A New Musical Tool for Composition and Play Based on Simulated Breeding

Tatsuo Unemi and Manabu Senda

Department of Information Systems Science, Soka University

1-236 Tangi-machi, Hachiōji, Tokyo 192-8577, Japan

unemi@iss.soka.ac.jp, msenda@intlab.soka.ac.jp

Abstract: This paper presents the design of new personal computer tool for interactive composition and play. The tool, named SBEAT, utilizes MIDI technology and is based on Simulated Breeding, an optimization technique from Evolutionary Computing, combined with a method of fractal composition. An individual, the unit of breeding, is a bar of sixteen beats including eight parts - four parts of solos and another four parts of rhythm section. The melody and rhythm are generated by a recursive algorithm from genetic information. By listening to and selecting favorite pieces from scores displayed on the screen, the user decides which should be the parents to reproduce offspring in the next generation. By collecting some bred bars and their mutants, the user can easily build up a short tune.

1 Introduction

Simulated breeding [1] is a type of Interactive Evolutionary Computing (IEC) [2], a promising technique to find solutions in some domain, and for optimization based on a user's subjective criteria. The user directly picks up his or her favorite individuals from a population as parents for the next generation.

One of the most successful application fields of IEC is Computer Graphics (CG) art, in which we can find numerous valuable works by researchers and artists, such as K. Sims [3, 4], P. Todd and W. Latham [5], S. Baluja [6], G. Greenfield [7], and so on. Information about this type of tool can be found in A. Rowbottom's survey [8].

Sound and music are also attractive targets for the application of IEC techniques because they are also strongly dependent on subjective criteria for artistic production. One of the key issues for building a successful system in this domain is how the user checks and selects suitable individuals from a population. J. A. Biles [9] has proposed an alternative method to solve this problem and implemented it in his system, named GenJam. This system helps the user to create improvisational phrases in Jazz music. In GenJam, the user listens to endless phrases generated from individual genotypes in turn. The user pushes the "g" or "b" key according to his or her good or bad feelings about the phrase. It is not necessary for the user to assign fitness values explicitly, or to know the correspondence between the individual and the phrase.

One of the goals of this research is to propose a possible design for a graphical user interface based, not on the style of GenJam, but on Simulated Breeding. A closely related system has already been developed as the Sonomorph system by G. L. Nelson [10, 11]. Alternative designs of the Sonomorph system are still needed to make the jump up from the laboratory to general society, because the interactive evolutionary part of

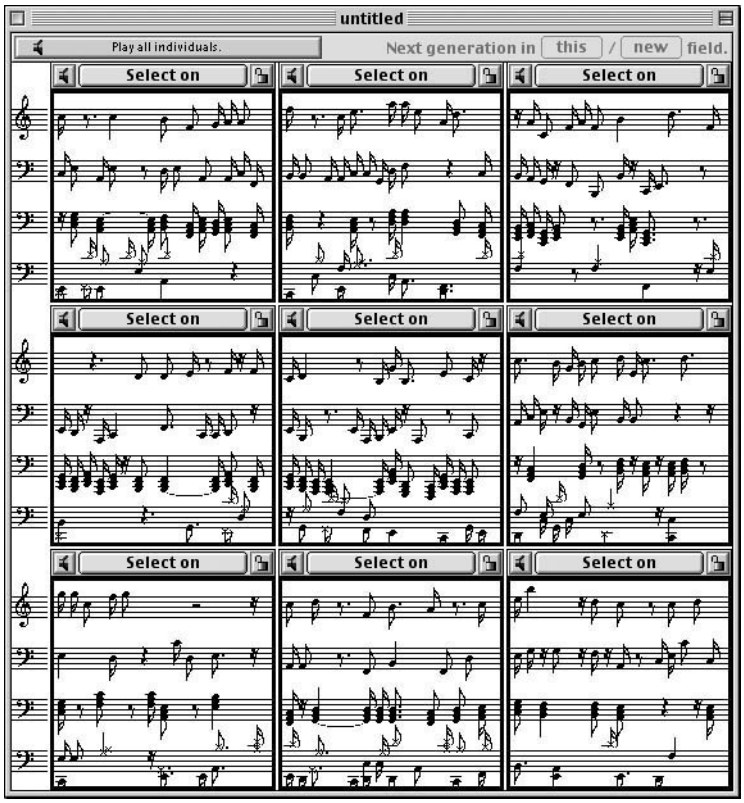

Fig. 1: A typical field window of SBEAT containing nine initial individuals.

