TOWARDS THE AUTOMATIC GENERATION OF SOUND SYNTHESIS TECHNIQUES: PREPARATORY STEPS

Ricardo A. Garcia

Media Lab, Massachusetts Institute of Technology
20 Ames Street Room E15-401, Cambridge, MA 02139 USA
PH: 617-253-0112 FAX: 617-258-6264
e-mail: rago@media.mit.edu

Abstract - An overview of an algorithm that searches through the space of the sound synthesis techniques is presented. A modular approach to construct sound synthesis techniques is introduced. The preparatory steps needed to use genetic programming as a search tool for this space are explained, focusing in the manipulation and evaluation of the modular descriptions of the topologies.

I. INTRODUCTION

The field of automatic generation of synthesized sound has been focused usually in the estimation of parameters for a predetermined synthesis technique (or topology) to try to reproduce a particular target sound (1), (2). While this is very plausible, it is known that different sound synthesis techniques perform better with different types of sounds. Therefore, some parameter matching techniques can give poor results when using a fixed topology. A search algorithm that includes the “sound synthesis techniques space” as part of the search can be used to select from a limited set of synthesis techniques (i.e. the “classic” synthesis techniques) before applying the parameter-matching algorithm. An even more adventurous goal would be to develop an algorithm capable of search through a broader set of synthesis techniques (i.e. classic and more). This kind of algorithm would be even able of suggesting “new” (i.e. not classic) synthesis techniques (topologies).

This paper describes the overall idea of an algorithm capable of finding sound synthesis techniques, and focuses in the preparatory steps necessary to accomplish this goal. Specifically, it describes the elements used as construction blocks to form the synthesis techniques, their descriptions, and the use of a genetic program that searches the sound synthesis techniques space.

II. ABOUT SOUND SYNTHESIS TECHNIQUES

A sound synthesizer is a device (or method) used to produce sound artificially (that is, without the use of acoustic instruments). This paper will discuss about the digital sound synthesizers (as the ones implemented in a digital computer), their algorithms and the digital sound samples that they produce. A sound synthesis technique will be then the particular arrangement of elements (algorithm or topology) that are used to synthesize sound using a digital computer.

Within this framework, we can find several techniques that are commonly applied by digital sound synthesizer to produce sound (3), (4), (5). Some of these techniques include additive, FM, subtractive synthesis, ring modulation, etc.

III. OVERALL IDEA

The ultimate goal is to have some automatic way to design sound synthesis techniques. Having just a sample of the desired sound (i.e. a couple of seconds) and perhaps some "control" parameters. If by example it is desired to find a synthesis technique well suited to synthesize a flute, it is possible to record the sound of the instrument (target sound) and some performance parameters (i.e. breath information, pressure on the keys, etc).

The hypothesis is that any synthesizer can be thought as a black box that accepts the performance parameters as inputs, and does some processing to them to produce the sound (output).

