MIDIEvo: Sistema Genético para Evolucionar Pistas MIDI

Antonio R. Quirós S.
Departamento de Informática
Universidad Carlos III de Madrid
Madrid, España

***Abstract*—Los algoritmos evolutivos han sido usados como alternativa para problemas de búsqueda y optimización en muchas y sobre todo diversas áreas. Una de las aplicaciones más interesantes es el arte. El uso de algoritmos evolutivos para la generación de expresiones artísticas, como pintura o música, es una de las formas más explícitas de percibir el funcionamiento de este paradigma. En este trabajo, se presenta el desarrollo de MIDIEvo, un sistema genético que modela las pistas MIDI como individuo y las evoluciona hasta parecerse lo más posible a una canción de entrada. Adicionalmente se implementan estrategias para mantener la diversidad genotípica.**

Keywords—Algoritmos evolutivos; MIDI; MIDIEvo; Algoritmos genéticos.

I. Introducción

Durante muchos años, los algoritmos evolutivos han sido una alternativa viable para la resolución de problemas de optimización y búsqueda. Estos métodos han sido implementados en un amplio espectro de dominios, como la economía, ingeniería y en años recientes el arte. El uso de estas técnicas con inspiración biológica para el desarrollo de sistemas artísticos es un área de investigación interesante y emocionante. Hay un creciente interés en la aplicación de estas técnicas en campos como: arte visual; generación, análisis e interpretación de música; arquitectura; video; poesía; diseño; y otras tareas creativas[1].

En este trabajo, se presenta el desarrollo de un sistema con inspiración biológica, que toma como entrada una pista de audio en formato MIDI; modela las pistas MIDI como individuos usando su estructura interna y evoluciona una población inicial de canciones generadas aleatoriamente, hasta que alguna llegue a ser igual a la pista de entrada; o después de un número de generaciones definido, sea lo más parecida posible.

El arte evolutivo, a diferencia de otras áreas de aplicación de estos métodos, tiene la particularidad de ser muy explícito, visual y tangible, lo que facilita la percepción de la influencia de los conceptos biológicos en la representación y desarrollo de las soluciones. Basado en esto, MIDIEvo está desarrollado pensado en servir como una plataforma didáctica, que permite comprobar la evolución de las soluciones a medida que se van dando; es decir, no se concentra solo en dar una respuesta final, sino que permite al usuario experimentar las soluciones a lo largo del proceso evolutivo.

Complementariamente, en este trabajo se implementan dos estrategias para fomentar la diversidad genotípica. Con una se utiliza una tasa de mutación adaptativa de acuerdo al coeficiente de variación de los valores de fitness de cada individuo; con el otro se busca evitar la generación de clones cuando dos padres manejan el mismo material genético; generando hijos aleatorios si se da el caso.

El trabajo está dividido de la siguiente forma: en la sección II se discuten trabajos previos; en la sección III se detallan conceptos técnicos de los ficheros MIDI; en la sección IV se discute el algoritmo evolutivo implementado; en la sección V se detallan las estrategias de diversidad genotípica implementadas; en la sección VI se explica a MIDIEvo como una herramienta didáctica; en la sección VII se registran los experimentos ejecutados y los resultados obtenidos; en la sección VIII se plantean las conclusiones y en la sección IX las referencias citadas.

II. Trabajos Previos

En el área de música evolutiva han habido diversos trabajos interesantes, desde la generación de música hasta la clasificación por géneros.

Maximos A. Kaliakatsos–Papakostas, Michael G. Epitropakis, Andreas Floros, y Michael N. Vrahatis propusieron un sistema de Evolución Interactiva que utiliza Programación Genética para evolucionar melodías simples de 8-bits[2]; la evolución es impulsada hacia los sonidos más agradables al usuario.

José Fornari desarrolló un sistema capaz de emular un amplio rango de canto de aves[3] y con ello presentar un paisaje sonoro evolutivo e interactivo, ya que además recibe y utiliza data externa.

III. MIDI

En este trabajo se desarrolló un sistema evolutivo que recibe como entrada una pista en formato MIDI y una serie de parámetros relativos al algoritmo y evoluciona una población de canciones para ir acercandose en similitud a la pista de entrada.

A. MIDI como interfaz

MIDI es un acrónimo (en Inglés) para Interfaz Digital de Instrumento Musical y es un estándar, un protocolo, un lenguaje y una lista de especificación[4].