Sonomorph and its interface design are still at the experimental level. The points of improvement to the system presented here are not only in the user interface, but also concern the type of object tune. Previous work treats only a simple single track, but this system can breed a piece consisting of multiple parts.

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

The following sections present our design of a prototype system named SBEAT by describing the type of music piece corresponding to each individual, the structure of the genotype, the development process, and the design of the graphical user interface for breeding. An example of the resulting product is shown in the appendix after the conclusion.

2 Phenotype

The phenotype, the object of selection, is a bar including sixteen beats of at most eight parts. The user evaluates each bar by viewing the score on the screen and listening to the sounds generated by a General MIDI (GM) [12] device or software attached to the computer. Fig. 1 shows a typical field window of SBEAT containing nine subwindows arranged in a three by three grid. Each sub window shows the score of four parts chosen from eight: soprano, alto, tenor, barytone, bass, piano, drums, and percussion. The first five parts from soprano to bass are played with at most one note for each beat. The piano part is played by a combination

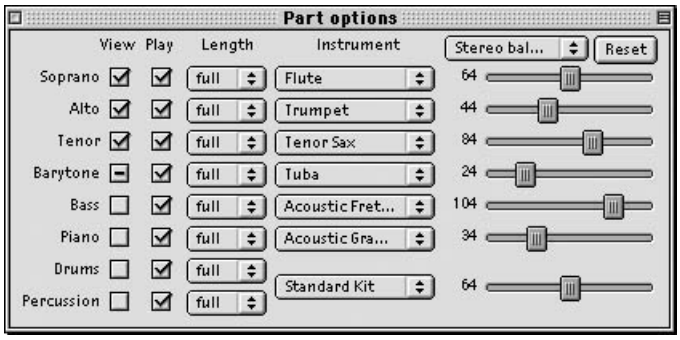

Fig. 2: Part option dialog.

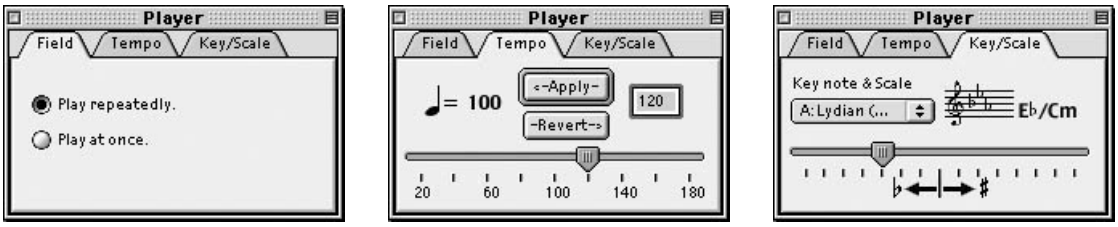

Fig. 3: Player option dialog.

of the three notes. The drum part is played using instruments from the usual drum set selected from the GM drum kit, and the others are allocated for the percussion part. Two of the selected instruments can be played simultaneously in the same beat position for both parts.

The user can also change some attributes for each part using a dialog window shown in Fig. 2. Checking off the button in the column labeled “Play” stops the sound for the part. The pop-up[1] menu buttons in the column labeled as “Instrument” are to select the type of sound for each part. The menu of each part contains some names of instruments suitable for the part, followed by a special item labeled “other ...” that allows the user to select an arbitrary instrument from all of the GM sound resources. A pop-up menu, a reset button and sliders at the right end of the window control some effects: pan; volume; octave shift; reverb; chorus and celeste.

Another dialog window is available to change tempo and scale. The middle part of Fig. 3 shows the dialog window to set tempo within the range from 20 to 180 quarter notes per minute. The right one is to change key and scale.

3 Genotype

We designed the structure of the genotype for one individual as a collection of three types of chromosomes for rhythm, melody, and fluctuation of velocity. The number of chromosomes is three times the number of parts, that is, $3 \times 8 = 24$. Each chromosome is a string of integers each of which corresponds to each beat.

The element of chromosome for rhythm is a four-bit integer that is interpreted as follows.

[1] “Pop-up” is the name used in the MacOS Toolbox for “option” menu in Motif and other GUI took kits.

(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.