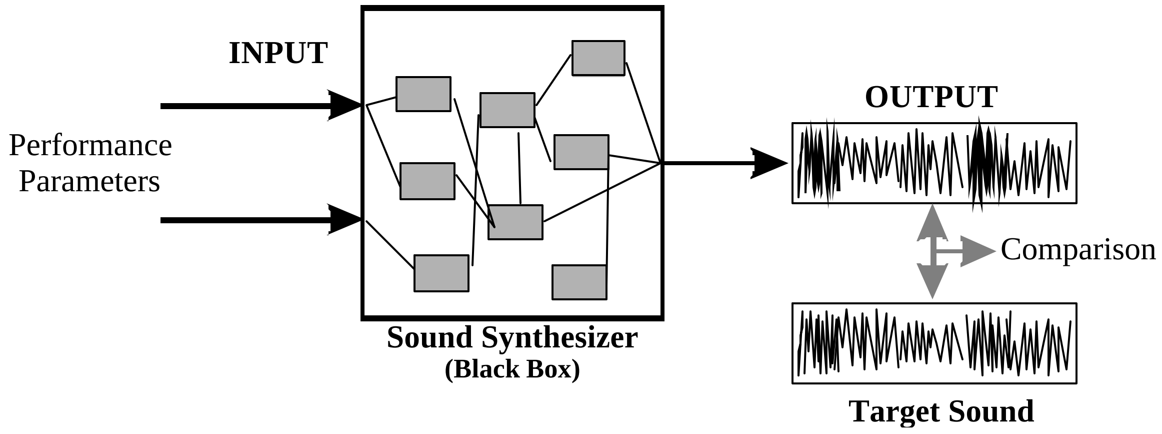

Fig. 1 Sound synthesizer as black box with simple construction Blocks

The black box inside will be composed by simple elements (construction blocks) that are arranged in a particular way (topology). They respond to the input signals to create the output sound. The goal is to find one arrangement of construction blocks that will produce an output that is "similar" to the target sound.

Because of the modular nature of the construction blocks, the possible number of combinations is virtually unlimited! Even after including some restrictions (i.e. restrict the number of construction blocks allowed, number of connections, type of blocks, etc) the possible number of combinations is too big. It is not practical to test all the possible block combinations for a good match, but it is possible to use a search algorithm well suited for high dimensional searches. A good searching technique is "genetic programming" (6). It uses principles derived from the theory of Darwinian Evolution to grow a population of "solutions" (in this case, sound synthesizers), measure their performance (how good are they, how close are they to the target sound) and create new populations of individuals using genetic operations. This process is repeated until a good solution is evolved.

The solution will be a block configuration or topology that will satisfy some kind of "similarity" criteria between the output produced by the grown synthesizer and the target sound. This similarity can be an objective error measure (i.e. Mean Square Error) or a subjective measure (i.e. perceived quality).

The strength of this approach resides in the modular nature of the construction blocks. They can be arranged in a multitude of ways. Some of these arrangements have correspondence with the "classic" synthesis techniques (i.e. additive, FM synthesis, etc). But also, many of these arrangements do not have a correspondence with any popular sound synthesis technique. A point in the middle is also possible, where the arrangement of blocks combines elements from different synthesis techniques.

IV. BASIC CONSTRUCTION BLOCKS

When analyzing some of the standard sound synthesis techniques it is expected to find some common elements that are present in all of them. It is known in advance that these elements are going to be used as construction blocks to assemble different synthesis techniques. Some desired features drive the selection of these blocks:

- Few types of blocks
- Highly functional (high level operations)
- Avoid over specification (do not make them too particular)

After analyzing the common elements in several of the "classic" sound synthesis techniques, the basic construction blocks were chosen (7). All the parameters and dynamic range of the connections are normalized [-1,1] to allow interconnection.

FTLO (Filtered Table Lookup Oscillator)

This is the most complex of all the selected construction blocks. It is composed by:

- wavetable: a single cycle wavetable with selectable parameters:
- Frequency: playback frequency (externally selected)
- Phase: Initial phase of the waveform
- Type: to select the "type" of the waveform to be reproduced. The waveforms are simple signals and include: sinusoidal, triangular, sawthoot, square, etc.
- Interpolation: To interpolate samples at the output of the wavetable.
- Filter: a two poles, two zeros digital filter.
- Amplification (gain): (externally selected)

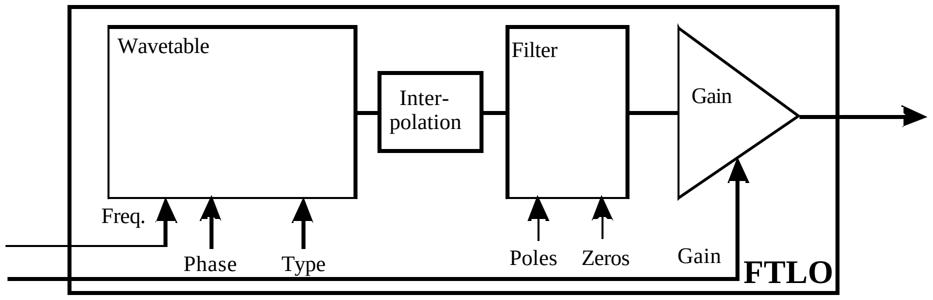

Fig. 2 Constitution of a FTLO block.