Los archivos MIDI no son grabaciones de música; en cambio, son secuencias de instrucciones, que pueden ocupar 1000 veces menos espacio en disco que una grabación.

B. *Estructura de un fichero MIDI*

La estrucura de un fichero MIDI se puede visualizar en la figura 1. Básicamente un fichero MIDI está compuesto de un conjunto de pistas (Track); cada pista es una lista de eventos (MidiEvent). Cada uno de los eventos se componen de un Mensaje (MidiMessage) y un valor numérico para un instante (Tick). Los mensajes pueden ser de 3 tipos (ShortMessage, MetaMessage y SysexMessage). Los mensajes de tipo ShortMessage contienen la información relevante al sonido y la música, por ende, nos concentraremos en este tipo de mensaje.

Un ShortMessage está compuesto por cuatro valores principales: (i) Command, que representa la acción a ejecutar, este puede tomar un valor entero. Los ficheros MIDI cuentan con capacidad de almacenar complejas estructuras musicales, por ende tienen un amplio rango de acciones. De todos los valores que puede tomar como comando, solo nos interesan dos, NOTE_ON y NOTE_OFF. NOTE_ON puede ser visto como presionar una tecla en un piano, y NOTE_OFF soltarla. (ii) Channel, indica el canal de audio en el que se ha de reproducir la nota y (iii) Key que representa la nota a ser reproducida; el Key puede tomar un valor numérico que va entre 0 y 127. Por último, Velocity, básicamente es la fuerza con la que se presiona la tecla; representa el volumen con el que sonará la nota.

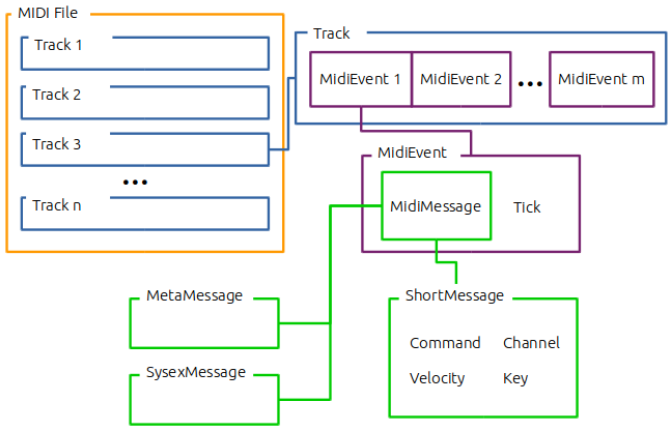


Fig 1: Estructura de un fichero MIDI

IV. Algoritmo Evolutivo

El primer paso para modelar el problema a ser resuelto usando algoritmos evolutivos, es representar una canción como un individuo a ser evolucionado. Este trabajo se basó en ficheros MIDI de una sola pista de audio (para disminuir el poder de cómputo necesario para procesar los individuos); sin embargo se puede extender sencillamente a ficheros de más pistas.

A. *Codificación del individuo*

Como se explicó antes, una pista está compuesta por un conjunto de eventos y cada evento tiene un mensaje. Para esta implementación se plantea la pista como un individuo como se muestra en la figura 2.

Se hicieron dos abstracciones de los componentes MIDI relacionados con el sonido: Track y ShortMessage. De ShortMessage nos interesa solo los mensajes de sonido (Command igual a NOTE_ON o NOTE_OFF); de estos mensajes nos interesa la acción (command), la nota (key) y el instante de tiempo (tick) ya que estas son las partes que se evolucionarán; además se incluye el canal para poder convertirlo en ShortMessage y reproducirlo como audio luego. Para el Track se creó un MIDIEvoTrack, que consta principalmente de una lista de SimplifiedShortMessages; además tiene una lista adicional de MidiMessages originales para todos los mensajes que no tienen componentes de nota (MetaMessage, SysexMessage y ShortMessage sin NOTE_ON o NOTE_OF).

El individuo entonces es un MIDIEvoTrack; el cromosoma es una lista de SimplifiedShortMessages de longitud igual al número de ShortMessages de la canción original; cada gen del cromosoma es un SimplifiedShortMessage.

