Departamento de Informática

PROYECTO FIN DE CARRERA

Sistema Inteligente de Composición Armónica

Autor: José Ángel Canabal Delgado

Tutores: Sergio Jiménez Celorrio
Tomás de la Rosa Turbides

Leganés, octubre de 2012

Título: Sistema Inteligente de Composición Armónica
Autor: José Ángel Canabal Delgado
Directores: Sergio Jiménez Celorrio y Tomás de la Rosa Turbides

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Parece que fue ayer cuando comenzó esta aventura, con demasiados personajes importantes e historias que recordar. Quiero agradecer a mis compañeros los buenos ratos pasados, a Raúl, Martín, Aitor, Rafa, Javier, Charlie, Diego, Laura, Andrés y tantos otros con los que he compartido más que aula, cafetería o mesas de pasillo, y que han sabido aguantar todas mis rarezas.

A Jose, que más que compañero y amigo es como el hermano mayor que vela por mí, sin cuyo apoyo quizás nunca hubiese decidido unir la música y la informática, por lo que parte de este proyecto se lo debo a él.

A Sara, por saber comprenderme y apoyarme cuando lo he necesitado ayudándome a mejorar y a superar cualquier dificultad que yo solo no habría conseguido, si soy mejor se lo debo a ella.

A mi familia por su apoyo incondicional durante todos estos años, por permitirme hacer lo que creía que debía hacer sin imponerme nunca sus preferencias.

No quiero olvidar tampoco a toda la gente del conservatorio, Sara, Nacho, Edu, con los que he pasado grandes momentos y compartido una misma pasión.

Por último, quiero agradecer a mis tutores Sergio y Tomás por su apoyo, dedicación y confianza en mí y en este proyecto desde el principio.

A todos gracias.

El genio se compone del dos por ciento de talento y del noventa y ocho por ciento de perseverante aplicación.
Ludwig van Beethoven

Resumen

Toda composición musical está sujeta a reglas que definen su estilo, regulando el esqueleto de la composición, del mismo modo que la experiencia musical adquirida durante años por el músico afectará a los patrones musicales de la obra, haciéndola diferente del resto de composiciones.

Este trabajo pretende dar un nuevo enfoque a cómo afrontar la composición armónica automática aunando estos dos conceptos, reglas y experiencia, generando una estructura armónica, a partir de una línea melódica, por medio de un Sistema Experto y modificando cada una de las voces generadas en base a la experiencia, almacenada en forma de partituras, usando un sistema de Aprendizaje Automático basado en Instancias. El resultado será una melodía escrita en MusicXML que se aproxime al resultado obtenido por una persona en términos de calidad de expresión armónica.

Se pretende demostrar que la composición de obras musicales expresivas, con estilo propio, no son exclusivas de las personas sino que son reproducibles utilizando técnicas de Inteligencia Artificial permitiendo que cualquier persona, sin necesidad de disponer de conocimientos musicales, pueda disfrutar de sus propias composiciones armónicas sin esfuerzo.

Palabras clave: MusicXML, Sistema Experto, Aprendizaje Automático basado en Instancias, Composición armónica automática, Inteligencia Artificial.

Abstract

Any musical composition is subject to rules that define his style, regulating the composition framework, in the same way that the musical experience acquired over the years by the musician will affect musical patterns of the work, making it different from other compositions.

This work tries to provide a new approach about how to address the automatic harmonic composition joining this two concepts, rules and experience, creating an harmonic framework, from a melodic line, through an Expert System and modifying each of the generated voices based on experience, stored in the form of scores, using a Machine Instance Based Learning. The result will be a melody written in MusicXML that comes close to the result obtained by a person in terms of harmonic expression quality.

This works aims to show that the composition of expressive musical works, with a own style, doesn't be exclusive of people but they are reproducible using Artificial Intelligence techniques allowing everyone, without need of music knowledge, to enjoy their own harmonic compositions effortlessly.

Keywords: MusicXML, Expert System, Machine Instance Based Learning, Automated Harmonic Composition, Artificial Intelligence.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Introducción

Durante siglos el arte ha sido considerado como el producto más excelso del ser humano, fruto de su sensibilidad y afán de trascendencia. Una forma de expresión nacida de las emociones para emocionar.

Este trabajo nace con la idea de unir dos mundos aparentemente muy distantes entre sí como son la informática y el arte musical pues hay belleza artística en la ciencia, y ciencia en el arte.

Se persigue demostrar que el proceso creativo realizado por una persona, en este caso un músico realizando una armonización, es reproducible por medio de técnicas de Inteligencia Artificial. Es fácil pensar que las actividades artísticas o creativas escapan de la lógica lo que las convierten en quimeras para máquinas carentes de sentimientos cuando en realidad puede ser solo un problema de enfoque.

La música hace uso de un lenguaje simbólico a diferencia de otras disciplinas artísticas que la hace más manejable computacionalmente. La composición, la improvisación o la armonización manejan estos símbolos para crear música utilizando para ello reglas, unas veces muy bien definidas por cánones estéticos, y otras veces simplemente por imitación de otras obras.

Podríamos decir que en la música, el proceso creativo obedece a dos máximas: respetar una serie de reglas que definen un estilo concreto y la modificación y creación de obras en función de la experiencia adquirida por el músico.

Todas las personas están condicionadas por estas máximas aun incluso si no se dedican a la música. Desde la infancia se ven expuestas a estilos propios de su cultura lo que afectará a su proceso creativo, es por ello que dos composiciones raramente son semejantes sin haber sido algo premeditado.

