MIT Student
May 10, 2010
21M.380 – C. Ariza

# GA Sonic System Report

Genetic Algorithms (GA's) have been applied since the 1950's to simulate the evolution of a population [1]. Soon enough, they became a widely known method to solve optimization problems in computer science [2] and several genetic algorithmic systems were developed, such as Evolver [3]. Genetic algorithms are not only suited for computer science; they may also be applied to artistic domains. For example, Ariza discusses an alternative application [4] of GA's. Rather than moving towards a complex solution, Ariza's genetic algorithm system pulls from a trajectory of rhythmic populations that moves towards a simple solution. My work with Genetic Algorithms also focuses on creating interesting trends from an evolutionary process that moves towards a simple solution. I show that my system can be applied to granular synthesis to make compelling gestures.

For my sonic system, I employ many of the principals described in Magnus's overview of genetic algorithms [5]. Solutions are mapped onto *chromosomes*, which contain many parameters encoded as *genes* (or *alleles*). Each chromosome can be rated with a *fitness function* by calculating the *error* from that chromosome to a target solution. Figure 1 shows a sample population with 3 chromosomes, each of

which has 3 alleles.

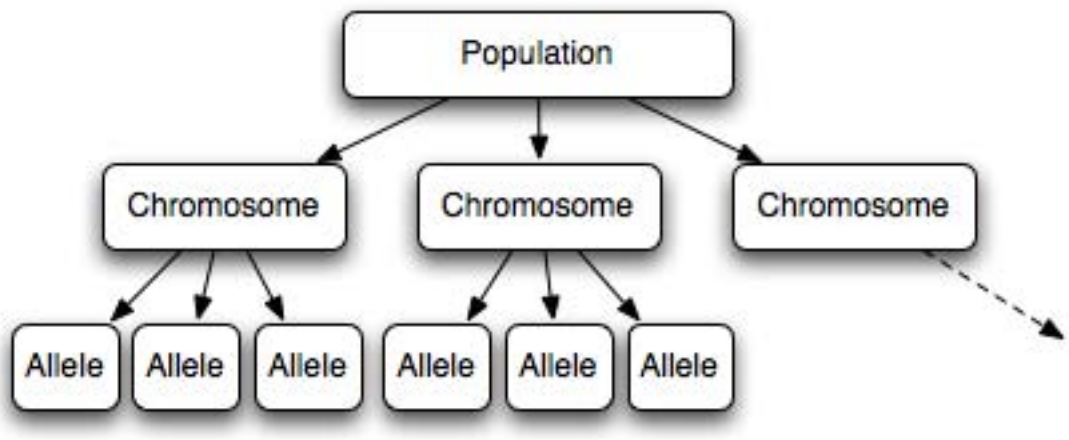


**Figure 1 Populations contains many chromosomes, each of which store multiple alleles.**

My GA system comes with several alleles and chromosomes. The *single note allele* contains a single midi value, and is used in *single note chromosomes* and *multi-note chromosomes* (chord chromosomes). The more interesting *grain allele* holds several important parameters for granular synthesis and is employed in the *grain chromosome*. My system also provides basic fitness functions that assign lower fitness to chromosomes that are farther from a *target* chromosome.

A simulation is run, in which some chromosomes *reproduce*, and others *die*. While traditionally fitness probabilistically determines which chromosomes reproduce, as described by Magnus, in my system every chromosome has an equal chance of reproducing. However, in my system less fit chromosomes have a higher chance of being replaced by an offspring. The outcome is similar: less fit chromosomes produce less offspring because they are replaced sooner.

During reproduction, offspring chromosomes have a probability that they will be *mutated* in several ways. *Crossovers* may occur between two parent

chromosomes, such that the offspring inherits some traits from one parent and some traits from the other. Individual alleles may also undergo *point mutation*, in which a single allele is mutated. For example, a note allele may be shifted up or down a few half steps. Mutations combined with reproduction and dying pushes the population in a direction towards a target.