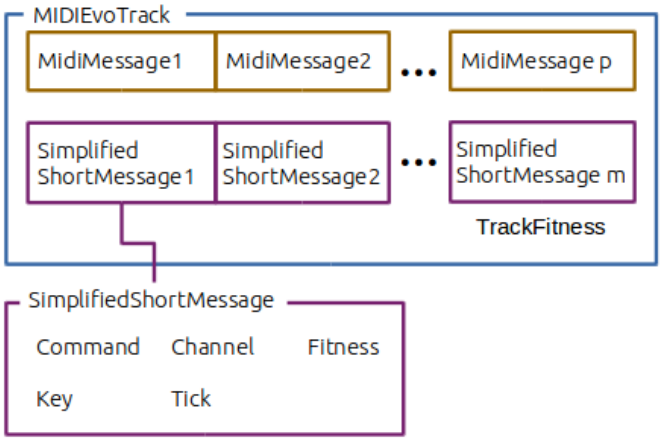


Fig 2: Abstracción de Track

B. *Función Fitness*

Se define la función fitness para un MIDIEvoTrack como cuán diferente es del Track original. Para esto es necesario calcular el fitness de cada SimplifiedShortMessage. Dado smo el ShortMessage original y ssm el SimplifiedShortMessage; se define la función fitness del gen

$$diffCmd = |smo.cmd - ssm.cmd|$$
$$diffKey = |smo.key - ssm.key|$$
$$diffTick = |smo.tick - ssm.tick|$$
$$fitness(ssm) = diffCmd + diffKey + diffTick$$

Definimos entonces la función fitness del MIDIEvoTrack como la sumatoria de los valores de fitness de cada SimplifiedShortMessage; por ende, a menor valor de fitness, mejor será el individuo (ya que menor será la diferencia con respecto a la canción objetivo). Dado SSM la lista de SimplifiedShortMessage del MIDIEvoTrack y $ssm \in SSM$

$$\sum_{ssm \in SSM} fitness(ssm)$$

C. Selección

Para la selección se usó la estrategia de torneo. Se escogen tres individuos al azar y se selecciona el mejor de los tres para reproducirse. Esto se repite un número especificado de veces.

D. Recombinación

De la lista de seleccionados, se escogen pares para reproducirse. Se genera un nuevo individuo escogiendo aleatoriamente un gen (SimplifiedShortMessage) de un padre u otro.

E. Mutación

Los hijos generados en la recombinación pasan a la mutación. Se elige aleatoriamente la parte del SimplifiedShortMessage a modificar (Key, Command o Tick). Por cada SimplifiedShortMessage del MIDIEvoTrack se genera un número aleatorio *m* entre 0 y 1, si es menor a la tasa de mutación (0.1) se muta el gen, cambiando la parte seleccionada por una nueva generada aleatoriamente.

F. Reemplazo

Se ordena la pobalción (usando el fitness) y se reemplazan los peores individuos (últimos en la lista) por los hijos generados. Es decir, los mejores individuos pasan a la siguiente generación.

V. Estrategias de Diversidad

Los algoritmos evolutivos en ocasiones sufren de convergencia prematura a un óptimo local. Este problema está altamente asociado a la pérdida de diversidad genética entre los individuos de la población; lo cual produce un decremento o estancamiento en la calidad de las soluciones encontradas.

En la implementación realizada, en ocasiones se presentaba este problema; por ende se implementaron alternativas para darle robustez al sistema.

Basado en el trabajo de Miguel Rocha y José Nieves[5], se implementaron dos estrategias para la diversidad genética, una para la mutación y otra para la recombinación: Tasa de Mutación Adaptativa (AMR) y Generación Aleatoria de Descendientes (ROG).

A. Tasa de Mutación Adaptativa

La mutación busca evitar la linealidad en las soluciones, incorporando nuevos genes creados aleatoriamente. Una tasa de mutación muy baja, puede tener muy bajo impacto en las soluciones encontradas. Por otra parte, una tasa de mutación muy alta puede agregar mucho ruido al sistema, y empeorar soluciones muy buenas.

Para superar estos obstáculos, se incorporó una estrategia adaptativa basada en la diversidad genética de la población, medida en espacios regulares de tiempo, usando la desviación estándar de los valores de fitness de toda la población como medida para estimar la diversidad.