ADD (Addition)

Used to add two signals. The output is clipped if less than -1 or greater than 1.

MULT (Multiplication)

Used to multiply two signals. The output is clipped if less than -1 or greater than 1.

SPLIT (Split)

Used to split the incoming signal in two identical copies.

These blocks can be used to represent any of the classical synthesis techniques, as seen in figures 3 and 4.

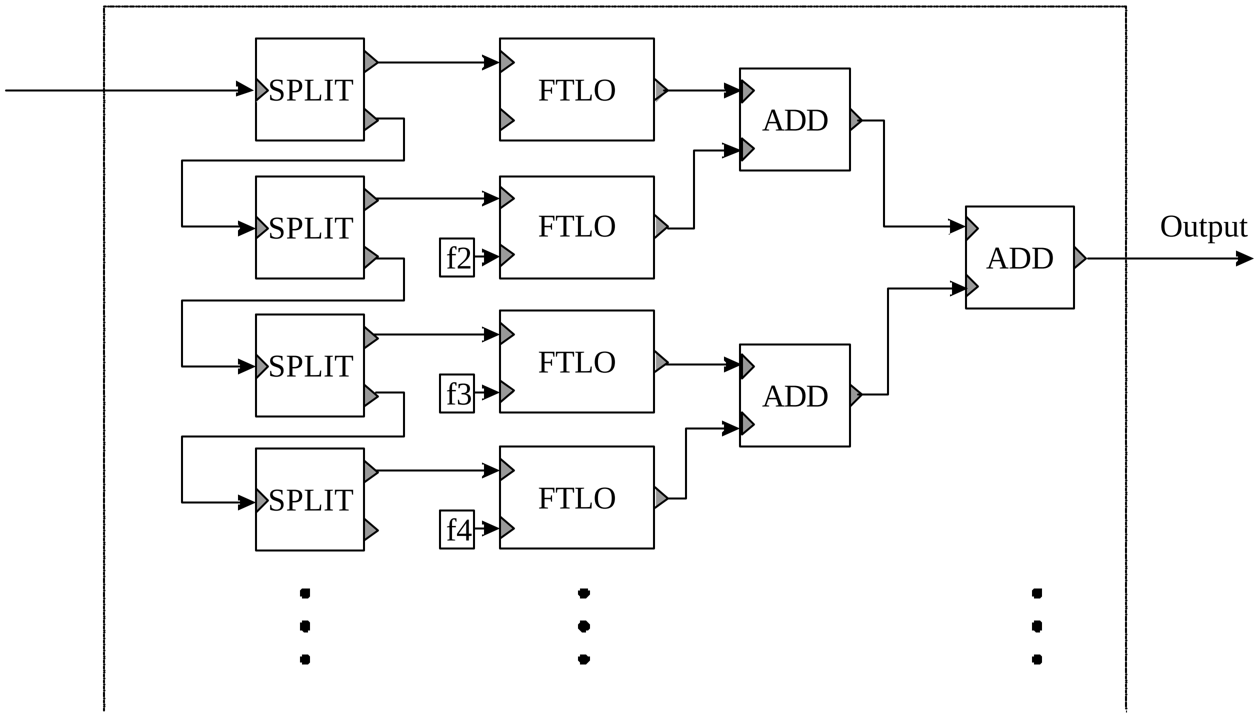

Fig. 3 Example of Additive Synthesis topology

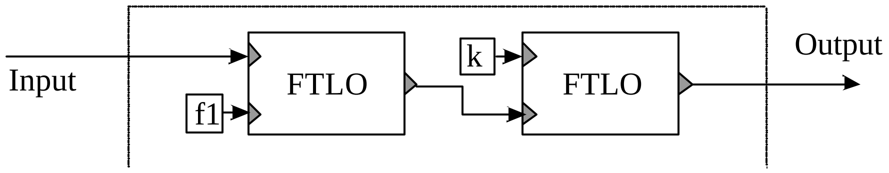

Fig.4 Example of FM Synthesis topology

V. GENETIC PROGRAMMING

Idea of Genetic Programming (GP)

