Sintetizador de Audio Evolutivo con Modulación en Frecuencia

Mariano Leonel Acosta

Asignatura: Modelización y Predicción con Tecnologías Emergentes
Profesor de Cátedra: Dr. Ing. Lucía Isabel Passoni
Carrera: Ingeniería Electrónica
Universidad Nacional de Mar del Plata
marianoacosta.003@gmail.com,isabel.passoni@gmail.com

Resumen. La síntesis sonora en el ámbito de la producción musical es una tarea ardua y requiere de experiencia previa. Resulta de gran utilidad obtener un sistema automático que aprenda a reecrear sonidos a partir de un timbre deseado. En este trabajo se presenta un modelo computacional implementado en Matlab®, utilizando tecnicas de modulación en frecuencia, analisis espectral y ajuste de parámetros mediante algoritmos genéticos, capaz de autoconfigurarse para imitar un archivo de audio precargado. Se muestra a modo de ejemplo la recreación exitosa de un piano acústico. Este método demuestra ser un punto de partida viable para el desarrollo de futuras aplicaciones comerciales. Sin embargo, todavía es necesario mejorar la velocidad de convergencia del algoritmo y evaluar nuevas técnicas a utilizar para el cálculo del fitness.

1 Introducción

Este trabajo fue presentado en el año 2015 como requisito para la aprobación de la materia de grado y postgrado Modelización y Predicción con Tecnologías Emergentes, dictada en la Universidad Nacional de Mar del Plata. El sintetizador es el instrumento más utilizado en la industria musical para la creación de sonidos. Diversas técnicas de síntesis se han desarrollado tanto para la emulación como el desarrollo de nuevos timbres sonoros. El ajuste de los parámetros de un sintetizador a través de la escucha y distinción de sonidos puede resultar en una tarea tediosa principalmente por la naturaleza compleja de la estructura de síntesis. La principal herramienta para juzgar los resultados es la intuición y percepción humana. Por lo tanto, el poder parametrizar y controlar el proceso de diseño sonoro aprovechando las técnicas de inteligencia computacional resulta en una propuesta interesante. Esfuerzos por lograr un sistema de tales características se expresan en [1], donde se estudian distintas topologías empleando programación evolutiva. Como principal inspiración se escogió los trabajos [2], [3] y [4] en los cuales fueron desarrollas interfaces gráficas de usuario empleando computación evolutiva, mostrando tanto ventajas y desventajas en la automatización de la síntesis en comparación con el enfoque convencional. Una aplicación web llamada *Evosynth* [5] utiliza está idea a modo de proceso creativo más que como simple emulación sonora. En [2] se propone un método de optimización de los parámetros de un sintetizador por *modulación en frecuencia* (FM) mediante el uso de algoritmos genéticos. Aunque la síntesis FM ha demostrado ser muy versátil en la creación de timbres acústicos, la expresión matemática de la modulación resulta demasiado compleja para predecir un comportamiento. Basandose en [2], Adam Johnson desarrolla en su tesis de grado un framework en Java [3] para experimentar con algoritmos genéticos, agregando también modulación de parámetros en función del tiempo como el uso de genes reguladores que mejora el tiempo de convergencia. El objetivo del trabajo es desarrollar un sistema propio en Matlab®, partiendo de las recomendaciones de [2] y [3], diseñando un sintetizador desde el principio, proponiendo estructuras nuevas y un genoma acorde para la manipulación del mismo.

2 Síntesis de Audio

La síntesis del sonido consiste en la generación de audio a partir de medios no acústicos. Esto se puede lograr con síntesis analógica, en el cual se utilizan circuitos electrónicos aplicando técnicas como la variación controlada de voltaje, o con síntesis del tipo digital, donde es común emplear programas de computadora para dar forma a las señales de audio, como también la posibilidad de utilizar microcontroladores y circuitos lógicos. Independientemente del tipo de realización, existen tres métodos básicos de síntesis:

- **Síntesis Aditiva:** Consiste en mezclar ondas sinusoidales para construir señales de mayor complejidad. Se basan en el análisis armónico y en el desarrollo en series de Fourier.

- **Síntesis Sustractiva:** Se filtra una señal de gran contenido armónico, atenuando o reforzando determinadas regiones del espectro. En este tipo de síntesis es de gran importancia las características de los filtros, tanto en el diseño en el dominio analógico como su implementación digital, especialmente utilizando la técnica denominada *Virtual Analog* [6].

- **Síntesis por Modulación:** Este método modifica algún parámetro, principalmente amplitud o frecuencia de una onda portadora en función de otra señal, denominada modulante.