En este proyecto se va a intentar simular estas dos máximas por medio de distintas técnicas de Inteligencia Artificial: el estilo va a ser definido por un Sistema Experto basado en reglas mientras que la experiencia del individuo va a ser simulada por medio de un Sistema de Aprendizaje Automático basado en Instancias, donde todo ha sido diseñado por módulos para facilitar ampliaciones en estilos y funcionalidad.

1.2 **Objetivos**

El propósito de este proyecto es diseñar e implementar un sistema informático capaz de armonizar una melodía dada obteniendo una estructura de tipo coral, de manera que la melodía proporcionada por el usuario sea considerada la voz más aguda del coro (voz soprano) generando de forma automática otras tres voces de acompañamiento armónico (contralto, tenor y bajo).

El objetivo es intentar simular el proceso de armonización que puede realizar una persona, para lo que será necesario cumplir una serie de subobjetivos:

- Ser capaz de leer una partitura musical en su formato electrónico, en concreto en MusicXML.
- Obtener una melodía reducida sobre la que se generará el esqueleto armónico.
- Crear un sistema experto que genere una armonización completa de una melodía dada: deberá seguir un procedimiento similar al que realizaría un experto en la materia.
    - Deberá cumplir las reglas de la armonía tradicional.
    - Soporte para acordes tríada en estado fundamental y primera inversión.
    - Que sea fácilmente ampliable en el futuro para soportar distintos estilos, reglas y tipos de acordes.
- Ser capaz de analizar una partitura, obteniendo diversa información útil para calcular el grado de similitud entre dos partituras cualesquiera o sus fragmentos.
- Modificar una armonía ya generada, creando nuevas estructuras rítmicas o melódicas en el acompañamiento inspiradas en patrones extraídos de un conjunto de partituras dado por medio de Aprendizaje Automático.
- Escribir la partitura resultante de todo el proceso en formato MusicXML.

1.3 **Estructura de la memoria**

Esta estructura se compone de nueve capítulos y dos anexos.

- Capítulo 1: Introducción y objetivos. Contiene una introducción del trabajo realizado así como los objetivos a cumplir.
- Capítulo 2: Estado del arte. Proyectos comerciales existentes así como diversas investigaciones sobre Inteligencia Artificial y composición automática.
- Capítulo 3: Análisis. Justificación de las tecnologías utilizadas frente a otras alternativas.
- Capítulo 4: Diseño. En este apartado se recoge la arquitectura y los diagramas de flujo del sistema.
- Capítulo 5: Implementación. En este capítulo se detallan los módulos que componen la arquitectura así como su comunicación.
- Capítulo 6: Experimentación. Incluye la descripción y análisis de los distintos experimentos realizados.
- Capítulo 7: Gestión del proyecto. Incluye la planificación estimada y final del proyecto así como el coste total del mismo.
- Capítulo 8: Conclusiones. Se hará una valoración de los resultados obtenidos y en qué grado se han cumplido los objetivos planificados.
- Capítulo 9: Líneas Futuras. Mejoras posibles al sistema así como posibles vías nuevas de investigación.
- Anexo I: Teoría musical. Definición de la teoría musical básica para entender el proyecto en su totalidad.
- Anexo II: Manual de usuario. Guía de instalación y uso del sistema.

Capítulo 2

Estado del arte

A diferencia de otras disciplinas artísticas como la pintura, la música es un medio de expresión artística que utiliza un lenguaje simbólico que puede formalizarse y representarse en un ordenador. Son muchos los trabajos que tratan de aplicar técnicas informáticas, más concretamente técnicas de Inteligencia Artificial, en proyectos musicales. Existen tres campos principales de investigación:

- Composición: el objetivo es la creación automática de melodías y armonías.
- Improvisación: trata de variar una melodía tratando de imitar a un intérprete improvisando.
- Interpretación: se puede dotar de expresividad a obras creadas por ordenador de modo que se asemejen a interpretaciones humanas así como modificar la interpretación de obras ya existentes.

Este trabajo está enmarcado en los sistemas de composición automática. En este campo existen dos bloques principales sobre cómo abordar el problema de la composición, un primer enfoque basado en aplicar reglas que generen la obra y un segundo que trata de extraer patrones de obras existentes. La propuesta planteada se inspira en estos dos enfoques para resolver el problema de la composición automática de armonías.

2.1 **Música e Inteligencia Artificial**

La música y la Inteligencia Artificial están ligadas desde hace más tiempo del que uno pueda creer. El primer trabajo del que se tiene conocimiento en esta área es el proyecto de Hiller e Issacson [Hil] en 1958. Su principal trabajo fue Illiac Suite, un cuarteto de cuerda basado en un modelo de generación y prueba. El sistema generaba notas de forma pseudoaleatoria utilizando para ello cadenas de Markov. Estas notas eran puestas a prueba por medio de una serie de heurísticas obtenidas de la composición clásica, la armonía y contrapunto. En el supuesto de que ninguna de las notas satisficiera las reglas en un punto determinado se recurría a técnicas de backtracking para comenzar desde un punto inmediatamente anterior. En palabras de Hiller e Isaacson primero hay que resolver cuestiones más simples sobre la composición antes de tratar la parte expresiva y emocional de la música.