El proceso es el siguiente, se inicia con un valor para la tasa de mutación, y en intervalos regulares de tiempo, se prueba el valor de la desviación estándar; si es menor a un límite predefinido, se incrementa la tasa.

En esta implementación, se agregó una modificación a la planteada por Rocha y Nieves; se define un límite superior; si la desviación estándar es mayor al límite superior se decrementa la tasa de mutación. Además se acotaron las tasas de mutación (0.0 el mínimo y 0.3 el máximo).

B. Generación Aleatoria de Descendientes

Una de las características que se presenta cuando la población converge en un óptimo local es que los individuos comparten el mismo material genético. Cuando esto ocurre, hay una gran probabilidad de que, al hacer el sobrecruzamiento puede recibir de entrada dos individuos con igual genotipo. En este caso la recombinación de su material genético se hace inefectiva, dado que el hijo será simplemente un clon de sus padres.

La idea detrás de esta estrategia es probar el material genético de los individuos antes de hacer el sobrecruzamiento; si se percibe la situación descrita anteriormente, no se ejecuta la recombinación; en su lugar, se genera un descendiente completamente aleatorio.

VI. MIDIEvo como herramienta didáctica

Una vez definidas las características del sistema, se realizó la implementación en Java (versión 1.8). El sistema recibe los siguientes parámetros:

- Fichero MIDI de entrada.
- Tamaño de Población
- Número de Torneos
- Número máximo de generaciones
- Estrategia de Diversidad
- Modo

El fichero de entrada servirá como objetivo. El tamaño de la población define el número de individuos con el que trabajará el sistema. El número de torneos indica la cantidad de individuos que serán seleccionados para reproducirse (Generarán #Torneos/2 hijos). El número máximo de generaciones es una condición de parada al algoritmo. Estrategia de diversidad definida para probar el rendimiento (None, AMR o ROG).

Modo es un parámetro que permite al usuario “asistir” al algoritmo. Para generar una pista nueva, el sistema debe ir creando mensajes indicando la acción (presionar o soltar una tecla), la nota (cual tecla presionar o soltar) y el instante (en que momento ejecutar la acción). En el modo EASY se le indica al sistema en cual instante debe ejecutar la acción, dejándole solo la tarea de generar la acción y la nota; esto hace que el proceso evolutivo sea más rápido. En modo HARD no se le da ninguna asistencia al sistema; y este debe, por consiguiente, generar la acción, la nota y el instante; esto complica mucho más el problema y por ende toma un tiempo mayor la ejecución.

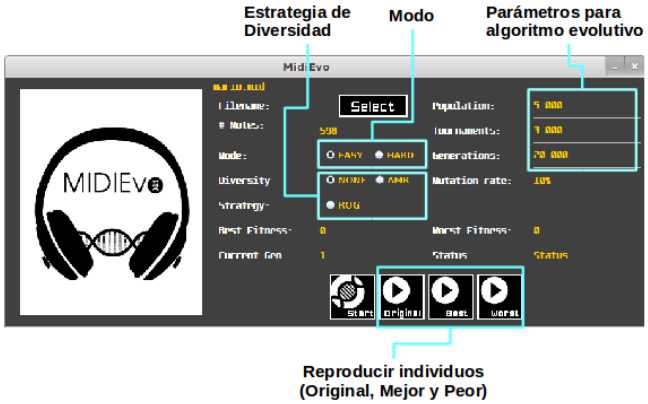


Fig 3: MIDIEvo Interfaz

Como se ha mencionado anteriormente, el arte evolutivo se presenta como una forma mucho más expresiva de percibir los algoritmos evolutivos. En este trabajo se presenta MIDIEvo como una herramienta para visualizar la evolución a medida que esta se ejecuta. La herramienta permite al usuario indicar los parámetros que controlan el ambiente del sistema y así verificar los cambios en los resultados. Además, al estar pensado para servir como una herramienta didáctica, permite reproducir, en cualquier momento de la ejecución, al mejor y peor individuo como una canción, a modo de escuchar, verificar y comparar la calidad de las soluciones a medida que las va encontrando.

Además, se van actualizando en tiempo real los valores de fitness del mejor y peor individuo; así como también la tasa de mutación actual, si se utiliza el AMR como estrategia de diversidad.