(3) Otherwise, it begins play 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 a restriction that prevents continuation at the first beat position in every eight beats to produce rhythmic patterns of a relatively stable feeling. This means that rest is 12.5% and play is 87.5% at these positions.

For the solo and piano parts, the element for melody is constructed as four bits per one beat. These integers are used to give parameter values to the recursive algorithm described in the next section to calculate the sequence of notes for each part. For the drum and percussion parts, nine bits are assigned for each beat to select maximally two instruments for each beat.

The chromosomes for velocity are in the same form as for the melody, that is, each integer is represented by four bits. They are also used to give parameter values to the recursive algorithm to calculate the sequence of velocity values for each part.

4 Development

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

One useful bias is that of the probability that some features will appear in a phenotype. The design of the genotype described above is based on this sort of heuristic knowledge.

Another useful method is to introduce an algorithmic development. We employ a type of recursive algorithm to develop a basic melody from a genotype as shown in Fig. 4. It fills out all of the sixteen beats with integers from 0 to 15. The integer for ith beat k_i is calculated by adding ith gene g_i and $k_{i-w/2}$, where w is the greatest value of 2^j less than i, when total number of beats is 2^n. Here, j and n are integers. Concretely saying, g_1 gives the basis of the whole of the sequence, g_2 is added to every second beat $(k_2, k_4, k_6, \ldots)$, g_3 is added to every fourth beat beginning with the third beat (k_3, k_7, k_{11} and k_{15}), g_4 is added to every fourth beat (k_4, k_8, k_{12} and k_{16}), g_5 is added to every eighth beat beginning with the fifth beat (k_5 and k_{13}), and so on. 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.

Each integer of the result is not interpreted as a note on the twelve pitches of the equally tempered scale but two octaves plus one step of the natural minor scale starting from A when the key is C/Am. Scale is changeable by operating the player dialog described in section 2 above.

The chromosome for the melody of the piano part is used to give values of $g[i]$ $(i = 0, 1, \ldots, 15)$ to the recursive algorithm to make a basic melody. The score of the piano part is generated by combining them with the rhythm information. If the rhythm part indicates continuation or rest then the melody information at the corresponding beat position is ignored. In addition to the note of basic melody, two notes, two steps above and

```
fill_notes(g, w) begin
    x := (w + 1) / 2;
    if x ≤ 1 then k[0] := (g[0] & 0111₂) + 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
```

Fig. 4: Recursive algorithm to generate a basic melody from genotype. $k[i]$ is the $i+1$st integer for the sequence of basic melody. The function **delta**(x) returns an integer in $[-2, 2]$ from the value of argument x.

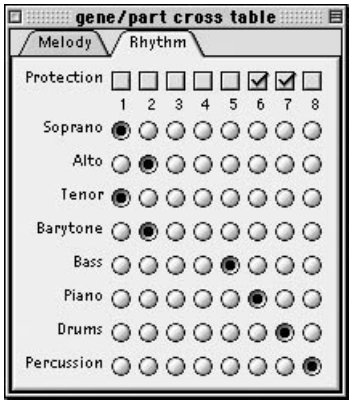

Fig. 5: Dialog window of gene/part cross table.

below the basic note, are played at the same time.

Integers in the chromosomes for the melody of the solo parts, from soprano to bass, are interpreted as one of three signs, "+," "0" and "−." If the sign is "+," the integer of note in the phenotype is calculated by adding two steps to the note of the basic melody. It becomes the note below the basic melody by two steps if the sign of the gene is "−." The note of the basic melody is copied for "0." This method helps to produce musical harmony of multiple parts rather than discord.

Four of nine bits in an integer in the chromosomes for melody of drum and percussion parts are used to indicate the first instrument, and the remaining five bits are for the second instrument. The first instrument is selected from 16 candidates in a GM drum kit. The second instrument is selected from 21 for drums and 26 for percussion. If the first and second instrument are the same or the integer for the second is greater than the number of candidates, only one instrument is played.

There are some options to control the development process. The pop-up menu buttons in the column labeled "Length" in the part option dialog shown in Fig. 2 above, make the score as repetition of half or quarter length of an ordinary bar. Latter half or three quarters of the chromosomes are ignored in these cases.

Another dialog window named gene/part cross table shown in Fig. 5 allows the user to change the correspondence between chromosomes and parts. The user can assign one chromosome to more than one part. The parts sharing the same rhythm chromosome are played synchronously. The parts sharing both rhythm and melody are played in unison.