Para el diseño del sintetizador acústico propuesto, se decidió implementar un modelo que contempla los tres métodos anteriormente mencionados, pero basando su funcionamiento principalmente en la síntesis por modulación en frecuencia o FM. La expresión (1) revela la forma de la señal FM resultante de modular una portadora sinusoidal de frecuencia ω_c , con una modulante de frecuencia ω_m [7]. El parámetro β es un escalar llamado índice de modulación, siendo $J_n(\beta)$ las funciones de Bessel de primera clase y orden *n*, con A_c la amplitud de la portadora [8].

$$x_c(t) = A_c \sum_{n=-\infty}^{+\infty} J_n(\beta)\cos(\omega_c + n\omega_m)\,t \qquad (1)$$

Por lo tanto, el espectro obtenido está conformado por tonos puros espaciados en múltiplos enteros de la frecuencia modulante, como muestra la Figura 1.

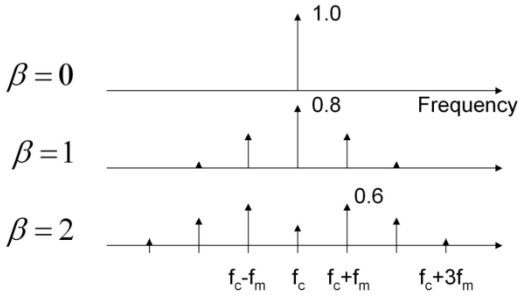

Fig. 1. Espectro de una Señal FM para Distintos Valores de β.

De esta forma, variando el índice de β se logra modificar el contenido espectral de manera dinámica. Si el cociente entre ω_m y ω_c *-denominado radio* - es una relación entera, entonces se produce un fenómeno particular llamado *espectro armónico.* En dicho caso, la forma de onda resultante posee solamente armónicos con frecuencia fundamental $2\pi\omega_c$. Esto resulta de gran utilidad para la generación sonidos musicales. No obstante, cuando esto no ocurre el sonido generado es totalmente *inarmónico*, es decir, que no posee frecuencia fundamental. El espectro inarmónico resulta práctico para la síntesis de sonidos percusivos, por lo que se tuvo en cuenta en el modelo de sintetizador desarrollado.

Cabe aclarar que la componente en continua u *offset* generada en ambos casos deriva en un problema. Si a la señal FM se la utiliza para modular una nueva portadora, el offset producirá una inestabilidad en el tono del sonido final, siendo inaceptable en la producción de un timbre musical. Por eso, es necesario *modular en fase* en vez de en frecuencia [6]. El desarrollo de las bandas laterales en modulación por fase es idéntico al caso FM, pero sin el surgimiento de una componente en continua. Entonces, la ecuación a programar para realizar esta técnica es:

$$x_c(t) = A_c \cos(\omega_c t + \beta \cos(\omega_m t)) \tag{2}$$

Utilizando filtrado dinámico de la señal y variando los parámetros del modulador en función del tiempo, por ejemplo, el índice de modulación, radio o ganancia, se produce un timbre sonoro que evoluciona temporalmente, lo que genera la síntesis de sonidos más 'vivos'. Para la variación paramétrica controlada, se programó la denominada *envolvente ADSR*, llamada así por sus siglas en inglés de *Attack* (Ataque), *Decay* (Decaimiento), *Sustain* (Sostenimiento) y *Release* (Relajación).

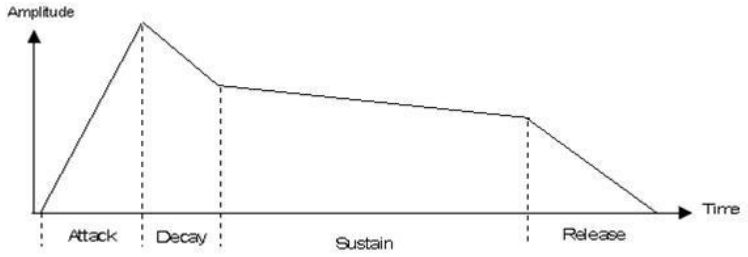

Fig. 2. Envolvente ADSR

3 Modelo del Sintetizador

El esquema básico de modulación FM mostrado en la figura 3 consiste en un bloque A que genera una señal modulante para la portadora del bloque B. Es posible ajustar la frecuencia fundamental de la señal A de forma relativa a través del Radio. Luego, modificando el índice de modulación β, establecemos la cantidad modulación a B y por lo tanto, el contenido espectral resultante. La altura o *Pitch* del sonido se establece con la frecuencia fundamental del oscilador B. Finalmente, existe una etapa para el control de ganancia. Todos estos parámetros mencionados anteriormente pueden ser modulados con las envolventes del tipo ADSR.