Es a partir de este momento cuando surgen numerosos trabajos basados en técnicas probabilísticas, concretamente usando para ello cadenas de Markov. Los resultados obtenidos en los distintos trabajos hicieron ver que no se puede obtener música lo suficientemente coherente valiéndose únicamente de métodos probabilísticos.

Cierto es que pese a todo no todos los trabajos de la época estaban basados en cadenas de Markov. El ejemplo más destacable es el trabajo de Moorer en 1972 [Moo] quien trató de simular las técnicas usadas por los músicos de manera algorítmica junto con el uso de heurísticas. Moorer no fue el único que evitaba el uso de probabilidades, Levitt [Lev] hace lo propio al sostener que la probabilidad oscurece los resultados impidiendo apreciar las reglas que generan las estructuras musicales más simples. Su trabajo se centra en la descripción de distintos tipos de estilos musicales por medio de restricciones.

Uno de los pioneros en la aplicación de técnicas de Inteligencia Artificial en la composición musical fue Rader [Rad] en el año 1974. Rader propone un sistema basado en reglas para decidir qué acordes deben acompañar a otros para crear melodías con cierta coherencia musical El sistema cuenta con una serie de reglas de aplicabilidad con un peso determinado que decide que regla generativa se emplea según el contexto. Es por ello el primer ejemplo de uso de metaconocimiento aplicado en el contexto de la composición musical.

En el año 1969 Rothgeb [Rot] crea un programa en Snobol que permite resolver la armonización de un bajo utilizando para ello un conjunto de reglas de enlaces de acordes, algunas de ellas se utilizarán en este trabajo. Pese a todo el objetivo del trabajo no era obtener un sistema de composición automática sino probar teorías de armonización de bajos del siglo XVIII computacionalmente.

Bharucha [Bha] creó MUSAT, un sistema pensado para capturar cualidades musicales de intuiciones armónicas utilizando redes neuronales para aprender modelos armónicos. MUSAT trata de aprender las cualidades que crean las sucesiones armónicas, como por ejemplo la íntima relación que existe entre la función tonal de dominante de crear tensión y la tónica de resolver dicha tensión.

Pero sin duda el trabajo musical más conocido que relaciona la música y la Inteligencia Artificial es EMI, realizado por David Cope [Cop]. Se trata de un sistema que analiza partituras de un autor específico buscando patrones musicales recurrentes llamados signaturas que más tarde recombina para formar nuevas composiciones tratando de emular el estilo del compositor analizado. El algoritmo principal está basado en cuatro fases:

- Análisis: intenta usar características de la música tonal para inferir el carácter y la función de determinados grupos de notas.
- Identificación de patrones: se buscan las signaturas que serán utilizadas en la estructura básica de la nueva composición.
- Deconstrucción: las partituras básicas de entrada se dividen en pequeños segmentos musicales.
- Reconstrucción: Se recombinan los segmentos creando una nueva obra.

Tras realizar la composición de los motivos musicales entre signaturas el sistema utiliza un analizador implementado como una red aumentada de transiciones que constituye el núcleo de la fase de reconstrucción, siendo el encargado de buscar las restricciones o propiedades que caracterizan a las obras del mismo compositor.

2.2 **Trabajos de investigación**

A continuación se incluyen una serie de trabajos que han servido de inspiración al mismo con el objetivo de conseguir mejores resultados.

2.2.1 **CHORAL**

En el año 1993 Ebcioglu [Ebc] desarrolla CHORAL, un sistema experto de armonización de corales según el estilo de Johann Sebastian Bach. El sistema genera una armonización a partir de una línea melódica dada por medio de reglas heurísticas y restricciones. La implementación del sistema fue realizada utilizando un leguaje de programación lógica creado por el propio autor. Gracias al uso de primitivas lógicas para reprensentar los distintos puntos de vista de la música se pueden manejar una gran cantidad de conocimientos musicales complejos.

2.2.2 **Extracción de Patrones**

El trabajo de Karsten Verbeurgt, Michael Dinolfo y Mikhail Fayer [KMM] describe un nuevo acercamiento a la composición musical algorítmica utilizando técnicas de extracción de patrones de composición por medio de cadenas de Markov. Se trata de un sistema de aprendizaje automático donde las probabilidades de las cadenas de Markov son aprendidas de secuencias musicales de las que se han extraído los patrones.

Se tienen en cuenta tres dimensiones a la hora de determinar la equivalencia entre patrones: el compás, el timbre tonal y la duración. Para demostrar la viabilidad de la aplicación como método de composición los autores utilizan como secuencias de entrenamiento música de carácter clásico.