The score information is finally translated into the tune sequence to be played by the computer[2].

[2] Current implementation uses Tune Player Functions of the QuickTime Music Architecture on MacOS.

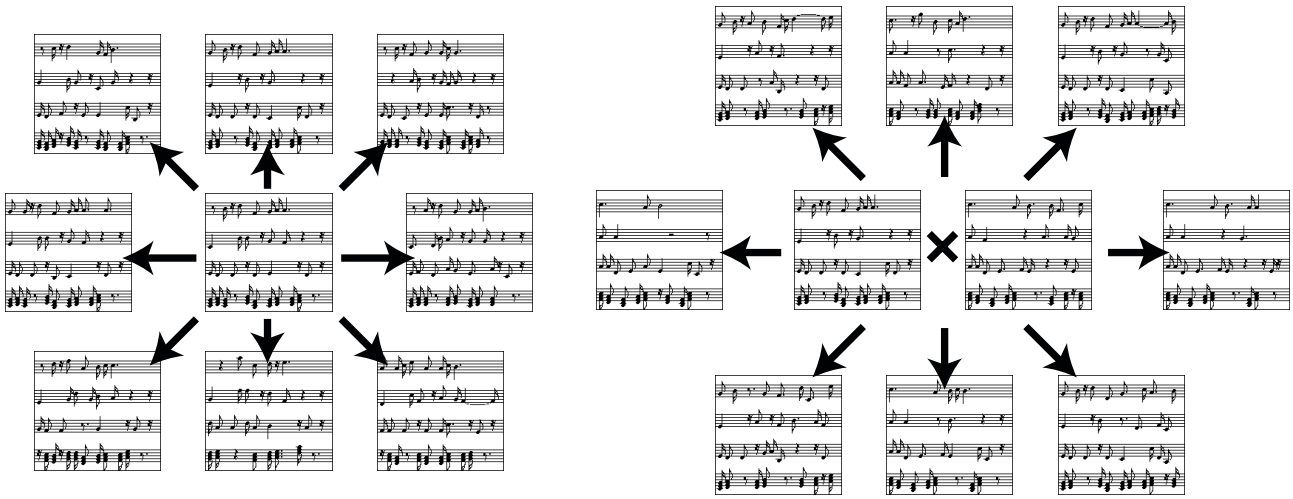

Fig. 6: Typical examples of mutation produced from a single parent on the left, and crossover from two parents on the right.

5 Breeding

The genetic operations currently implemented are mutation and crossover. These are in the ordinary style of the basic Genetic Algorithm. Mutation is done by flipping each bit with a constant probability, mutation rate μ. The current setting is μ = 5%. The crossover operation always occurs if the user selects more than one parent from the population. One point crossover is used, dividing each chromosome into two parts. The cutting position is randomly chosen from 15 beat boundaries. Fig. 6 shows typical examples of mutation produced from a single parent and crossover produced from two parents.

Field windows have a lock button at the top right of each individual, by which the user can prohibit modification of a genotype. The check box in the gene/part cross table allocated to each chromosome ID is to switch the protection of each chromosome against operations of reset, mutation, and reproduction. These protection functions allow the user to breed each of the parts independently by prohibiting destruction of completed parts.

6 Graphical User Interface

It is important to design an effective graphical user interface for any type of interactive software. This section describes the unique features in SBEAT.

6.1 Population size

One of the important parameters for IEC applications is the number of individuals simultaneously shown on the screen. As shown in Fig. 1, the population size is nine here. This number was determined through some preliminary experience with several different numbers of individuals. Twelve may be possible, but more than sixteen seems redundant for the following reason. Users can hardly memorize many pieces to compare them because it takes some time to check an individual in the acoustic domain. In contrast to the case of CG, it is difficult to compare acoustic samples by listening to them simultaneously.

6.2 See, play and listen

As described in section 2, only four of eight parts are shown in a sub window because of screen space restrictions. The user can choose the parts to be shown by clicking on the check box in the column labeled "View" of the part option dialog. This exposes or hides the score of the corresponding part in the field window.