Genetic Programming is an extension of the Genetic Algorithms (6), were computer programs compose the population. In this specific case, the computer programs will be algorithms for sound synthesis.

The GP searching technique is based on ideas borrowed from the theory of natural evolution and survival of the fittest (6), (8). The GP works with "individuals" and "generations". Usually, the number of individuals is kept constant between generations. The individuals are meant to be some kind of computer program (in this case, they are sound synthesizers), and the final goal is to find a computer program (sound synthesizer) that performs as specified by the requirements of the GP (in this case, produces an acceptable sound output).

In the first generation, a random population of individuals is created (random synthesizers, composed by random construction blocks and connections). Then, each individual is evaluated and tested for performance. The result is usually a numerical value that is called "fitness" (following the lexicon of evolution). This fitness value is a measure of how good the computer program (synthesizer) performs compared to the established goal (target sound). After all the individuals have been assigned a fitness value, the algorithm checks to see if a good solution has been reached. If this is the case, the "best" individual is chosen from the generation and the algorithm ends. If the desired solution has not been reach, the algorithm then creates a "new" population, using the individuals in the past population.

The individuals from the past population are randomly chosen to perform "genetic operations" on them and after that, the resulting individuals are put on the new population. The probability of an individual to be chosen to participate in a genetic operation is related to its fitness value. An individual with a better fitness value will have more probability to be chosen. The most widely used genetic operations are copy, mutation, and crossover. Copy simply copies the selected individual to the population of the next generation. Mutation, changes randomly some parameters or aspects of the selected individual, and introduces this mutated individual in the next population. Crossover selects two (or more) individuals, and then randomly copies part of their internal information and structure to generate two (or more) different individuals with traits from all the parents.

This process is repeated until a good solution (an individual with fitness equal or better than expected) is found, or a maximum of generations has been tested.

Key elements in GP

To apply Genetic Programming to the task of the automatic generation of sound synthesis techniques, several points have to be decided beforehand.

GP uses computer programs as "individuals". For this research, instead of using a standard sound synthesis computer language, a custom object oriented "language" was developed. This was because of the idea that GP works better when very few instructions with very high functionality are used. Also, because of the random and automatic nature of the GP, it is possible that very ill topologies are grown and all of them have to be tested. This custom approach permits the testing of a very high number of topologies (even thought that it might be not efficient).

The individuals will be "transformed" between generations using "genetic operations". Because of the cyclical nature (loops) of the sound synthesis topologies, a second level of abstraction is needed. The GP does not work directly with the topologies, but with a "description" of them. These descriptions can be formed to be of the non-cyclical graph type, and the genetic operations are performed on them. Each description will have a unique interpretation to "grow" a sound synthesizer.

The fitness score should measure the level of "closeness" of the sound synthesizer to the target sound. Two approaches can be taken: an objective quantitative measure like the Mean Squared Error, or a subjective qualitative measure like the Perceptual Similarity. In any case, the description of the synthesizer being evaluated has to be first "compiled" (interpreted) in a working topology and then be run to produce sound samples.

The genetic operations have to transform the description of the topologies. The most difficult to implement genetic operation is the crossover, because of the cyclical nature of the topology, where some connections will not exist if just part of the topology is used.

In this paper about the preparatory steps, some of these issues are discussed, and the rest are left open as part of the continuing research in this topic by the author.