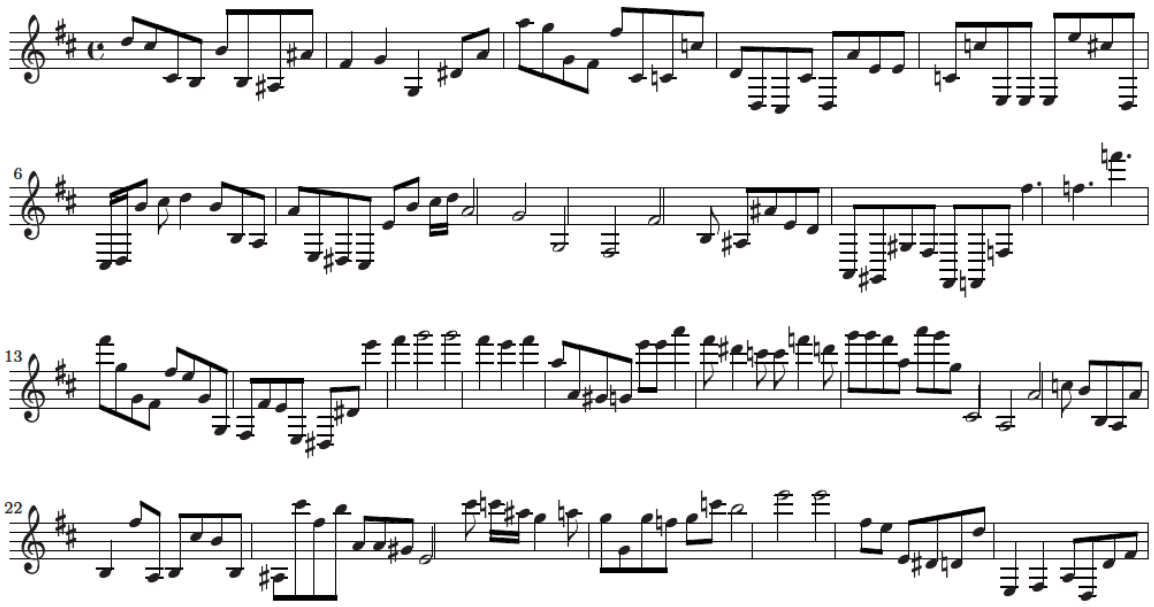


Figura 1: Resultado de composición automática [KMM]

La figura 1 muestra una composición realizada por el sistema. El resultado muestra indicios de cierta consistencia musical, concretamente en los patrones rítmicos, no así en los melódicos. Esto se debe principalmente a la dificultad de generar melodías consistentes por medio de métodos probabilísticos tal y como señalaban distintos autores como Levitt.

2.2.3 **SaxEx**

La tarea de SaxEx [ALM] es aprender a inferir un conjunto de transformaciones expresivas para aplicar a cada nota de una entrada inexpresiva monofónica. En otras palabras, dotar de expresividad a una obra. Para ello el sistema utiliza una memoria de casos con ejemplos de interpretaciones humanas analizadas por medio de técnicas de modelado espectral y conocimiento musical. SaxEx además cuenta con una puntuación de cada interpretación.

El sistema trata de inferir para cada nota entrante su rol en la frase musical, identificando y sustituyéndolas por notas de roles similares del conjunto de datos de las interpretaciones humanas, y transformar la nota de entrada para que contenga sus propiedades expresivas (dinámicas, vibrato, rubato, articulaciones, y ataques de la nota).

SaxEx utiliza dos tipos de análisis diferentes, primero utiliza técnicas de modelado espectral que proporcionan un valor cualitativo en la expresividad de cada nota de la interpretación humana. Estos posibles valores son: muy bajo, bajo,

medio, alto y muy alto. Para calcularlos el sistema realiza una media de la obra total y en función de su valor define el valor de cada nota como la diferencia con respecto a la media.

El segundo análisis aplica la teoría generativa de la música tonal y el sistema Narmour's para determinar el rol de cada nota en la frase musical. Una vez el sistema ha realizado una búsqueda sobre qué notas se parecen más a la de la entrada utiliza lógica borrosa para ajustar la expresividad lo mejor posible.

2.3 **Software comercial**

A continuación se describen una serie de de productos comerciales que realizan tareas similares a las de este proyecto.

2.3.1 **Harmony Builder**

Harmony Builder [Har] es un software de composición musical creado por *Musilogic* que combina el poder de la automatización en la detección de errores con la conveniente colección de herramientas de composición, como la rueda de selección de acordes y el generador de voces.

El panel de construcción permite al usuario componer, tanto armónicamente utilizando para ello la progresión de patrones armónicos sugeridos por el programa, como melódicamente seleccionando desde el menú la armonización total de todas las notas de la melodía.

El programa también permite utilizar inversiones de acordes en las generaciones automáticas así como tener una respuesta instantánea sobre los acordes que se han utilizado.

2.3.2 **Band in a Box**

Band in a Box [Ban] es un software inteligente de acompañamiento automático creado por *PG Music*. Entre sus características más importantes se encuentra un editor de notación, lírica, pistas melódicas, armonización y generación de estilos. Cuenta con una opción denominada *Solista* que genera solos sobre cualquier progresión de acordes.

2.3.3 **Tonica Fugata**

Tonica Fugata [Ton] es un software de composición automática creado por la compañía alemana *Capella Software*. Se trata de una herramienta de edición musical capaz de generar de forma automática construcción de acordes, armonizaciones, cánones y fugas, esta última a partir de un motivo creado por el usuario previamente.

Es en la generación automática de fugas donde realmente destaca *Tonica Fugata*. El sistema es capaz de generar composiciones imitando los estilos de algunos de los compositores contrapuntísticos más célebres destacando por encima de todos Johann Sebastian Bach.

El sistema se vale de distintas técnicas de análisis musical y redes de neuronas para obtener composiciones que intentan imitar a los grandes maestros pero permitiendo cierta variabilidad en el sistema que hace que una tarea pueda ser resuelta de múltiples formas.

Capítulo 3

Análisis

3.1 Análisis de tecnologías y alternativas

Este capítulo describe las diferentes alternativas de tecnologías valoradas, estableciendo en qué medida satisfacen o no los objetivos perseguidos y justificando finalmente las opciones seleccionadas.