The basic idea to play sound by clicking the sub window of an individual is quite the same as Sonomorph, GA Music [13], and the IEC application for tuning a hearing aid by M. Ohsaki [14]. The early version of Sonomorph and other systems use simple push buttons to play a tune, 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 correspond to the score. As an improvement, SBEAT shows a score 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 a music expert, but the beginner can also learn to catch the outline of the sound through using this system. By pushing the button labeled "play all individuals" at the top left corner of the field window, the user can listen to all of the nine individuals in the population consecutively. This is useful to find the best candidate from the population as described in [14].

The user can choose one of two play modes, repeating one tune and playing it once, by operating the player dialog shown in Fig. 3. The former mode repeatedly plays the selected tune until clicking it again or clicking another individual. The latter mode plays a tune once.

6.3 Migration and integration

The other important features of SBEAT are the multi-field user interface and the score window. The former was proposed in [15] for CG applications, to enhance the diversity of production similarly to the island model of evolutionary theory. The user can breed different populations using several fields independently, and then make some individuals migrate to another field by the *drag and drop* operation. The score window is used to collect individual bars into a longer musical piece. The user can copy the score of an individual into any bar in a score window, and edit it with some simple editing functions. By pressing the *option* key during the dragging operation, only the indicated part can be copied. This function makes it easier to breed each part independently in combination with the protection function described in section 4.

In a score window, each bar has different settings of some options including play on/off, instrument, effect control and tempo. This function helps the user to integrate a variety of bars in the form of an ensemble, Jazz tune including theme and improvisatorial solos, a variety of chord progressions, and so on.

6.4 Genome editor

Breeding is a good method for producing novelty, but it is mostly redundant when we know what type of direct modification brings a better result. The genome editor was designed to answer this requirement by allowing the user to edit chromosomes directly. Fig. 7 shows two windows, a score of an individual and an editing panel. The user selects a part to be edited using a pop-up menu, and then operates buttons allocated to beats and chromosomes.

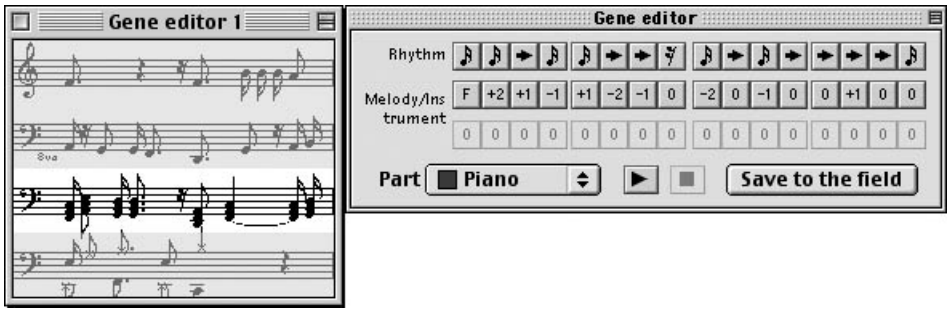

Fig. 7: Genome editor.

7 Conclusions

A type of design tool to create short pieces of music by means of breeding was presented above. This system is still at the beginning of its development, but we have obtained some expectation of the usefulness of the proposed method as a practical support tool for beginners to compose their favorite music. Our future work will include:

(1) to increase the number of parts up to sixteen,

(2) to make the system able to breed instrument-dependent effects such as legato, staccato, trill, and so on,

(3) to build a larger system to be combined with the other features of music such as the organization of pieces, chord progressions, orchestration, and so on.

Some other researchers have already developed similar systems for rhythms [16] and sounds [17]. It would be valuable to incorporate their ideas within our system.

The current version of SBEAT running on MacOS 9 is freeware that any person can download from the URL:

`http://www.intlab.soka.ac.jp/~unemi/sbeat/` .

A number of sample tunes in MIDI file format are also available from the above web page. The authors hope many people will enjoy such an evolutionary tool for subjective selection to create new culture.

Acknowledgement

The authors thank Eiichi Nakada who cooperated with the first author to develop the previous version of SBEAT[18] that treats only three parts.

References

[1] 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.

[2] Takagi, H. (2001) "Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation," Proceedings of the IEEE, Vol. 89, No. 9, 1275–1296.

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

[4] 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.

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

[6] 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.

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

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

[9] 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.

[10] 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.

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

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

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

[14] 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.

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

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

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

[18] Unemi, T. and Nakada, E. (2001) “A Tool for Composing Short Music Pieces by Means of Breeding,” Proceedings of the IEEE International Conference on Systems, Man and Cybernetics 2001, 3458–3463.

Appendix: an example of production.