Description of topologies

The topologies of the sound synthesis techniques have cyclical nature, where it is possible (and often desirable) to have closed loops within the circuit. This presents a big problem when the genetic operations have to be performed. It is required to have some kind of non-cyclic description of the topology. The advantage of using a non-cyclical description of the topology resides in the fact that some advances have been done in genetic programs that use this kind of descriptions of circuit topologies (9) that can be used as guidelines for this research. Most of them are involved in the use of genetic operations over non-cyclic graphs.

A good approach to find a description that captures uniquely all the characteristics present in a topology is to make a mapping between the components and connections in both types of graphs. Each description should "grow" or generate a unique synthesizer topology, but it is possible to have multiple descriptions that will generate at the end the same topology. In any case, the GP will evolve multiple descriptions, but the important part is the final result that is always measured by the sound produced by the synthesizer, so the description plays an intermediate role.

Each element (FTLO, ADD, SPLIT, MULT) is mapped with a special symbol into the description. The direction of the connections is very important. Note that the convention followed was to have a single connection at the top, differentiating the direction (in/down or out/up). This top connection will also help to determine the name or type of the element in the description. Figure 5 shows a list of all the elements used by the description.

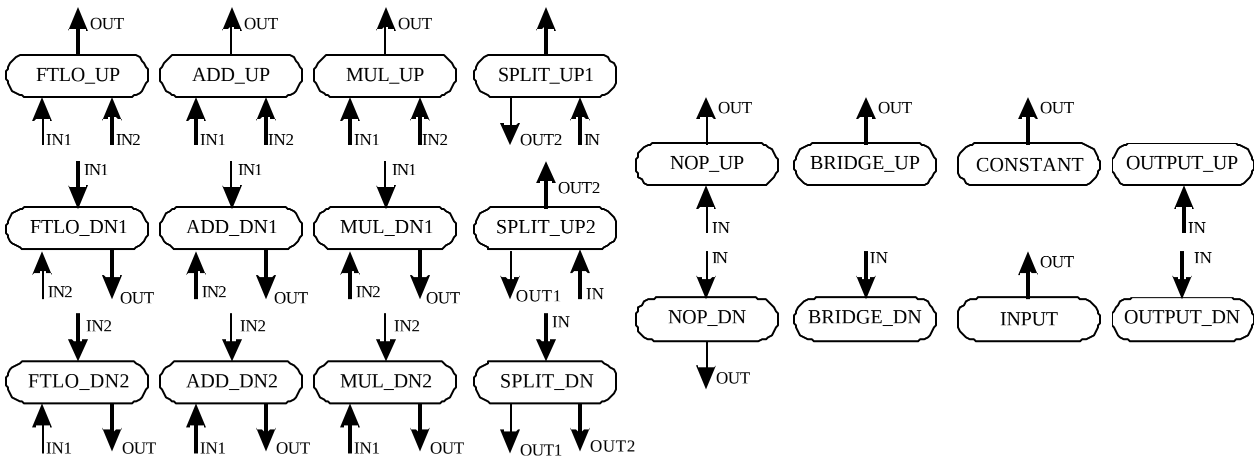

Fig. 5 All the “elements” used by the non-cyclical description

In addition to the elements that will represent the basic construction blocks, some new elements are introduced to help the creation of the topology from the description.

NOP will delay the execution of this branch for one step. If this element is found, the actual branch will stop evolving until the next step.

The pair of **BRIDGE_UP / BRIDGE_DN** make a connection between these two points in the topology. They follow the left to right, top to down convention. Each element will connect with the next one available of the complementary type.

The description of topologies is very much like a “recipe” that should be followed to grow a sound synthesizer (the actual topology or circuit). Each description when compiled (interpreted) will generate an unique sound synthesizer; but each topology could be represented by many possible descriptions. All of them use different combinations of the description elements. The convention adopted to interpret a description is from top to bottom, left to right. Just one element of one branch is executed at a given time.