Todo esto hace del MIDIEvo una herramienta útil para adquirir una visión del funcionamiento de un sistema genético; pudiendo percibir (mediante la reproducción de audio) la evolución de los individuos en tiempo real, y a su vez, pudiendo controlar los parámetros para la ejecución, y de ese modo, comprobar cuales combinaciones arrojan mejores resultados.

VII. Experimentación

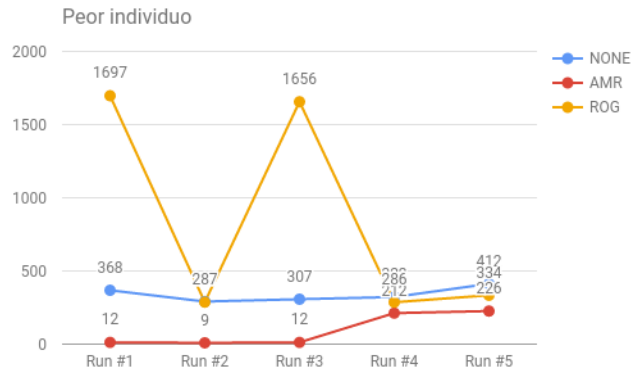
Para poner a prueba el sistema implementado, así como la influencia de las estrategias de diversidad, se preparó un conjunto de pruebas distribuidas de la siguiente manera:

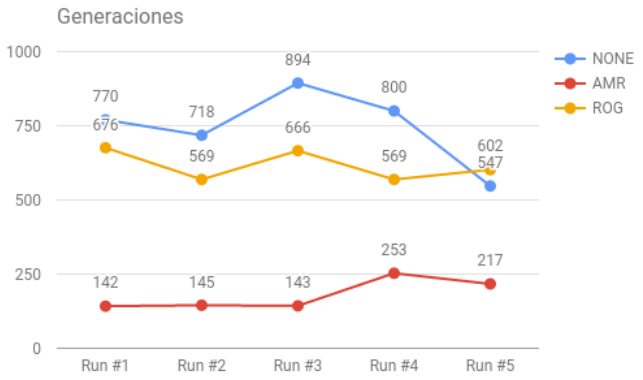
Se selección la canción “Mario - Game Over Sound.mid”. Esta canción es de longitud 48 (El número de genes (cantidad de SimplifiedShortMessages) es 48).

En primer lugar se realizan un conjunto de pruebas usando la configuración EASY, esto es, el algoritmo debe generar la acción a realizar y la nota (El instante de tiempo se le especifica); esta prueba es mucho más ligera en tiempo y procesamiento.

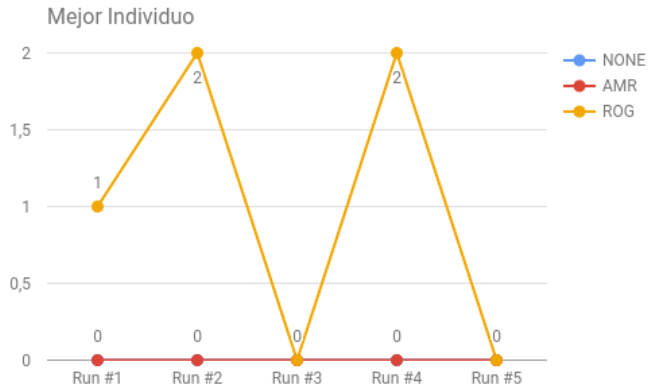
Por cada una de las 3 estrategias de diversidad, se ejecutaron 3 pruebas y por cada prueba se hicieron 5 ejecuciones. En cada prueba se cambian los parámetros usados para población, cantidad de torneos y número de generaciones.

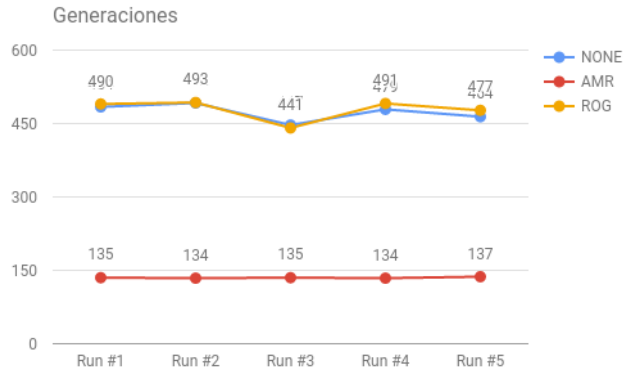
Prueba #1				
Modo	**Población**	**Torneos**	**Max. Gen.**	**Tests**
Easy	5000	3000	20000	5

