A METHOD FOR SOUND SYNTHESIS BASED ON GENETIC ALGORITHMS

Jônatas Manzolli [1,2], Adolfo Maia Jr.[1,3], Jose Fornari [1,4] and Furio Damiani [1,4]
[1]Interdisciplinary Nucleus for Sound Studies – NICS
[2]Arts Institute Music Department – DM/IA
[3]Applied Mathematics Department - IMECC
[4] School of Electrical and Computer Engineering – DSIF/FEEC
University of Campinas – UNICAMP
BRAZIL
[jonatas, adolfo, fornari, furio]@nics.unicamp.br

ABSTRACT

A new method for interactive sound synthesis based on the application of genetic algorithms is presented. This method generates sequences of waveforms through the application of genetic operators on an initial population of waveforms. The waveforms are suitable to be played by a conventional wavetable synthesizer engine. We describe how the waveforms can be treated as genetic code, the fitness evaluation methodology and how genetic operations such as crossover and mutation are used to generate new waveforms. Finally, we discuss the results evaluating the generated sounds.

INTRODUCTION

As it is seeing in late researches of electronic music, genetic algorithms (GAs) have been consistently applied to generate evolutionary manipulation of musical material. In his work, Koza [12,15] defines genetic programming as a domain-independent problem-solving approach in which computer programs are evolved to solve, or approximately solve, problems. Garcia [16] uses GAs to solve the problem of automating the design of the sound synthesis through Genetic Programming and perceptual distance metrics in order to measure the distance between target and produced sounds. Biles [4] presented a genetic algorithm-based program that mimics a student learning to improvise jazz solos under the guidance of a human mentor. In Horowitz's [5] development, an interactive system uses GAs to develop a criterion for distinguishing rhythmic patterns producing a large number of variations. We have also studied applications of GAs to interactive composition [6]. Similar to the approaches described above, our previous research used MIDI data to control music events in real time. Yet, using a different heuristic, we created a system named Vox Populi [7], a hybrid system composed of an instrument and a compositional environment. Vox Populi produces sounds moving from clusters to sustained chords, from pointillist sequences to arpeggios, depending upon the number of chords in the population, the duration of the generation cycle, and drawings made by the user over a graphic interface (GUI) control pad.

Evolutionary Computation was used by Johnson [11] to develop a computer system for sound design. Roads [14] also used genetic algorithms in Granular Synthesis to facilitate the regulation of its parameters. It is also employed in Computer Graphics to create scenes and animations [9]. In these applications the rules are *learned* by the system through its interaction with the user as described by Fogel [8]. Since it is, in general, difficult to combine quantitative and qualitative descriptions of a given sound, we apply the concept of Evolutionary Systems to develop a methodology for Timbre Design. This was named as *ESSynth Method* and is the focal point of this paper. *ESSynth* uses a set of

Target waveforms to describe a timbre tendency generated from an initial population of waveforms. So, new variants (generations of waveforms) "similar" to those ones in the target arise and are played in real time. This similarity is measured by evaluations through a suitable *Fitness Function*. From an algorithmic point of view *ESSynth* can be seen as a man-machine process that uses an implicit set of rules for generating waveform variations.

The main goal of this research is to formulate robust mathematical and computational models for measuring waveform similarities and to define genetic operations such as *crossover* and *mutation* (see definitions below) to operate as waveform transforms. By controlling the *Target Population* as well as the *Initial Population*, it is possible to create organized sound patterns or, at a higher level, a musical composition. For the latter, an external device linked to the *ESSynth* such as a wavetable engine is necessary in order to playback the musical sequences in real time.

1. THE GENOTYPE AND THE PHENOTYPE OF THE WAVEFORMS

The representation of timbre cognition is one of the most interesting and complex issues in the context of the audio signal processing. It is a difficult task to produce a taxonomy to classify timbre using quantitative and/or qualitative features. Schaffer [1] in the "*Traité des objets musicaux*" introduced a distinction between *form* and *matter* in the context of *concret musique*. As described by Risset [2], Schaffer's point of view could be exemplified as relating the amplitude envelope to the *form* and the spectral content to the *matter*. Although elegant, nowadays this definition is not completely accepted by the academic community. We present a new methodology to work with timbre that uses as paradigm: the sound of a given waveform used as a static structure unity differs drastically from the sound of the same waveform transformed dynamically. Spectral changes in time carry important musical features, Smalley [3] stated that "*spectral typology cannot realistically be separated from time: spectra are perceived through time, and time is perceived as spectral motion*". Risset [2] presented an interesting concept of sound variants: "*by changing the parameters of the synthesis models, one can produce variants. Variants can be intriguing because they can be very close to the original sound in some ways and yet quite different in other ways"*.