3.1.1 Lenguajes de Representación

La notación de la música ha sido siempre un elemento complejo, ya que no sólo debía indicar la altura de los sonidos, sino también los restantes parámetros de la música: duración, tempo, intensidad sonora, carácter, articulación, etc. En este apartado se repasan los formatos electrónicos de representación musical más comunes.

3.1.1.1 MIDI

Las siglas de MIDI proceden de Musical Instrument Digital Interface. Se trata de un lenguaje de representación musical que permite la comunicación entre instrumentos digitales y otros aparatos electrónicos mediante una interfaz común.

Se caracteriza por poder almacenar obras de gran complejidad, con múltiples instrumentos utilizando para ello muy poco espacio de almacenamiento. Esto es

posible dado que un fichero MIDI no contiene las ondas de sonido en sí, sino únicamente los mensajes y eventos que son interpretados por un sintetizador que lo traduce a señales sonoras.

MIDI únicamente almacena eventos referentes a la interpretación musical, no pudiendo incorporar su forma escrita lo que hace inapropiado su uso en tareas donde la forma escrita es importante.

Ventajas	Desventajas
Permite reproducir la melodía sin necesidad de un editor.	Imposible distinguir las enarmonías lo que lo hace inapropiado para representar obras armónicas.
Estándar de comunicación entre instrumentos digitales.	Imposible discernir elementos expresivos
	La traducción de una partitura a MIDI no es una función inyectiva y por tanto no es invertible el proceso.
	Difícil manejo sin editor gráfico

Tabla 1: Ventajas y desventajas de MIDI

3.1.1.2 **Otros formatos de audio**

WAVE

WAV (o WAVE), apócope de WAVEform audio file format, es un formato de audio digital normalmente sin compresión de datos desarrollado y propiedad de Microsoft e IBM que se utiliza para reproducir y almacenar sonidos, admite archivos mono y estéreo a diversas resoluciones y velocidades de muestreo, su extensión es .wav.

Su uso en Internet no es popular, fundamentalmente porque los archivos sin compresión son muy grandes. Son más frecuentes los formatos comprimidos con pérdida, como el MP3.

MP3

MPEG-1 Audio Layer III o MPEG-2 Audio Layer III, más comúnmente conocido como MP3, es un formato de compresión de audio digital patentado que usa un algoritmo con pérdida para conseguir un menor tamaño de archivo. Es un formato de audio común usado para música tanto en ordenadores como en reproductores de audio portátil.

Ventajas	Desventajas
Permite reproducir la melodía sin necesidad de un editor.	Complicada manipulación al no estar representado por un lenguaje simbólico.
Mismo resultado sonoro para todos los usuarios al no depender de un banco de sonidos para su reproducción.	La única forma de manejar la información tonal es mediante el uso de frecuencias.

Tabla 2: Ventajas y desventajas de otros formatos de audio

3.1.2 **Sistemas Expertos**

Según la RAE, un sistema experto es un "programa de ordenador o computadora que tiene la capacidad de dar respuestas semejantes a las que daría un experto en la materia". Son por tanto programas que contienen el conocimiento de un experto en un tema concreto, por lo que pueden ayudar a mejorar la productividad en una materia particular, ahorrando tiempo y dinero.

Los Sistemas Expertos se pueden entender como una rama de la Inteligencia Artificial, donde el poder de resolución de un problema viene dado por el conocimiento de un dominio específico. Se basan en el conocimiento declarativo y conocimiento de control, esto son hechos e información sobre el seguimiento de una acción.

Un sistema experto [CA] está formado por tres grandes núcleos: una Base de Conocimientos, La Memoria de trabajo o Base de Hechos y el Motor de Inferencia.

- La Base de Conocimientos es donde se almacena el conocimiento modelado que se ha extraído del experto.
- La Memoria de trabajo contiene los datos iniciales del problema a resolver.
- El Motor de Inferencia es el módulo de software que modifica los hechos iniciales por medio del conocimiento del experto para alcanzar la solución al problema. Existen distintas estrategias para alcanzar una solución, entre ellas el encadenamiento de reglas para alcanzar una solución o la inversa, dada una solución tratar de inferir si es correcta por medio de las reglas.

Esta estructura dividida en módulos permite que la lógica y los datos se mantengan separados facilitando la mantenibilidad del sistema, además de permitir extender a dominios cruzados y multidominio permitiendo, como es en el caso de este trabajo, organizar las reglas en distintos ficheros. Proporciona por tanto la estructura de un sistema escalable y rápido.

La base de conocimientos se basa en la programación declarativa, esto implica que la solución es alcanzada por medio del encadenamiento de reglas. Cada regla modifica los hechos que activan las precondiciones de otras reglas que a su vez vuelven a modificar los datos hasta alcanzar una solución al problema.

El algoritmo Rete y sus derivados proporcionan métodos eficientes para implementar un sistema de producción de reglas. El algoritmo Rete [For] fue creado por el Dr. Charles L. Forgy en el seno de la Carnegie Mellon University siendo documentado en su tesis doctoral en 1979 pese a que su primera referencia escrita data de 1974. Este algoritmo se encuentra hoy día en la base de muchos sistemas expertos, algunos de ellos muy famosos, incluyendo CLIPS, Jess o Drools.