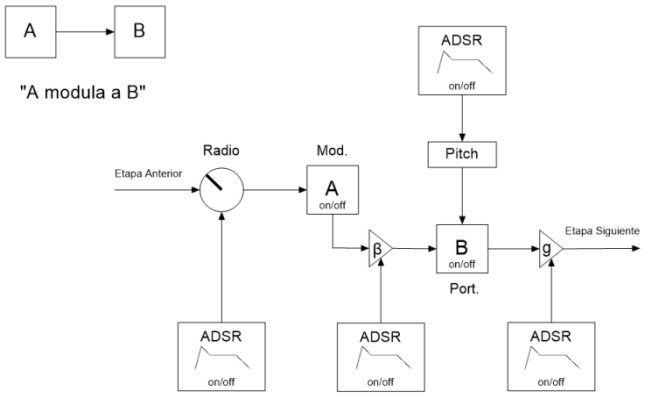

Fig. 3. Esquema Básico de Modulación FM.

Es frecuente modificar la topología de los osciladores con el propósito explorar aún más las posibilidades de síntesis. Por ejemplo, se suele disponer de múltiples portadoras funcionando a la vez, además de generar lazos de realimentación, donde la portadora forma parte de la señal modulante. Por este motivo, fueron programados seis estructuras posibles utilizando cuatros osciladores *A,B,C* y *D* (Figura 4) . La estructura VI corresponde a un modelo DFM, que utiliza portadoras de 0hz. Según enuncia Gan Seum-Lim en su trabajo [7], la síntesis DFM permite la generación de sonidos con envolventes armónicas más complejas que las producidas en FM de forma común. Por lo tanto, se tendrá en cuenta como posible arquitectura para que ser utilizada por el algoritmo. A cada oscilador se le permite generar ondas sinusoidales, triangular, cuadrada y diente de sierra. Existe la posibilidad que la señal generada sea filtrada utilizando un filtro pasabajos denominado *Moog Ladder* [6], con atenuación de 24 dB por octava y frecuencia de corte y sobrepico Q ajustables.

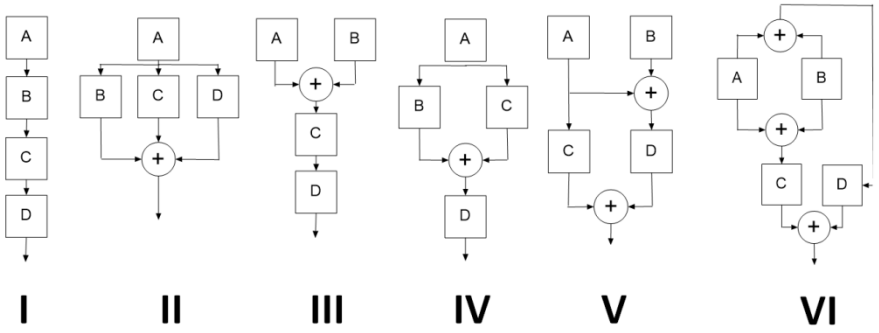

Fig. 4. Topologías de Modulación Programables.

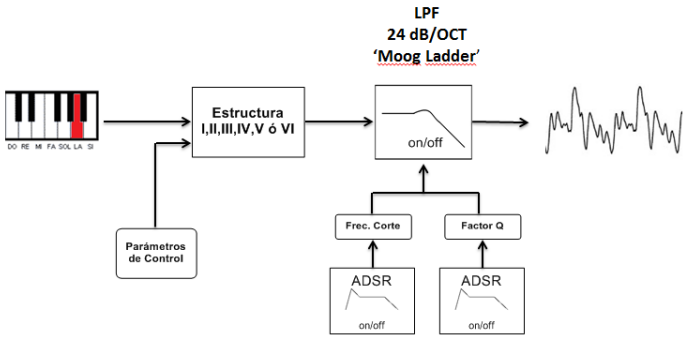

Fig. 5. Diagrama del Sintetizador Diseñado.

4 Algoritmo Genético

Los algoritmos genéticos son algoritmos de optimización de búsqueda múltiple en donde las potenciales soluciones a un problema son codificados mediante *cromosomas*. Estos algoritmos modelan la evolución genética, en donde las características de cada solución o individuos son representadas usando genotipos. La efectividad de cada solución se evalúa a través de una *función de fitness*. Mediante procesos de selección se eligen un conjunto de mejores individuos, donde se obtiene una de-

terminada cantidad de descendencia utilizando operadores de reproducción. Son independientes de conocer posibles métodos de resolución, por lo que resultan adecuados para el problema de programar un sintetizador parametrizable que busca generar un sonido objetivo. Estos algoritmos son de características *metaheurísticas*, es decir, que evitan la convergencia a mínimos locales guiando la búsqueda mediante aproximaciones y consideraciones prácticas. Además, la obtención de la solución óptima no está garantizada.