Inspired by Risset's sound variants idea [2], it is possible to imagine variants as a kind of genetic transformation applied into a population of sound patterns. Smalley's [3] integration of time and spectra induces to think in a timbre evolution over time or, using another terminology, a dynamic process in which an *Evolutionary Timbre* is generated. In ESSynth, waveform populations are taken as *genotype sets* and the resultant transforming timbre is taken as *phenotype*. In this sense the *genotype* is changed (i.e. the waveforms in the population), but the *phenotype* is preserved (i.e. the overall timbre) producing a variant. In the way Risset interpreted Schaffer's concept we can interpret waveform (i.e. genotype) as "form" and timbre (i.e. phenotype) as "matter" in the case of ESSynth. Therefore, these two elements are used in the same way evolution uses genetic information to generate new individuals of the same specie.

2. THE EVOLUTIONARY COMPUTATION OF WAVEFORMS

The ESSynth method is a man-machine interaction cycle. Firstly, the user specifies a set of *Target Waveforms*. Secondly, the program produces generations of waveforms using the target set the and a suitable fitness criterion. The user is free to change the *Target set* any time. When this happens, a new population generation starts, and so forth. There are four basic structures of control:

1) $\mathbf{B}^{(n)}$, the n-th waveform population generation.
2) $\mathbf{B}^{(0)}$, the waveforms *Initial set*.
3) **T**, the waveforms Target set.
4) f, the *Fitness Function* used to evaluate the best waveform **w*** of each generation.

We denote $\mathbf{w}^{(*,n)}$ as the most similar or, in mathematical terms, the closest waveform of $\mathbf{B}^{(n)}$ to the target set **T**. In each generation, say the n-th, its best waveform $\mathbf{w}^{(*,n)}$ is sent to a buffer and played as a wavetable cyclically (see Fig 1.).

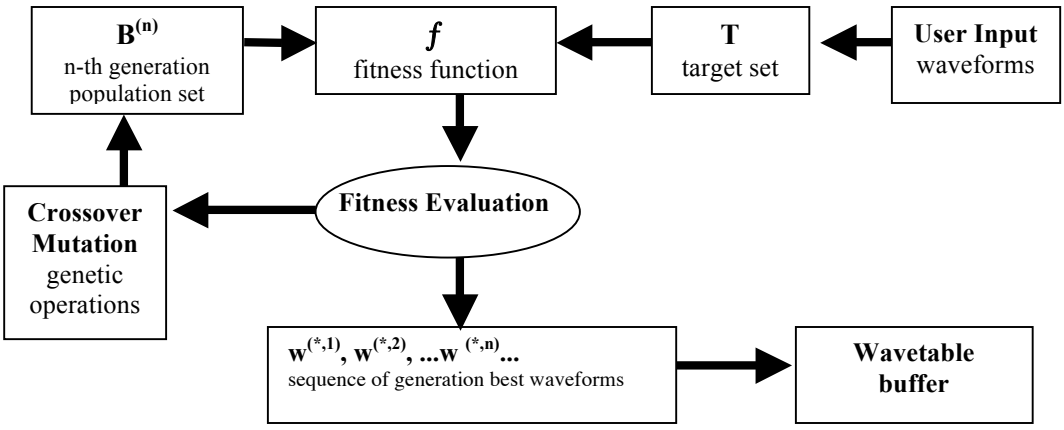


Figure 1. *Basic control structures of the ESSynth.* $\mathbf{w}^{(*,n)}$ *denotes the best waveform of the n-th generation sent to the wavetable buffer.*

All waveforms are normalized as floating point arrays with 1024 values defined in the real interval [-1,1]. **T** is defined by the user and $\mathbf{B}^{(0)}$ can be given by the user or it is generated randomly by the program.