El algoritmo Rete se basa en la construcción de una red de nodos, donde cada uno de ellos, a excepción de la raíz, representa un patrón que aparece en la parte izquierda de una regla, en el condicional. Cada nodo tiene una lista de los hechos que satisfacen su patrón. Se crea por tanto una estructura Trie en la que se sacrifica memoria para incrementar la velocidad de procesamiento, lo que lo convierte en una de las bases de las implementaciones más eficientes de sistemas expertos.

3.1.2.1 **Jess**

Jess [Fri] es un motor de reglas y entorno de programación escrito en Java por Ernest Friedman-Hill. Se trata de un superconjunto de CLIPS [Cli] que permite generar software Java con cierta capacidad de razonamiento usando conocimiento formado por hechos y reglas declarativas. Se trata de uno de los motores de reglas más rápidos a la vez que pequeño y ligero. Utiliza una versión mejorada del algoritmo de Rete en el procesamiento de las reglas y su motor de inferencia utiliza un tipo de encadenamiento hacia adelante.

En Jess las reglas pueden ser escritas usando el lenguaje de reglas propio de Jess, con una sintaxis muy similar a la sintaxis de CLIPS, o definirlas utilizando XML. Las reglas a su vez pueden ejecutar código Java, código que puede ir embebido entre las reglas sin causar ningún tipo de conflicto.

Ventajas	Desventajas
Permite ejecutar código Java	No es open source, licencia comercial cara
Muy fácil de integrar en un proyecto Java	Algo Menos de rendimiento que CLIPS 6.30
Algoritmo Rete mejorado	
Uno de los motores de reglas más rápidos	
Buen soporte, con pocos bugs encontrados	

Tabla 3: Ventajas y desventajas de Jess

3.1.2.2 **Drools**

Drools [Mcw], desarrollado por Bob McWhirter en 2001 y registrado en SourceForge, es un sistema de reglas con un motor de reglas basado en inferencia de encadenamiento hacia adelante. Al igual que CLIPS o Jess utiliza una implementación avanzada del algoritmo Rete denominada ReteOO haciendo referencia a que está preparada para sistemas orientados a objetos.

El sistema implementa el estándar JSR-94 para el motor de reglas y framework de empresa para construcción, mantenimiento y ejecución de las políticas de negocio en una aplicación, organización o servicio. Además, Drools utiliza JackRabbit para gestionar el respositorio de reglas, y el estándar JAAS para la autorización y autenticación.

Drools cuenta con un sistema nativo de lenguaje de reglas, con un formato muy permisivo en puntuación y soporte de lenguajes naturales y de dominio específico por medio de expansiones.

Ventajas	Desventajas
Escrito en java	Menos estable que CLIPS
Preparado para sistemas orientados a objetos	Menor apoyo que otras herramientas
Código abierto	Rendimiento inferior a CLIPS y Jess

Tabla 4: Ventajas y desventajas de Drools

3.1.3 **Aprendizaje Automático**

El Aprendizaje Automático [AA] es una rama de la Inteligencia Artificial cuyo cometido es el de desarrollar técnicas que permitan al programa aprender de la experiencia.

En el desarrollo software existen problemas en los que no existen algoritmos, estos se encuentran mal definidos o propuestos de manera informal donde resulta atractivo introducir conocimiento a través de ejemplos. En concreto el Aprendizaje Automático se centra más en la búsqueda de soluciones factibles a problemas donde las técnicas de desarrollo de software más tradicionales no proporcionan un buen resultado, resultando muy útiles en la búsqueda de patrones en los ejemplos.

Vivimos en un marco en el que la sociedad genera una gran cantidad de datos que es difícilmente analizable donde el Aprendizaje Automático proporciona una forma de explotarlos tratando de encontrar relaciones entre ellos.

Existen distintos tipos de algoritmos de Aprendizaje Automático que se agrupan distintas taxonomías dependiendo de la salida de los mismos.

- Aprendizaje supervisado: el algoritmo genera una función de correspondencia entre los datos de entrada y las salidas de dichas entradas.
- Aprendizaje no supervisado: en este tipo de aprendizaje todo el proceso de modelado se lleva a cabo sobre el conjunto de ejemplos formado únicamente por las entradas al sistema. El sistema tiene que ser capaz de reconocer los patrones para poder etiquetar nuevas entradas.
- Aprendizaje semisupervisado: se trata de un híbrido de los dos anteriores, el sistema recibe datos de entrada supervisados y no supervisado.
- Aprendizaje por refuerzo: el algoritmo aprende observando el mundo que le rodea. Cada acción es retroalimentada con la información obtenida de sus acciones. El sistema aprende por tanto a base de ensayo-error.
- Aprendizaje multitarea: se trata de métodos de aprendizaje que usan conocimiento previamente aprendido en un dominio para enfrentarse a problemas de índole similar.

Nuestro cerebro realiza un proceso de aprendizaje constante, recibiendo nuevos datos, comparándolos con la experiencia y realizando acciones en base a ello. Este proceso se realiza de forma natural e inconsciente lo que dificulta saber qué factores influyen para poder ser representado en un modelo matemático o de software.

Por tanto, para realizar un buen aprendizaje es necesario tener en consideración el concepto de conocimiento, que es el entendimiento de un determinado tema o materia, sin el cual no se puede dar el aprendizaje. En el aprendizaje automático podemos encontrar tres tipos de conocimiento:

- Crecimiento: almacenamiento de ejemplos.
- Reestructuración: nuevo conocimiento resultante de la interpretación del conocimiento.
- Ajuste: adaptación del aprendizaje.