4.1 Población Inicial

Inicialmente se generan *N* cantidad de individuos, donde cada uno representa una configuración del sintetizador, codificada en 68 cromosomas como posible solución al problema. Para cada individuo se genera un genoma de forma aleatoria. Cada cromosoma o gen puede ser del tipo *binario* (0;1), *discreto* (1;2;3;4;…) o *Real*. Los cromosomas están acotados sobre cierto intervalo de valores, según el parámetro que representen. Existen genes reguladores que determinan si ciertas características van a estar presentes. Por ejemplo: un individuo presenta una configuración para el filtro y sin embargo puede estar desactivado.

Nombre	Descripcion	Rango	Tipo de dato	Nombre	Descripción	Rango	Tipo de dato
on	Activa envolvente (gen regulador)	{0 ,1}	Binario	Ib	Índice de modulación OSCB	[0,40]	Real
Ai	Ataque (envolvente)	[0,1]	Real	Ic	Indice de modulación OSCC	[0,40]	Real
Di	Decay (envolvente)	[0,1]	Real	Idfm	Indice de modulación DFM	[0,40]	Real
Si	Sustain (envolvente)	[0,1]	Real	Ra	Radio OSCA	[0,15]	Real
Level	Nivel despues del Decay	[0,1]	Real	Rb	Radio OSCB	[0,15]	Real
r	Tipo de envolvente (H-L, L-H, H-H)	{0 ,1, 2 }	Discreto	Rc	Radio OSCC	[0,15]	Real
ona	Activa envolvente OSCA	{0 ,1}	Binario	Rd	Radio OSCD	[0,15]	Real
Aai	Ataque (envolvente OSCA)	[0,1]	Real	Radtype	Tipo de radio	{0 ,1}	Binario
Dai	Decay (envolvente OSCA)	[0,1]	Real	Rai	Radio Armónico OSCA	[0:.25:10]	Discreto
Sai	Sustain (envolvente OSCA)	[0,1]	Real	Rbi	Radio Armónico OSCB	[0:.25:10]	Discreto
levela	Nivel después del Decay	[0,1]	Real	Rci	Radio Armónico OSCC	[0:.25:10]	Discreto
ra	Tipo de envolvente OSCA	{0 ,1, 3 }	Discreto	Rdi	Radio Armónico OSCD	[0:.25:10]	Discreto
onb	Activa envolvente OSCA	{0 ,1}	Binario	osca	Tipo de onda OSCA	{1,2,3,4}	Discreto
Abi	Ataque (envolvente OSCB)	[0,1]	Real	oscb	Tipo de onda OSCB	{1,2,3,4}	Discreto
Dbi	Decay (envolvente OSCB)	[0,1]	Real	oscc	Tipo de onda OSCC	{1,2,3,4}	Discreto
Sbi	Sustain (envolvente OSCB)	[0,1]	Real	oscd	Tipo de onda OSCD	{1,2,3,4}	Discreto
levelb	Nivel despues del Decay	[0,1]	Real	inta	Interruptor (ON/OFF) OSCA	{0 ,1}	Binario
rb	Tipo de envolvente OSCB	{0 ,1, 3 }	Discreto	intb	Interruptor (ON/OFF) OSCB	{0 ,1}	Binario
onc	Activa envolvente OSCC	{0 ,1}	Binario	intc	Interruptor (ON/OFF) OSCC	{0 ,1}	Binario
Aci	Ataque (envolvente OSCC)	[0,1]	Real	filt_on	Interruptor (ON/OFF) Filtro.	{0 ,1}	Binario
Dci	Decay (envolvente OSCC)	[0,1]	Real	fcorte	Frecuencia de Corte	[80,18000]	Real
Sci	Sustain (envolvente OSCC)	[0,1]	Real	Qfac	Factor de Calidad	[1,10]	Real
levelc	Nivel despues del Decay	[0,1]	Real	onf	Activa envolvente frec. corte	{0 ,1}	binario
rc	Tipo de envolvente (H-L, L-H, H-H)	{0 ,1, 3 }	Discreto	Afi	Ataque (envolvente frec. corte)	[0,1]	Real
onp	Activa envolvente Pitch	{0 ,1}	binario	Dfi	Decay (envolvente frec. corte)	[0,1]	Real
Api	Ataque (envolvente Pitch)	[0,1]	Real	Sfi	Sustain (envolvente frec. corte)	[0,1]	Real
Dpi	Decay (envolvente Pitch)	[0,1]	Real	levelf	Nivel después del Decay	[0,1]	Real
Spi	Sustain (envolvente Pitch)	[0,1]	Real	rf	Tipo de envolvente frec. Corte	{0 ,1, 3 }	Discreto
levelp	Nivel despues del Decay	[0,1]	Real	onq	Activa envolvente Q	{0 ,1}	Binario
rp	Tipo de envolvente (H-L, L-H, H-H)	{0 ,1, 3 }	Discreto	Aqi	Ataque (envolvente Q)	[0,1]	Real
pam	Cantidad de modulación (Pitch)	[0,100]	Real	Dqi	Decay (envolvente Q)	[0,1]	Real
fm	Frecuencia de Nota	[50,5000]	Real	Sqi	Sustain (envolvente Q)	[0,1]	Real
EST	Estructura del Sintetizador	{1 ,2, 3, 4, 5,6 }	Discreto	levelq	Nivel despues del Decay	[0,1]	Real
Ia	Indice de modulacion OSCA	[0,40]	Real	rq	Tipo de envolvente Q (H-L, L-H, H-H)	{0 ,1, 3 }	Discreto