3. THE EVALUATION OF THE FITTEST

It is firstly defined an suitable (and simple) metric or distance function. Our mathematical model considers waveforms as vectors in a real vector space $W = \Re^{1024}$ i.e. each vector in the space has 1024 components.

3.1 DISTANCE BETWEEN VECTORS

Given two vectors **v** and **w** in **W**, we define the usual Euclidian Metric between them:

$$\mathbf{d_2}(\mathbf{w}, \mathbf{v}) = (\Sigma_{i=1,...,1024} (w_i - v_i)^2)^{1/2} . \quad \textbf{(1)}$$

This metric induces the norm $|\mathbf{w}| = (\Sigma_{i=1...1024} (w_i)^2)^{1/2}$ which can be interpreted, from the acoustic point of view, as the *Total Energy* of the resultant sound. However, other metrics could be used and tested.

3.2 DISTANCE BETWEEN SETS

Now, we define a distance function between two sets. So, let $\mathbf{T} = \{\mathbf{t}^{(1)}, \mathbf{t}^{(2)}, ..., \mathbf{t}^{(L)}\}$ to be the *Target Waveform Set* and $\mathbf{B}^{(n)} = \{\mathbf{w}^{(n,1)}, \mathbf{w}^{(n,2)}, ..., \mathbf{w}^{(n,M)}\}$ the n-th waveform generation set. Since these are sub-sets of **W**, we can define the distance between them as follows:

$$\mathbf{d}(T, B^{(n)}) = \min \{\mathbf{d_2}(\mathbf{t}^{(j)}, \mathbf{w}^{(k)})\} \quad \textbf{(2)}$$

with $\mathbf{j = 1,..,L}$ and $\mathbf{k = 1,..,M}$, and **L** is the number of waveforms in **T** and **M** is the number of waveforms in $\mathbf{B}^{(n)}$.

As pointed above, **T** and $\mathbf{B}^{(n)}$ are finite sets, therefore the minimum in Eq. **(2)** is obtained for at least one vector in $\mathbf{B}^{(n)}$, which we denote by $\mathbf{w}^{(n,*)}$. This vector is the n-th generation best waveform, obtained using the metric of Eq. **(2)**. Now we define the n-th generation Fitness Function $\boldsymbol{f}$: $\mathbf{T} \times \mathbf{B}^{(n)} \rightarrow \mathbf{B}^{(n)}$ as

$$\boldsymbol{f}(\mathbf{T}, \mathbf{B}^{(n)}) = \mathbf{w}^{(n,*)} \quad \textbf{(3)}$$

which indicates the best individual of n-th generation population.

In a process of genetic improvement the best individuals with better-adapted phenotype, survive for the next generation. In Nature adverse conditions select these individuals. In our model this is accomplished by a distance function that measures how far a new waveform population departs from a Target Set fixed by the user.

4. THE EVOLUTIONARY SOUND SYNTHESIS METHOD

Waveform variants are produced by applying genetic operations such as crossover and mutation to $\mathbf{B}^{(n)}$. An interesting ESSynth feature is to collect these waveform patterns in order to make dynamical sequence in real time, which is heard as a sound environment.

Another importante aspect is that biologic evolution produces species diversity: with ESSynth one can create and manipulate a complex generation of sound material. Crossover increases the waveform co-variance and mutation produces random population variations. These two genetic operations, which are the generators of ESSynth diversity, are defined below.

4.1 GENETIC OPERATION: CROSSOVER

In order to define Crossover Operation we need a auxiliar vector which we named *Crossover Vector*.

It is simply defined as a vector with M elements, that is $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots \alpha_M]$, where $0 \leq \alpha_i \leq 1$ are chosen by the user. Now it is possible to define a kind of continuous waveform crossover as follows.

The n-th generation best waveform is used as a *Parent Waveform* $\mathbf{w}^{(n,*)} = (s_1, s_2, s_3, \ldots s_{1024})$. Any other $\mathbf{B}^{(n)}$ waveform is denoted by $\mathbf{w}^{(n,i)}$ with $\mathbf{0 \leq i \leq M.}$

The following steps define the n-th generation Crossover Operation:

Step 1: Set a random integer number generator in the interval [1,1024].

Step 2: Take two integer numbers $(k_1^{(n)}, k_2^{(n)})$ in the interval [1,1024] with $k_1^{(n)} < k_2^{(n)}$.