Running the topology

The GP deals with "descriptions" of topologies, but when these descriptions have to be tested, it is necessary to compile (interpret) then in a working sound synthesizer and produce an output sound.

One of the important points to consider is that the grown synthesizers can be in virtually "any" possible configuration. This means that no restrictions are applied to a synthesizer (or its description). In example, a common restriction when designing a topology based on construction blocks is the "ordering" of evaluation of the blocks. Another restriction is in the type of loops (closed paths) that are allowed. Because the grown synthesizer can be crowded with these kind of loops, the evaluation scheme should be very strong, and allow evaluation of as many types of individuals in the population. This will ensure that no possible solutions are thrown away because of its trouble to be evaluated. The custom synthesis language was developed with the required characteristics for the evaluation. It uses an object oriented and message driven approach. The synthesizer produces one sample at each time step, and the process is driven by messages that are passed by each object when they have a "new" sample ready to be sent. In example, following the input samples in an ADD block. If a new sample gets in IN1, no processing is done. When a new sample gets in IN2 the process is done because "all" of the inputs have new, unprocessed samples. After the process, the output will post a message that a new sample is ready to be delivered.

The message loop will be running during the whole rendering of the sound, and it is in charge of receiving messages, and delivering them in First Input First Output infinite loop. This approach ensures that any topology that has no "dangling" connections will be able to be evaluated and an output sound will be rendered.

Grade the performance (fitness evaluation) (future)

Each individual have a fitness score that should be related to the "closeness" of its output sound to the target sound. The choice of the method to assign this fitness score is very important to the GP, because it will drive the search through the "sound synthesis techniques space".

Two main types of evaluation can be done:

Objective: This will use some kind of error measure between the output and target sounds. It can be in the time or frequency domains. The GP will produce a sound that is analitically close to the target, but it may be perceived very different.

Subjective: This will use some kind of perceptual distance measure between the sounds. This approach would be more preferable, but it will be more expensive in terms of computation.

Genetic Operations (future)

A fundamental part of a GP is the definition of the genetic operations. They are in charge of altering the description of the individuals to generate new individuals for the next generation.

Because of the nature of the GP, the individuals don't have a pre-determined length or structure. This becomes a problem when a mutation or crossover operation wants to be performed. Some techniques have been suggested and applied by Koza (6), (9) to the design of analog electric circuits.

VI. AKNOWLEDGMENTS

The author wishes to thank his colleagues of the Machine Listening Group (MIT Media Lab) for their support and ideas in this research. In special, to Paris Smaragdis for his suggestions about the object oriented approach that was taken for the evaluation of the construction blocks.

REFERENCES

(1) When, K., "Using Ideas from Natural Selection to Evolve Synthesized Sounds," Proc. Digital Audio Effects (DAFX), Barcelona, 1998

(2) Horner, A., Beauchamp, J., Haken, L., "Machine Tongues XVI: Genetic Algorithms and Their Application to FM Matching Synthesis," Computer Music Journal, 17:4, pp. 17-29, Winter 1993

(3) Moore, F. R., Elements of Computer Music, Prentice Hall, New Jersey, 1990.

(4) DePoli, G., "A Tutorial on Digital Sound Synthesis Techniques," Computer Music Journal, 7☹4) 8 - 26 1983.

(5) Boulanger, R., The Csound Book, The MIT press, Cambridge, MA 2000

(6) Koza, J. R., Bennett III, F. H., Andre, D., Keane, M. A., Genetic Programming III, Darwiniang Invention and Problem Solving, Morgan Kaufmann Publishers, San Francisco, 1999.

(7) Garcia, R., "Using Natural Modes to Decompose Sound Synthesis Techniques", Final class paper, MIT, 1999

(8) Gershenfeld, N., The Nature of Mathematical Modeling, Cambridge University Press, New York, 1999

(9) 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.

(10) Garcia, R., "Sound Synthesis using a Cellular Automata and Genetic Algorithms", Final class paper and project, MIT, 1999