At each step in the simulation, one or more chromosomes may be *extracted*. Extraction can be determined by fitness (*most fit chromosome* or *least fit chromosome*) or *random*. Extracted chromosomes are accumulated into an array. The simulation ends when the most fit chromosome fitness, or the average chromosome fitness reaches a certain threshold. Several simulations can be strung together, to produce multi-trend gestures. The simulation process is summarized in Figure 2. The corresponding code can be found in GA.py.

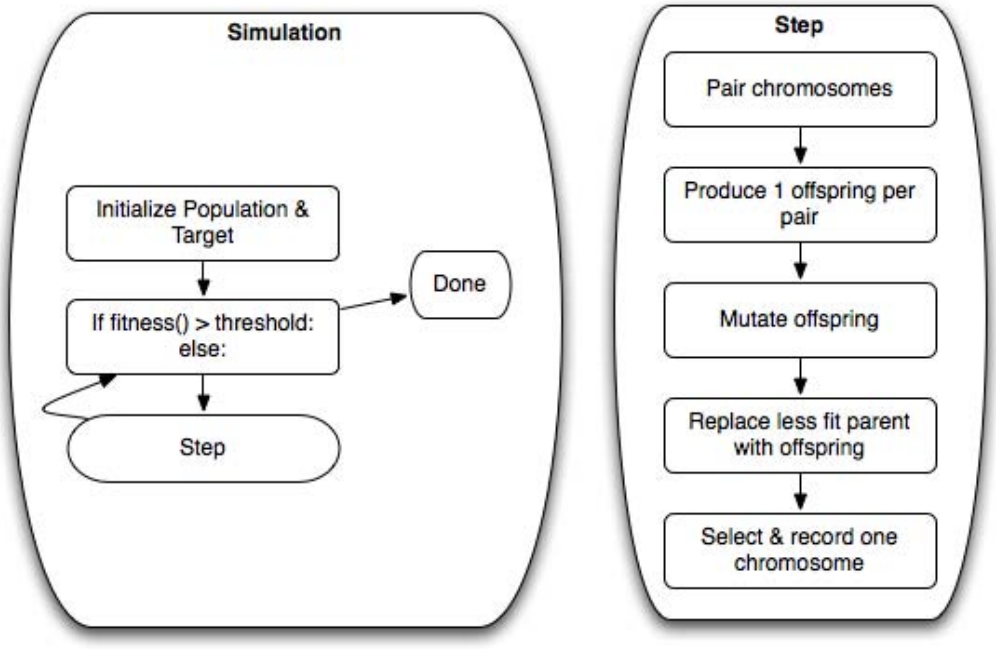


**Figure 2 the control logic of Simulation and Step.**

To demonstrate the capabilities of my GA system, I applied it to granular synthesis. In granular synthesis, small pieces of sound, or *grains*, are played in rapid succession to create larger events as described by Roads [6]. The interesting gestures of granular synthesis are made possible due to the collective trends applied to grains over time. The compelling trends that can be expressed with genetic algorithms are what motivated me apply genetic algorithms to granular synthesis.

I chose to vary grains according to three parameters that Roads identifies as important: *duration*, *frequency*, and *amplitude*. For this reason, each grain allele encodes three values, one per parameter. Each grain chromosome stores it's own grain allele. Grain chromosomes can also undergo crossover mutations, in which an offspring inherits one or more parameter from one parent and the rest from another parent. Grain chromosomes can also undergo point mutations, in which one or more parameters of the offspring's allele are increased or decreased. The fitness of a grain is measured with respect to the distance between itself and a target grain. A larger distance produces a lesser fitness.

I created three sound samples of genetic algorithms applied to granular synthesis to demonstrate different features of my system. In all three of my samples, output from the genetic algorithm was mapped to granular synthesis parameters in the csoundNative mode of athenaCL, using the LineGroove texture module to create SineUnitEnvelope grains (see Line_Groove.py). In sample1, random selection is used to generate a gesture where grains go from long duration, low volume, and low pitch, to short duration, high volume, and high pitch. This creates a linear transformation with a bit of up and down randomness. In sample2, I

show how the selection and mutation rate affect the gesture, by applying a best selection and a high mutation rate to the same population. As seen, the solution converges much faster (~150 steps as opposed to ~750 steps), and the fitness of the recorded samples strictly increases. In sample3, I use multiple-selection to select the best-fit chromosome and the least fit chromosomes, and play them in parallel. In this sample, grains go from high pitch, low volume, and short duration to low pitch, low volume, and long duration. Sample1, sample2, and sample3 showcase the features of my system and it's ability to be applied to specific domains, such as granular synthesis.

