced from gene of both parents by a recursive procedure as described in Section 4. Figure 8 shows an example of the window just after a crossover operation.

If one or two parts of tune satisfies the user's criteria but the other should be still revised, it would be

better to protect completed parts using the part option dialog window.

7 Conclusions

A type of design on a tool to create short music pieces by means of breeding was presented above. The phase of development of this system is still at the beginning, but we have obtained some expectation on usefulness of the proposed method toward an practical support tool for beginners to compose their favorite music. Our future works will includes:

(1) to increase the number of parts up to sixteen,
(2) to make it able to breed instrument-dependent effects such as breath, slur, and fluctuation,
(3) to add a gene editor to allow the user to modify a genotype directly,
(4) to add a score editor to allow the user to modify a score directly and to bring it back into the genotype and
(5) to build more large system to combine the other features of music such as organization of pieces, chord progression, orchestration, and so on.

Some other researchers have already developed similar systems for rhythms [15] and sounds [4]. It must be valuable to combine their ideas to our system.

The authors are hoping for many people to enjoy such a evolutionary tool for subjective selection to create new culture.

References

[1] Baluja, S., Pomerleau, D. and Jochem, T. (1993) "Simulating User's Preferences: Towards Automated Artificial Evolution for Computer Generated Images," CMU Computer Science Technical Reports, CMU-CS-93-198.

[2] Biles, J. A., Anderson, P. G. and Loggi, L. W. (1996) "Neural Network Fitness Functions for a Musical IGA," IIA'96/SOCO'96. International ICSC Symposia on Intelligent Industrial Automation And Soft Computing, B 39–44.

[3] Greenfield, G. R. (2000) "Evolving Expressions and Art by Choice," Leonardo, Vol. 33, No. 2, 93–100.

[4] Iwai, M. (1994) "Tuning Parameters of FM Sound Resources by a Genetic Algorithm," Summer Programming Symposium – Entertainment and Computers (in Japanese).

[5] Midi Manufactures Association (1995) "The Complete MIDI 1.0 Detailed Specification," Midi Manufactures Association, La Habra, CA.

[6] Moore, J. H. (1995) http://www-ks.rus.uni-stuttgart.de/people/schulz/fmusic/gamusic.html

[7] Nelson, G. L. (1993) "Sonomorphs: An Application of Genetic Algorithms to Growth and Development of Musical Organisms," Proceedings of the Fourth Biennial Art & Technology Symposium, Connecticut College, 155–169.

[8] Nelson, G. L. (1995) "Further Adventures of the Sonomorphs," Proceedings of the Fifth Biennial Art & Technology Symposium, Connecticut College, 51–64.

[9] Ohsaki, M. and Takagi, H. (2000) "Design and Development of an IEC-based Hearing Aid Fitting System," Proceedings of the Fourth Asian Fuzzy Systems Symposium, Tsukuba, Japan, 543–548.

[10] Rowbottom, A. (1999) "Evolutionary Art and Form," in Bentley, P. J. (ed) Evolutionary Design by Computers, 261–277, Morgan Kaufmann.

[11] Sims, K. (1991) "Artificial Evolution for Computer Graphics," Computer Graphics, Vol. 25, No. 4 (SIGGRAPH 91), 319–328.

[12] Sims, K. (1992) "Interactive Evolution of Dynamical Systems," in F. J. Varela and P. Bourgine (Eds.), Toward a Practice of Autonomous Systems – Proceedings of the First European Conference on Artificial Life, 171–178, MIT Press.

[13] Takagi, H. (1998) "Interactive Evolutionary Computation – Cooperation of computational intelligence and human KANSEI," 5th International Conference on Soft Computing (IIZUKA'98), 41–50, World Scientific.

[14] Todd, S. and Lathum, W (1992) "Evolutionary Art and Computers," Academic Press.

[15] Tokui, N. and Iba, H. (2000) "Music Composition with Interactive Evolutionary Computation," Proceedings of the Third International Conference on Generative Art, Milan, Italy.

[16] Unemi, T. (1998) "A Design of Multi-Field User Interface for Simulated Breeding" Proceedings of the third Asian Fuzzy Systems Symposium, 489–494.

[17] Unemi, T. (1999) "SBART2.4: Breeding 2D CG Images and Movies, and Creating a type of Collage," The Third International Conference on Knowledge-based Intelligent Information Engineering Systems, 288–291.