Tabla 1. Genoma Completo Diseñado.

4.2 Función de Fitness

Un espectrograma consiste en una representación matricial de la variación en frecuencia de una señal en función del tiempo. Esta se basa en segmentar el audio y obtener sus respectivos espectros a partir de la Transformada Discreta de Fourier de Tiempo Reducido (STFT), expresada en la ecuación (3).

$$S(m,\omega) = \left| \sum_{n=\mathrm{m}}^{\mathrm{m+M}} x[n]w[n-m]e^{-j\omega n} \right|^2 \tag{3}$$

Donde $x[n]$ es la secuencia discreta de audio de longitud N y $w[n]$ una función ventana de longitud M ≤ N. A partir del audio generado por un individuo y del sonido que se quiere sintetizar, se calcula los respectivos espectrogramas con la función *spectrogram* del entorno Matlab®. Dicha función devuelve una matriz *NxM*, donde *N* es la cantidad de muestras usadas en la FFT y *M* el número de segmentos. Se eligió utilizar la ventana de Hamming 8192 muestras con un porcentaje de solapamiento del 75%. La función de fitness se basa en calcular dos parámetros definidos en [2], de los cuales se obtiene un único valor mediante una suma ponderada. Se define la norma espectral (4) como la distancia euclidiana entre el espectrograma del audio generado con el del sonido objetivo, siendo W y B la cantidad de filas y columnas respectivamente.

$$\sum_{w=1}^{W}\sum_{b=1}^{B} |f(w,b) - f(w,b)|^2 \tag{4}$$

Para cara segmento se obtiene el centroide espectral (5), el cual está directamente relacionado con el brillo del timbre sonoro:

$$C(a,w) = \frac{\sum_{b=1}^{B} f_a(w,b)xb}{\sum_{b=1}^{B} f_a(w,b)} \tag{5}$$

La diferencia de centroides entre dos espectrogramas (6) se obtiene restando el vector de centroides objetivos con el vector de centroides del sonido generado.

$$C_{dif} = C_{Objetivo}(w) - C_{Sintetizado}(w) \tag{6}$$

Entonces, la función a optimizar resulta en una combinación de la expresión (4) con (6) mediante un balance de puntaje A, definido en el intervalo *[0,1]*.

$$Fitness = A\sum_{w=1}^{W}\sum_{b=1}^{B} |f(w,b) - f(w,b)|^2 + (1-A)\sum_{w=1}^{W} \left|C_{dif}(w)\right| \tag{7}$$

Un puntaje pequeño se interpreta como un sonido sintetizado que posee un timbre y evolución temporal similares al sonido que se quiere imitar. Por lo tanto, se trata de un problema de *minimización* de la función objetivo.