Step 3: Select the waveform segment in $\mathbf{w}^{(n,*)}$ as $\mathbf{S}^{(n,*)} = (s_{k1,\ldots,}s_{k2})$.

Step 4: Combine the waveform segment with a equivalent waveform segment $\mathbf{S_i}^{(n,i)}$ in $\mathbf{B}^{(n)}$ applying a Hamming Window $\mathbf{H(\ )}$ on $\mathbf{S_i}^{(n,*)}$ and a Convex Combination as:

$$\mathbf{S}^{(n+1,i)} = \alpha_i \,.\mathbf{H}(\mathbf{S_i}^{(n,*)}) + (1- \alpha_i \ \mathbf{S}^{(n,i)}), \qquad \textbf{(4)}$$

Notice, we denote the new segment as $\mathbf{S}^{(n+1,i)} = (s'_{k1},..,s'_{k2})$.

Step 5: The crossover operation is the replacement of each $\mathbf{S}^{(n+1,i)}$ in the original waveform making $\mathbf{w}^{(n+1,i)} = (s_1, s_2,..,s'_{k1}..s'_{k2},..s_{1024})$.

Step 6: Repeat steps (4) and (5) for all waveforms $\mathbf{w}^{(n,i)}$ in $\mathbf{B}^{(n)}$ with $\mathbf{w}^{(n,i)} \neq \mathbf{w}^{(n,*)}$.

As said above in step (4), we are applying a convex combination and using a Hamming window to smooth the border of the waveform segment $\mathbf{S}^{(n,i)}$. Intuitively, this is equivalent to merge waveforms using the slider of a sound mixer in an electronic studio.

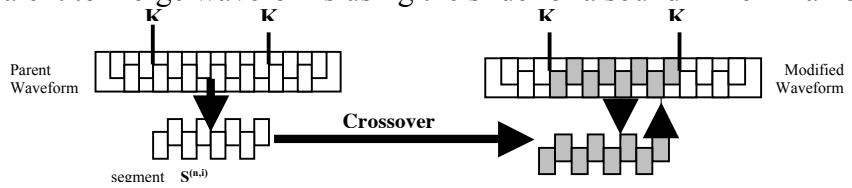


Figure 2. *Diagram of the crossover operation in which a segment extracted from the parent waveform and defined by* $\boldsymbol{k_1}$ *and* $\boldsymbol{k_2}$ *is mixed with an equivalent segment of a waveform from* $\mathbf{B}^{(n)}$.

Similarly to living organisms that reproduce through meiosis, our crossover operation mixes the *waveform codes*, which are $\mathbf{B^{(n)}}$ elements. In order to get better adapted individuals, they are crossed with $\mathbf{w^{(n,*)}}$. This can be seen as a kind of waveforms *natural selection*. As the user can interfere any time, we better call it a *driven genetic selection*. For each generation the best waveforms are sent to a sound output (wavetable). This makes ESSynth an excellent tool for real time synthesis. The user can manipulate the *Target set* as well the *Initial Population* in order to get a musically significant sequence of waveform patterns.

4.2 GENETIC OPERATION: MUTATION

On living organism mutation makes strong modifications on population individuals, generally due to external factors. It can be understood as a kind of disturbance of the reproduction process. We used this characteristic to define the mutation below. It starts with a definition of a *Mutation Coefficient* $0 \leq b \leq 1$ that *sets* the amount of disturbance applied to $\mathbf{B^{(n)}}$. Since the waveforms belong to $\mathbf{W} = \Re^{1024}$, a *Mutation Vector* $\boldsymbol{\beta}$ is generated with 1024 entries randomly generated in the interval [1-b, 1], named as *disturbance interval*. Now the *Mutation Operation* is defined on the *n-th* generation by the following steps:

Step 1: Create the *Mutation Vector* $\boldsymbol{\beta} = [\beta_1, \beta_2, \beta_3 ... \beta_{1024}]$, where each β_j belongs to the disturbance interval [1-b, 1].

Step 2: Apply the disturbance $\mathbf{w^{(j,\,n+1)}} = \mathbf{w^{(j,n)}} . \boldsymbol{\beta}$ on all elements of $\mathbf{B^{(n)}} = \{\mathbf{w^{(1,n)}}, \mathbf{w^{(2,n)}}, ..., \mathbf{w^{(M,n)}}\}$.