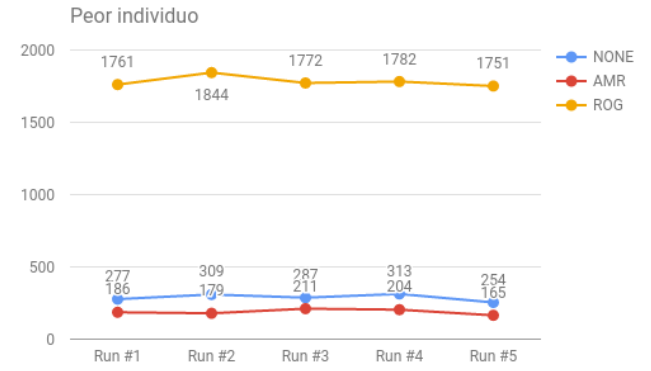




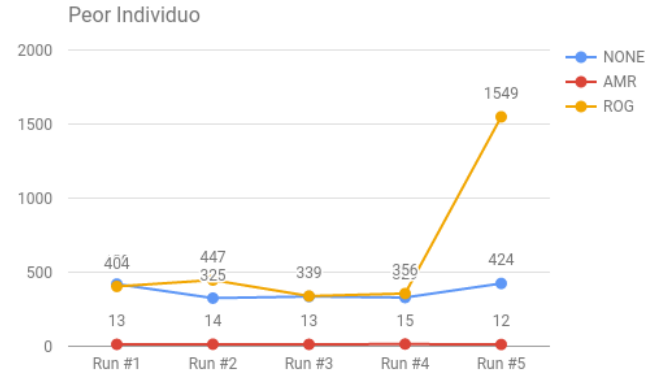
Prueba #2				
Modo	**Población**	**Torneos**	**Max. Gen.**	**Tests**
Easy	1000	500	20000	5







Prueba #3				
Modo	**Población**	**Torneos**	**Max. Gen.**	**Tests**
Easy	12000	8000	20000	5



En las pruebas 1 y 3 todas las ejecuciones lograron obtener el individuo óptimo (Fitness 0) por lo que no se grafican.

En la primera prueba, se hacen las ejecuciones con un tamaño “medio” de población y número de torneos. Aunque todas las estrategias logran obtener la solución óptima, queda evidenciado que la tasa de mutación adaptativa (AMR) influye positivamente en el algoritmo, tanto en la calidad de las soluciones (Mejores peores individuos al final); como en el número de generaciones necesarias para obtener la solución. ROG por su parte, empeora la ejecución del algoritmo; aunque el observar el peor individuo obtenido al final, se puede interpretar como que se maximiza la diversidad o la diferencia entre los individuos; aunque esto sin embargo, no se traduce en una mejora para el algoritmo (esta diversidad no mejora las mejores soluciones).

La segunda prueba, se parametrizó un tamaño pequeño de población y número de torneos. En este caso el resultado fue el mismo que en la prueba uno; la diferencia está en los mejores individuos, que ROG en 3 ocasiones no logra obtener el resultado óptimo.

La tercera prueba se amplía el tamaño de población y número de torneos. Con un número mayor de individuos ROG logra un comportamiento muy similar a no usar ninguna estrategia, por lo que no representa ninguna mejora. En esta caso, una vez más AMR reporta mejores resultados, tanto en tiempo (Número de generaciones) como calidad de los individuos.

Prueba #4				
Modo	**Población**	**Torneos**	**Max. Gen.**	**Tests**
Hard	5000	3000	20000	3

	Run	Best	worst	Gen
NONE	Run #1	91	117298	20000
	Run #2	92	106240	20000
	Run #3	105	101455	20000
AMR	Run #1	23	23	20000
	Run #2	32	32	20000
	Run #3	21	21	20000
ROG	Run #1	321	509370	20000
	Run #2	218	479390	20000
	Run #3	657	491437	20000

La cuarta prueba, se usa la misma parametrización que la prueba 1; sin embargo esta vez se utiliza el modo HARD. Con esto, el algoritmo debe ubicar las notas en el instante de tiempo correcto (además de determinar la acción y la nota). En el caso de prueba el número de instantes posibles es de 23.040, esto da un total de 5.898.240 posibilidades por cada una de las 48 notas que componen la canción.