Others have applied genetic algorithms to granular synthesis, such as Fijinaga [7]. My system differs in several ways. First, where Fijinaga's GA deploys bit manipulation mutation, my system applies domain specific mutation. Second, my granular synthesis chromosome utilizes a fitness functions specific to its domain. For example, rather than making the fitness function be additive over the different granular parameters, I deploy a different fitness functions for each parameter and take the minimum of these functions to represent the fitness of a chromosome. The result is that all of the features converge at the same rate, which I feel is desirable for granular synthesis. Finally, my system is different because of its flexible selection strategy.

To summarize, my system can be formally described according to Ariza's seven descriptors [8]. My system produces semi-macro-scale gestures (not full pieces, but not just single events) by combining micro-scale sounds according to a non-real-time process model. While the granular synthesis extension is tied to a

single idiom, the system as a whole exhibits a plural idiom-affinity. The GA system is openly extendable – it is coded generally enough such that users can define their own functions and / or subclasses to alter or extend it. My genetic algorithm follows a generative event production model because parameters are generated and pulled from the population. It is conceivable that one could use it as a transformational tool by supplying key events from a source as targets for the GA's fitness function. My system does not directly produce sound – it must be mapped to an external sound source, such as AthenaCL as I did when generating granular synthesis samples. Finally, the user interacts with my system through the python scripting language. They can either supply arguments to one of many functions defined at the bottom of GA.py to run a simulation, or expose more parameters by writing functions of their own.

I have several ideas for further research that would build upon my sonic system. First, it would be interesting to experiment with a population that could grow or shrink over time. Varying-sized populations could provide an interesting application to granular synthesis in which the density of grains is proportional to the population size. Second, it would be informative to apply the GA to domains other than granular synthesis. It would also be interesting to gauge the ease of use with which an outside party could extend or use my system. Finally, it would be useful to support more chromosomes, such as a scale chromosome or a key chromosome and see how they could be applied to generating interesting gestures.

Sources


[1] Fraser, Alex (1957). "Simulation of genetic systems by automatic digital computers. I. Introduction". *Aust. J. Biol. Sci.* 10: 484–491.

[2] Schwefel, Hans-Paul (1981). *Numerical optimization of computer models (Translation of 1977 Numerische Optimierung von Computor-Modellen mittels der Evolutionsstrategie*. Chichester ; New York: Wiley. ISBN 0471099880.

[3] Markoff, John (1990-08-29). "What's the Best Answer? It's Survival of the Fittest". New York Times. http://www.nytimes.com/1990/08/29/business/business-technology-what-s-the-best-answer-it-s-survival-of-the-fittest.html.

[4] Ariza, C. 2002. "Prokaryotic Groove: Rhythmic Cycles as Real-Value Encoded Genetic Algorithms." In: Proceedings of the International Computer Music Conference. San Francisco: International Computer Music Association. 561-567. Internet: http://www.flexatone.net/docs/pgrcrvega.pdf

[5] Magnus, C.: Evolving electroacoustic music: the application of genetic algorithms to time-domain waveforms. In: Proceedings of the 2004 International Computer Music Conference. (2004) 173–176.

[6] *Roads*, C. (1988). 'Introduction to Granular Synthesis' Computer Music Journal. MIT Press 12(2): 11-13.

[7] Fijinaga, I., and J. Vantomme. 1994. Genetic algorithms as a method for granular synthesis regulation. *Proceedings of the International Computer Music Conference*. 138-41.

[8] Ariza, C. 2005. "Navigating the Landscape of Computer-Aided Algorithmic Composition Systems: A Definition, Seven Descriptors, and a Lexicon of Systems and Research." In *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association. 765-772. Internet: http://www.flexatone.net/docs/nlcaacs.pdf

MIT OpenCourseWare
http://ocw.mit.edu

21M.380 Music and Technology: Algorithmic and Generative Music
Spring 2010

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.