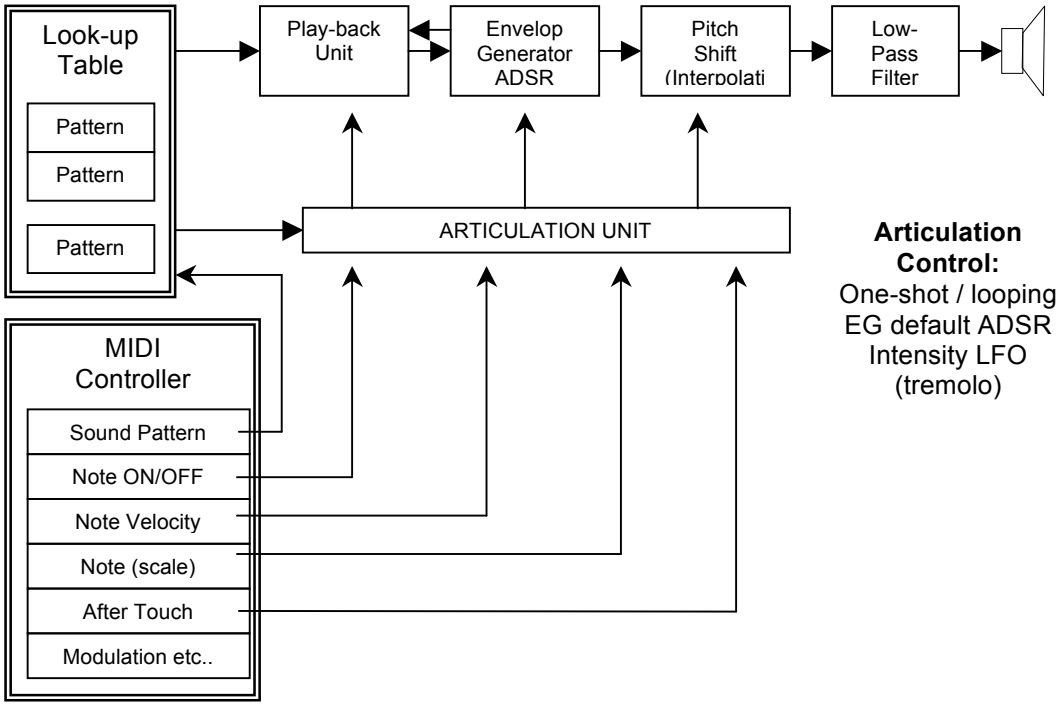
Step 3: Repeat steps (1) and (2) in every generation.

The mutation strength is controlled by parameter b in the real interval [0,1]. The closer is b to 0 the weaker is the mutation. As b gets close to 1 the mutation gets stronger, that is, random disturbance are produced in all [0,1] interval. Notice that ESSynth Mutation Operation the can be seen as a wave-shaping process where the waveforms are modified by a random Mutation Vector in an amount given by the Mutation Coefficient.

5. THE WAVETABLE SYNTHESIZER ENGINE

For the playback of waveforms generated by the evolutionary computation method it is important to have a method well-suited to handle with sound sequences. The wavetable synthesis provides a simple yet flexible way to control the waveforms in real-time. It is based on a table of sounds, or a "wavetable", that stores the sound patterns, or waveforms. The wavetable synthesizer is accessed and managed by a controller, usually a MIDI control, such as a MIDI keyboard or a MIDI sequencer, that sends instructions to the wavetable to play the sound pattern. Because of being computationally inexpensive, able to operate in real-time and handling predefined waveforms, the wavetable

synthesizer method is choosen to work with the ESSynth. A basic wavetable synthesizer is show in the figure below:



There are two major sections in the wavetable synthesizer: the look-up table and the engine. The look-up table is the heart of the synthesizer that contains the population of the waveforms. The engines takes care of the access and control of the waveforms within the look-up table throughout articulation instruction coming from the MIDI controller.

The major advantage of the wavetable is the way the information of each sound is organized and stored. This allows to save a great deal of memory while making the sound very realistic and easy to control. We aim to implement and present in further works the whole system compounded of the ESSynth generating waveforms for the lookup table of a wavetable synthesizer engine controlled by a MIDI device. For now we present the results of the ESSynth waveform manipulation of a population of waveforms that is going to be in further researche of the lookup table of a wavetable synthesizer.