En este caso 20.000 generaciones no son suficientes para que ninguna de las estrategias logre obtener la solución óptima; sin embargo, AMR una vez más reporta los mejores resultados, tanto en el mejor individuo como en el peor.

Como prueba adicional, se decidió comprobar cual estrategia puede alcanzar el individuo óptimo en menor número de generaciones. Se elevó el número máximo de generaciones a 100.000 y se ejecutaron las tres estrategias dos veces cada una. ROG y NONE no lograron culminar con éxito en ninguna ocasión; AMR una vez más demuestra su potencial, logrando obtener la mejor solución en una ocasión (84.664 generaciones).

Todas las pruebas se realizaron en un ordenador con procesador Intel Core i7-3770 3.4GHz con 8 Gb de RAM.

VIII. Conclusiones y Trabajo a Futuro

Los algoritmos evolutivos ha cobrado fuerza a través de los años como alternativas efectivas para la obtención de soluciones a problemas de alta complejidad en optimización y búsqueda. En años recientes, se han aplicado las técnicas con inspiración biológica en diversas ramas del arte, como la pintura, video y música.

En este último aspecto, varios trabajos se han desarrollado para generar y componer de forma evolutiva música, siguiendo patrones y reglas extraídas de corpus de datos, o de teoría musical. En este proyecto no se genera música nueva; sino que se utiliza un algoritmo evolutivo que va evolucionando canciones hasta que se parezcan lo más posible a una de entrada.

Si bien, este proyecto no es convencional en el sentido de que no busca una solución, o visto de otro modo, la solución "óptima" ya se conoce de antemano (la canción original); la idea principal detrás de su creación es contar con una herramienta que explote ese conocimeinto previo en el sentido de poder mostrarse de una forma muy explícita, las diferencias de calidades entre los individuos generados (el mejor y el peor) y el individuo objetivo (Canción original).

Además, se implementan dos conceptos interesantes para fomentar la diversidad genética, que son AMR y ROG; el primero aplica directamente a la mutación y el segundo a la recombinación.

Los experimentos realizados demostraron que ROG no mejora la ejecución de los algoritmos, esto debido a que, aunque si aumenta la diversidad genética, no necesariamente mejora a los mejores individuos; símplemente funciona como un elemento estocástico para "refrescar" a la población. Por otro lado, AMR si tienen un impacto muy positivo, mejorando al algoritmo genético tanto en tiempo (menor número de generaciones necesarias para obtener la solución óptima), como en calidad de las soluciones encontradas (El peor individuo es muy bueno).

Como trabajo a futuro se plantea la extensión del sistema para considerar canciones que incluyan más de una pista. Esto pudiera hacerse ejecutando el proceso evolutivo por cada pista encontrada en el archivo original y al momento de reproducir, se unen los mejores individuos de cada pista.

Otro trabajo a futuro va motivado a la extensión de la componente didáctica de la herramienta; la implementación de nuevas estrategias para la diversidad genética como por ejemplo la Técnica de Desastre Social (Social Disaster Technique o SDT)[5].

IX. Referencias

[1] J. R. J. McDermott and A. Carballal, "Evolutionary and Biologically Inspired Music, Sound, Art and Design," 2013.

[2] M. A. Kaliakatsos--Papakostas, M. G. Epitropakis, A. Floros, and M. N. Vrahatis, "Interactive Evolution of 8--bit melodies with Genetic Programming towards finding aesthetic measures for sound," in Evolutionary and Biologically Inspired Music, Sound, Art and Design, Springer, 2012, pp. 141–152.

[3] J. Fornari, "Evolutionary and Biologically Inspired Music, Sound, Art and Design: First International Conference, EvoMUSART 2012, M{á}laga, Spain, April 11-13, 2012. Proceedings," P. Machado, J. Romero, and A. Carballal, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 96–107.

[4] R. Guérin, MIDI Power! Thomson Course Technology, 2006.

[5] M. Rocha and J. Neves, "Preventing premature convergence to local optima in genetic algorithms via random offspring generation," in Multiple Approaches to Intelligent Systems, Springer, 1999, pp. 127–136.