3.2 **Tecnologías empleadas**

A continuación se describe el conjunto de tecnologías seleccionadas para el desarrollo del proyecto.

3.2.1 **Lenguajes de representación**

3.2.1.1 **MusicXML**

En el plano musical los instrumentos y programas informáticos musicales se entienden por medio del estándar MIDI.

Pese a su gran utilidad en este marco, presenta un importante número de limitaciones de representación de la información musical dado su enfoque interpretativo lo que imposibilita diferenciar entre notas de distinto nombre pero misma sonoridad, repeticiones, compases y múltiples aspectos en cuanto a notación.

En este marco han surgido una serie de lenguajes de representación musical con el objetivo de paliar las limitaciones del lenguaje MIDI. Estos lenguajes están enfocados en la representación digital de toda la información contenida en una partitura siendo perfectos para ser utilizados por programas de edición musical, facilitando el proceso de guardado y renderizado de una partitura.

De los numerosos lenguajes que existen destaca MusicXML [Mus]. Este lenguaje de notación musical creado por la compañía Recordare es hoy día la propuesta más madura y extendida, entre otros motivos por haber sido desarrollado paralelamente con un software de uso extendido.

Además, el uso de una licencia gratuita para el DTD y XSDs ha propiciado su uso entre terceras compañías, quedando así como la propuesta más utilizada actualmente siendo utilizado por las empresas líderes en el sector de la edición musical tales como Sibelius o Finale.

MusicXML está basado en las normas y formatos definidos en el meta-lenguaje XML, lenguaje que es un subconjunto de SGML (Standard generalizadas Mark-up Language). El uso de XML está fuertemente ligado al almacenamiento y manejo de información así como con la filosofía Internet-friendly, por lo que resulta muy natural almacenar el contenido de una partitura en un archivo de este tipo

Estructura

La estructura de un archivo MusicXML viene definida dentro del tag <score-partewise> donde se define la versión MusicXML del mismo.

El contenido viene definido por dos elementos de carácter obligatorio. En un primer lugar se encuentra el tag <part-list> en cuyo contenido se define la información de las partes que forman la partitura y que serán definidas en la segunda sección. Es aquí donde se definirá el id, nombre, instrumento y otras características de cada una de ellas.

A continuación se encuentra cada una de las distintas partes que componen la partitura. Una parte está definida por el tag <part> y está identificada por un id único que debe haber sido especificado en la sección anterior.

Cada parte se compone a su vez de al menos un compás, definido con la etiqueta <measure>. Cada compás va numerado con el atributo number indicando su posición en la partitura.

Cada compás contiene información sobre las notas que lo forman, y algunos de forma opcional, aunque recomendable en el primer compás de cada parte, contienen información adicional sobre la parte. Dichas características adicionales vienen especificadas dentro del tag <attribute> siendo las siguientes las más importantes:

- <divisions>. Indica el número de intervalos de tiempo que representan a la unidad de pulso.
- <key>. Especifica la tonalidad. Se define mediante los tags <fifths> y <mode> que indican el número de alteraciones en un intervalo [-7,7] dependiendo de si son bemoles o sostenidos, y el modo de la tonalidad (menor o mayor).
- <time>. Es el tipo de compás que viene indicado por los tags <beats> y <beat-type> que indican el número de pulsos y la figura por pulso del compás respectivamente.
- <clef>. Especifica la clave utilizada. Utiliza los tags <sign> para especificar el tipo de clave y <line> para indicar la línea que va a dar nombre la clave.

Por último, la parte más importante para representar una partitura son las notas. Las notas se especifican dentro de cada compás con el tag <note>. Para garantizar que la información de cada nota aparecida en la partitura es trasladada con total grado de fidelidad a MusicXML se utiliza un número elevado de atributos, siendo los siguientes los más importantes.

- <pitch>. Indica la altura de la nota. Está definido por los tags <step> que especifica el nombre de la nota y por tanto su frecuencia base, y por el tag <octave> que indica su altura tonal. Este atributo no se pone cuando lo que se represente sea únicamente un silencio.
- <duration>. Especifica el número de unidades de tiempo dura la nota.
- <voice>. Este tag permite indicar a qué voz pertenece la nota, de modo que podamos tener música polifónica en un mismo pentagrama.

- `<type>`. Con este type se indica el tipo de figura con la que se va a representar.
- `<stem>`. Representa la orientación de la plica si la figura la tuviera. Al igual que el pitch no se pone si la nota es un silencio.
- `<rest>`. Indica que la nota es un silencio.