6. GRAPHICAL EXAMPLES OF WAVEFORMS GENERATION

In this section we analyze graphic examples of sound results. These were obtained using Matlab 5.2 to simulate the ESSynth method. A Target Waveform Population was used with two diverse Initial Populations, namely:

a) Random distributed sine waveforms

b) Harmonic distributed sine waveforms,

as shown in **Table 1**.

All waveforms have 1024 values normalized in the real interval [-1,1]. It is worth to mention that it is possible to use complex waveform patterns as Target and Initial Populations. Our strategy is to show ESSynth behavior with simple patterns.

Table 1. E*xperimental parameters used to simulate ESSynth.*

ESSynth Examples			
Index	**Population Description**	**Waveforms in the population**	**Frequency Distribution (Hz)**
01	Target Waveform	05	100, 200, 300, 400, 500
02	Random Distributed Sine Waveforms	10	From 180 to 16,000
03	Harmonic Distributed Sine Waveforms	10	100, 200, 300, 400, 500… 10,000

shown in Fig.3 and Fig.4, the method converges making erosions in the waveform population produced by the Crossover Operation. Thus, the sound envelope produced by ESSynth starts with strong amplitude followed by a smooth decay, as instrumental sounds behave. This sound is also dynamically changed in its spectral contents due to the Mutation Operations.

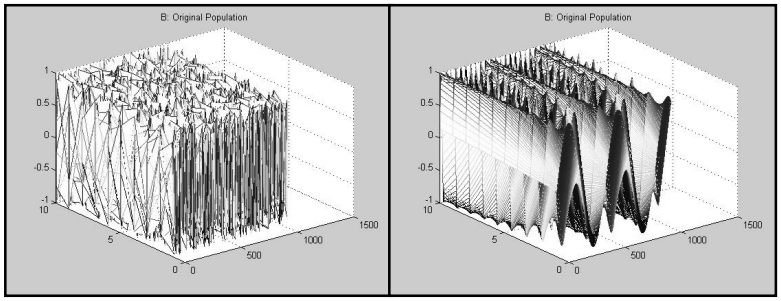


Figure 3. *3D plot of the populations used in the examples. The plot on the left side is the Random Distributed Sine Waveforms population. The plot in the right side is the Harmonic Distributed Sine Waveforms population. The X-axis displays the 1024 values waveform. The Y-axis displays the waveform pattern in an increasing order and the Z-axis is displaying the samples amplitude in the interval [-1, 1].*

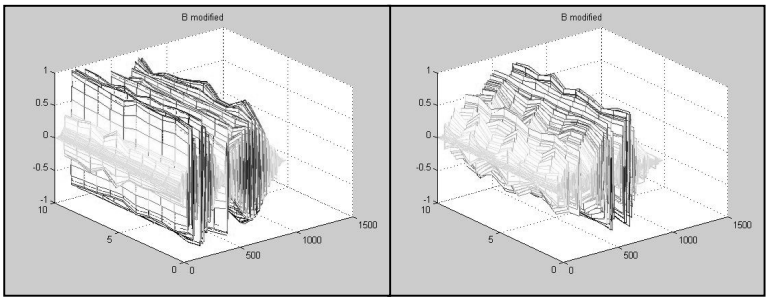


Figure 4. *3D plot of the waveform* p*opulations after 500 interactions. It is apparent how the ESSynth works making erosions in the waveform population as the Crossover Operation dumps waveforms.*

ESSynth produces a driven disturbance that reflects a sound tendency controlled by the user input Target Set. During the Matlab simulations we realized that the crossover operation produces a waveform sequence that converges to the Target Set, which can be viewed as a waveform attractor set. This numerical behavior produced sounds that become more similar one to another along the time evolution and, of course, to the waveforms in the Target Set. On the other hand, despite the mutation operation that can add drastic changes on the waveform trajectory, it did not affect the overall sequence which still converge to the Target Set. Finally, it is always possible to modify the Target Set Population and/or the Initial Population leading to a very strong departing from the actual waveform pattern as shown in the graphic examples above.