4.3 Selección de la Nueva Generación

Se diseñó un sistema de selección por torneo para elegir los individuos que generarán descendencia. De la población actual se eligen de manera aleatoria una cierta cantidad de individuos, siendo este valor un parámetro programable. Luego, se devuelve el individuo con menor fitness del grupo. Como

se realiza crossover utilizando dos padres, la selección por torneo es realizada dos veces, una para cada progenitor. Cada par de padres genera cuatro hijos, utilizando métodos de cruza y mutación según sea el cromosoma de tipo binario, discreto o real. Para el último caso se programó el algoritmo SRX según se detalla en [10]. Una vez obtenida la descendencia, se elige una cantidad determinada de individuos de la generación anterior y se emplea la estrategia *Kill Tournament.* En cada torneo formado, se busca el individuo con peor fitness (mayor puntaje) y es reemplazado por un hijo

4.4 Sistema Propuesto

En primera instancia se establece la frecuencia de muestreo en la cual el sistema debe trabajar, detallando además la frecuencia fundamental del sonido objetivo, el tamaño de los torneos de selección, la probabilidad de mutación y la ruta de ubicación del archivo de audio. Luego, el sistema genera de una población inicial de *N* cantidad de individuos inicializados con un genoma totalmente aleatorio que, mediante un intérprete de genoma, son traducidos en parámetros de funcionamiento para el sintetizador FM. Para cada individuo, el sintetizador devuelve un sonido sintetizado que es comparado con el timbre objetivo a través de (7). En base a esto, a cada individuo se le asigna un valor de fitness el cual se utiliza para producir la selección de padres y su consecuente descendencia, dando lugar a una nueva generación, comenzando el proceso nuevamente. El programa funciona iterativamente hasta que un cierto número de generaciones es alcanzado. En la Figura 6 se muestra una representación gráfica del sistema.

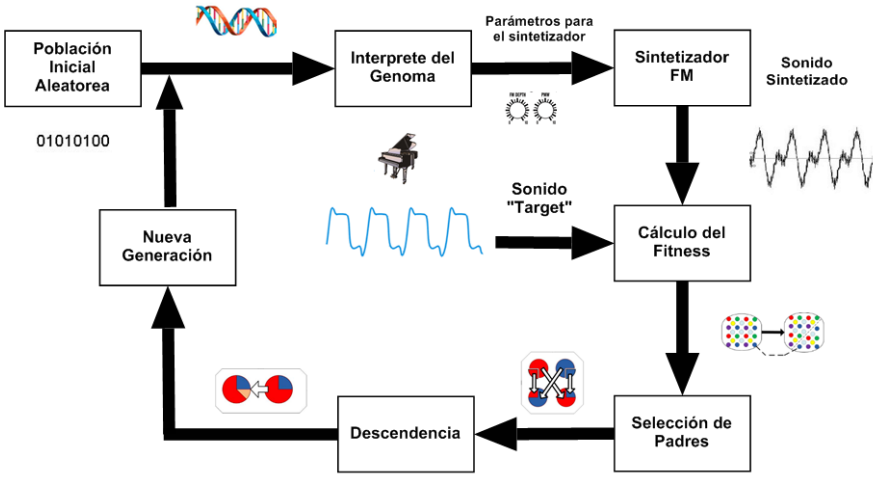

Fig. 6. Representación Gráfica del Sistema Final.

5 Resultados

5.1 Recreación de un Piano Acústico

En la Tabla 2 se muestran los parámetros iniciales para el algoritmo genético. Nótese que no fue diseñado un algoritmo de detección tonal, por lo que la frecuencia fundamental debió ser ingresada de forma manual. El tiempo total para correr 1000 iteraciones en una computadora de escritorio con procesador i7 2700k y 8gb de RAM fue de 5 horas. Como resultado, se describe la configuración obtenida interpretando el genoma del mejor individuo (Tabla 3).

Parámetro	Valores
Frecuencia de Nota	523,25 hz. (Do 5ta Octava)
Tamaño de Torneo	6.
Tamaño Kill Tournament	10.
Cantidad de Individuos	100.
Cantidad de Generaciones	1000.
Probabilidad de Mutación	5%
Balance de Puntaje	0,5.
Resolución/Frec. Muestreo	32 bits/44,1 kHz.

Tabla 2. Parámetros utilizados para la recreación.

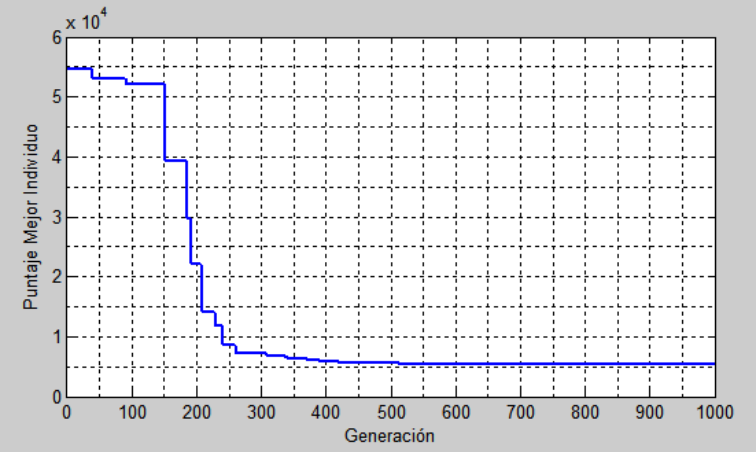

Fig. 7. Evolución del Mejor Fitness a Través de las Generaciones.

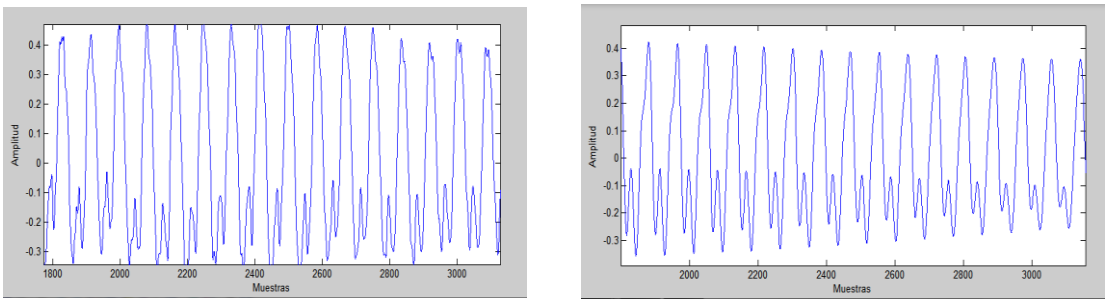

Fig. 8. Forma de Onda Objetivo (Izquierda) y Resultante (Derecha).

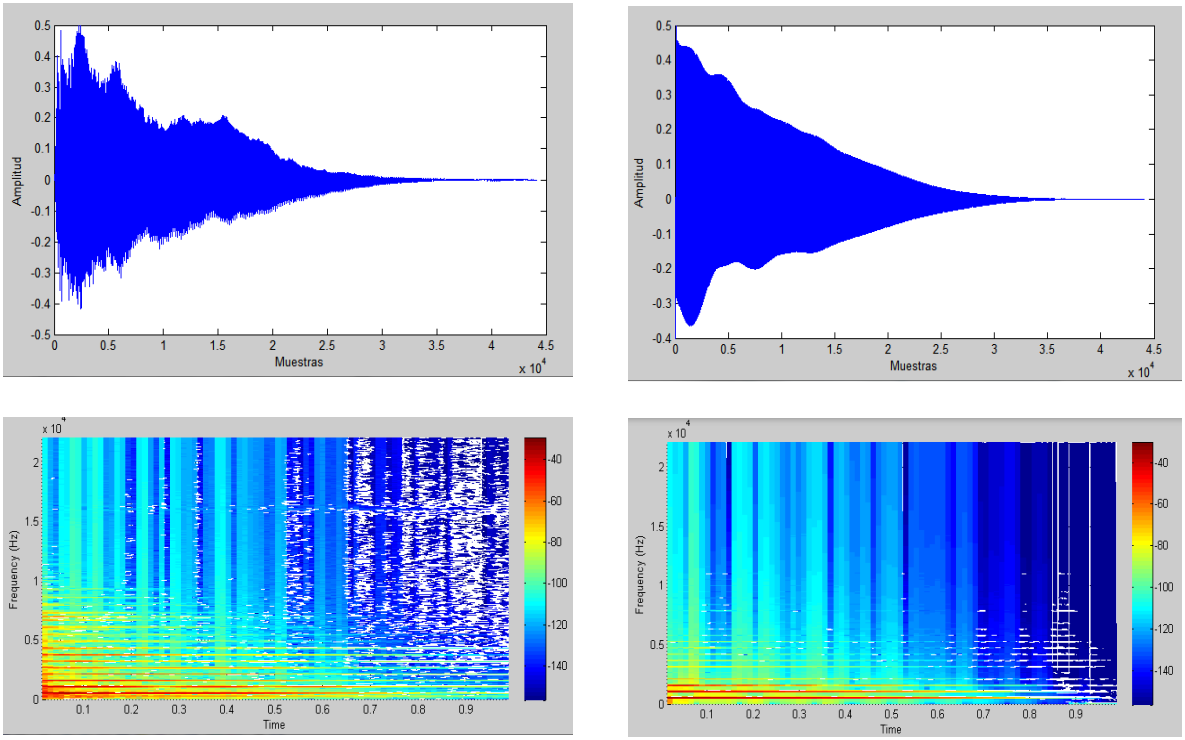

Fig. 9. Evolución Temporal y Espectral del Sonido Objetivo (Izquierda) y Resultante (Derecha).