7. CONCLUSION AND FINAL COMMENTS

This paper has described a new methodology for sound synthesis and a mathematical model to drive waveform generation towards a chosen fixed Target Set. ESSynth is a sound synthesis method that integrates the Mathematical Approximation Theory to the Genetic Algorithms. It can be seen also as a new framework for Timbre Design. The basic features of the method are: a) the use of a Target Set of waveform patterns, b) a Fitness Criterion implemented by a distance function (we used Euclidian Distance) which measures how much a waveform population departs from the Target Set and c) an evolutionary process, based on crossover and mutation operations, applied to waveform sets to produce a sequence of waveform generations which converge to the Target Set according the Fitness criterion and the distance function.

In the Matlab simulation, we have used a random number generator to set the *Crossover Vector* and the *Mutation Coefficient*. It is possible to develop a general model in which the components of the parametrical controls have some aditional pre-defined cross-correlation. Of course, best sonic results will be also dependent on the waveforms in the Target set: the user is free to experiment with several waveform populations during the generation process. The wavetable synthesis could work quite well with ESSynth since the wavetable patterns, generated by this evolutionary process, will have interesting dynamical proprieties. Indeed, the wavetable synthesizer engine will provide an easy and fast environment to manipulate and play waveforms generated by the ESSynth in real time.

REFERENCES:

[1] Schaeffer, P. "Traité des objets musicaux".: *Editions du Seuil*. Paris 1966.

[2] Risset, J.-C. 1991. "Timbre Analysis by Synthesis: Representations, Imitations, and Variants for Musical Compostion". *In Representations of Musical Signals*, ed. De Poli, Piccialli & Road, Cambridge, Massachussetts: The MIT Press, ISBN 0-262-04113-8, pg. 7-43.

[3] Smalley, D. 1990. "Spectro-morphology and Structuring Processes". *In The Language of Electroacoustic Music*, ed. Emmerson, pg. 61-93.

[4] Biles J. A.. GenJam: "A Genetic Algorithm for Generating Jazz Solos" *Proceedings of the 1994 International Computer Music Conference*, (ICMC'94), 131—137, 1994.

[5] Horowitz D., "Generating rhythms with genetic algorithms". *Proceedings of the 1994 International Computer Music Conference*, 142 – 143, 1994.

[6] Manzolli, J., A. Moroni, F. Von Zuben & R. Gudwin. 1999. "An Evolutionary Approach Applied to Algorithmic Composition". *Proceedings of the VI Brazilian Symposium on Computer Music*, Rio de Janeiro, p. 201-210.

[7] Moroni, A., Manzolli, J., Von Zuben, F. & Gudwin, R., 2000, "Vox Populi: An Interactive Evolutionary System for Algorithmic Music Composition". *San Francisco, USA: Leonardo Music Journal* - MIT Press, Vol. 10, p. 49-55.

[8] Fogel D. B.. "Evolutionary Computation - Toward a New Philosophy of Machine Intelligence*". IEEE Press*, USA, 46 – 47, 1995.

[9] Fogel, James D. Andries van Dam, Steven K. Feirner and John F. Hughes. "Computer Graphics Principles and Practice"*, Addison-Wesley Publishing Company*, p. 1018, 1996.

[10] Polansky, L. 1996. "Morphological Metrics". *Journal of New Music Research*, Vol. 25, pp. 289-368.

[11] Johnson C. G.. "Exploring the sound-space of synthesis algorithms using interactive genetic algorithms in G. A." *Wiggins, editor, Proceedings of the AISB Workshop on Articial Intelligence and Musical Creativity*, Edinburgh, 1999.

[12] Koza, J. R., Bennett III, F. H., Andre, D., Keane, M. A., Dunlap, F., "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming," *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 2, July 1997.

[13] Cheung, N. M., Horner, A., "Group Synthesis with Genetic Algorithms," *Journal of the Audio*
Engineering Society, 44(3): 130 –147, 1996

[14] Roads C. "Genetic Algorithms as a Method for Granular Synthesis Regulation". *Proceedings of the 1993 International Computer Music Conference*, 1994.

[15] Koza, John R.. "Genetic Programming". *Encyclopedia of Computer Science and Technology.* 1997.

[16] Garcia, Ricardo A. "Automatic Generation of Sound Synthesis Techniques". *Master of Science Dissertation, MIT* 2001.
See internet page http://ragomusic.com/content/view/18/42/.