Parámetro	Valores
Osciladores Activos:	B y D
Envolventes	Amplitud, OSC B y Pitch Desactivadas
Estructura del sintetizador:	VI (DFM)
Índice de Modulación B a D	4,516
Tipo de Radio	Armónico
Radio B	1
Radio D	0
Filtro	Activado
Frecuencia de Corte	561,08 hz.
Factor Q	1,72 (Activado)
Envolvente Frec. Corte	Activada
Formas de Onda	Sinusoidales

Tabla 3. Configuración Encontrada por el Algoritmo Genético

6 Conclusiones y Futuros Trabajos

Se detalló e implementó un sistema de síntesis de audio en Matlab® utilizando tecnicas de modulación en frecuencia, analisis espectral y ajuste de parámetros mediante algoritmos genéticos. A modo de prueba se realizó la recreación del timbre de un piano acústico. Comparando las formas de onda y envolventes obtenidas con la señal de audio original mostradas en las Figuras 8 y 9, se puede apreciar que presentan características similares. Si bien la evolución espectral fue imitada correctamente, el audio generado presentó menos intensidad en las altas frecuencias. Perceptualmente, el timbre sintetizado coincidió con el de un piano acústico, por lo que se concluye que la experiencia fue exitosa. Un resultado apreciable fue que el mejor individuo presentó codificado en su genoma la estructura de síntesis DFM. Esto puede significar que dicha configuración en particular es la más versátil de las seis propuestas, considerando además que es capaz de producir señales más complejas que en FM simple. Según se observa en la Figura 7, luego de 500 iteraciones no hubo mejoras apreciables, pudiéndose estimar que la cantidad óptima de generaciones puede estar cercana a ese valor. Para corroborarlo es necesario probar el sintetizador con diferentes timbres y una cantidad considerables de muestras.

Dentro de las futuras mejoras del sistema, se precisa experimentar con parámetros adaptativos de balance de puntaje y probabilidad de mutación. También es necesario estudiar funciones de fitness alternativas, tales como la utilización de transformadas wavelets o coeficientes cepstrales en escala Mel. Principalmente, deben ser abarcadas funciones que reduzcan la exigencia computacional. Si se quiere desarrollar una aplicación que muestre resultados en un tiempo aceptable, es necesario realizar una implementación más eficiente en un lenguaje compilado como C++. Esto puede ser crítico en el proceso musical creativo, ya que el momento de inspiración del artista es corto en comparación con el tiempo de respuesta del sintetizador. Un enfoque que no fue abarcado en este trabajo es el concepto de realimentación en el esquema FM, tal como el utilizado en la matriz de modulación del sintetizador comercial FM8 de la empresa Native Instruments® [11], pudiendo ofrecer una modulación más práctica. Con estas mejoras presentes el siguiente paso sería diseñar un plug-in del tipo *Virtual Studio Tecnology* para ser utilizado en el entorno de una estación de audio digital.

Referencias

1. Garcia R.: Growing Sound Synthesizers using Evolutionary Methods. MIT Media Lab, Machine Listening Group 20 Ames St. E15-401, Cambridge, MA 02139, USA.
2. Lai Y., Kang Jeng S., Tzung Liu D., and Chung Liu Y.: Automated Optimization of Parameters for FM Sound Synthesis With Genetic Algorithms. International Workshop on Computer Music and Audio Technology (2006).
3. Johnson A.: Sound Resynthesis with a Genetic Algorithm. Final Year Project. Imperial College of London. (2011).
4. McDermott J., Griffith N.J.L. And O'Neill M.: Evolutionary GUIs for Sound Synthesis. Applications of Evolutionary Computing Vol.4448, 547-556 (2007).
5. Yee King, M.: Evosynth. http://www.yeeking.net/evosynth/
6. Pirkle W.: Designing Software Synthetizer Plug-Ins in C++. Taylor & Francis eBooks. (2014),
7. Seum –Limd G.: Digital Synthesis of Musical Sounds, National University of Singapore (1992).
8. Chowning J. M.: The Synthesis of Complex Audio Spectra by Means of Frequency Modulation.
9. Marshall D.: MATLAB Digital Audio Synthesis. Laboratory Worksheet Week 7, Module No: CM0340.
10. Engelbrecht, A. P.: Computational Intelligence. University of Pretoria, South Africa (2007).
11. Native Instruments: FM8, Manual de Usuario. (2006).