Ejemplo

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC
    "-//Recordare//DTD MusicXML 2.0 Partwise//EN"
    "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="2.0">
  <part-list>
    <score-part id="P1">
      <part-name>Music</part-name>
    </score-part>
  </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>1</divisions>
        <key>
          <fifths>0</fifths>
        </key>
        <time>
          <beats>4</beats>
          <beat-type>4</beat-type>
        </time>
        <clef>
          <sign>G</sign>
          <line>2</line>
        </clef>
      </attributes>
      <note>
        <pitch>
          <step>C</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <type>whole</type>
      </note>
    </measure>
  </part>
</score-partwise>
```

Figura 2: Ejemplo Hello World en MusicXML

El ejemplo mostrado es el equivalente a un “Hello World!” de cualquier lenguaje de programación. La representación más sencilla posible en musicXML. En este caso el código representaría la siguiente información musical.

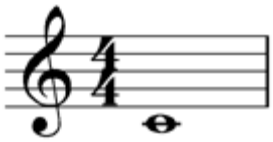


Figura 3: Representación gráfica del código anterior en MusicXML

Ventajas	Desventajas
En los últimos años se ha convertido en un estándar de la representación musical.	Gran cantidad de etiquetas hace difícil la lectura para una persona, fácil para una máquina.
Soportado por numerosos programas de edición musical	Necesarias demasiadas estructuras para representar poca información.
Permite la representación de gran cantidad de información musical y gráfica de la partitura.	Es necesario la utilización de un editor para la reproducción de las melodías.

Tabla 5: Ventajas y desventajas de MusicXML

3.2.1.2 **µousiké**

Se trata de un compilador MIDI escrito en Java y creado por el autor de estas líneas. Permite generar archivos MIDI utilizando una gramática capaz de describir una obra musical en ficheros de extensión *mke*.

La estructura de una obra musical escrita en µousiké se divide en dos partes bien diferenciadas: cabecera y cuerpo.

Cabecera

La cabecera indica el título, tempo, compás y tonalidad inicial de la obra. Además, añade un parámetro al final, llamado modo cuyos valores “grupo” y “orquesta” sirven para indicar la forma en que se tratarán los canales MIDI. La cabecera se especifica con la etiqueta 'cabecera' y su contenido entre llaves. Dispone de las palabras reservadas título, tempo, compás, tonalidad, y modo, las cuales tendrán asignado un valor. Este valor se definirá mediante el símbolo ‘=’ y los distintos valores que puedan tomar en su dominio.

A continuación, se muestra un pequeño ejemplo de una cabecera en µousiké:

```
cabecera{
     titulo = "Ejemplo1";
     tempo = (1/4, 120);
     compas = 4/4;
     tonalidad = mibM;
     modo = grupo;
}
```

Figura 4: Ejemplo de una cabecera escrita en µousiké

Cuerpo

El cuerpo se definirá mediante la sentencia ‘obra’ y encerrado entre llaves irán las distintas partituras definidas por la sentencia 'partitura', donde redactaremos la partitura por compases. Estas, pueden contener variables, repeticiones, cambios de parámetros de la obra (tonalidad, tempo, intensidad, etc.), repeticiones y la melodía en sí misma. Además, es donde se declaran las variables que se usen (fuera de la partitura serán globales, dentro de estas, locales y sólo visibles en la propia partitura). Podrá haber desde una a N partituras, una por cada instrumento o voz de la obra.

Las palabras reservadas para indicar la clave, el instrumento, y definir cambios de tonalidad y tiempo son: 'clave', 'instrumento, 'intensidad', 'tonalidad', ‘compas’ y 'tempo'. Las palabras ‘tonalidad’, ‘tempo’ y ‘compas’ sólo podrán utilizarse tras el símbolo reservado '||' de fin especial de compás. La intensidad se podrá cambiar en cualquier momento al igual que la clave.

Las posibilidades expresivas son ilimitadas, pudiendo representar todo tipo de agrupaciones, cambios de intensidad y tempo como crescendos, acelerandos, ritardandos, etc. Partituras rítmicas con instrumentos de percusión, codas, etc.
A continuación se muestra un ejemplo de dos partituras escritas en lenguaje µousiké para tratar de plasmar brevemente la gramática del programa.

Ejemplo sencillo de varias voces

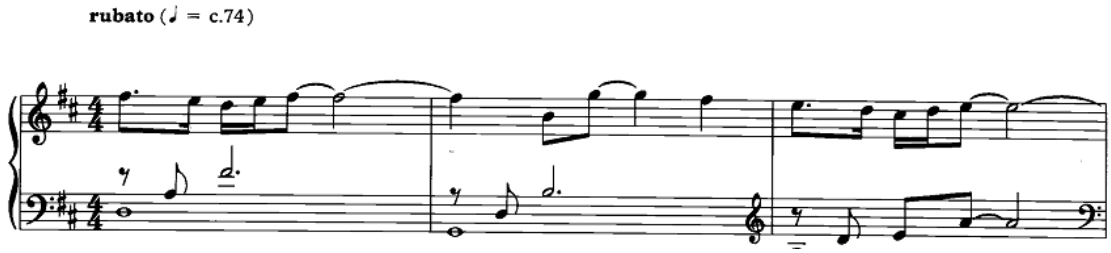


Figura 5: Ejemplo de varias voces en µousiké

```
cabecera{
  titulo = "Sentinel";//de Mike Oldfield.
  tempo = (1/4, 74);//duración de una negra.
  compas = 4/4;
  tonalidad = reM;
  modo = grupo;
}
obra{
  partitura{
    clave = c-sol; //no sería necesario al ser el de por defecto
    instrumento = 1; //no sería necesario al ser el de por
defecto
    intensidad = mf;
    melodía{
      fa'1/8., (mi', re', mi)1/16, L(|, fa', 1/8, 1/2)|
      L(fa', 1/4), B1/4, L(sol', 1/8, 1/4), fa'1/4|
      mi'1/8., (re', do', re')1/16, L(mi', 1/8, 1/2)|
    }
  }
  partitura{
    clave = c-fa;
    intensidad = mf;
    melodía{
      $1/8, la'1/8, re''1/2.|
      $1/8, re'1/8, si'1/2.|
      clave = c-sol;
      $1/8, (re, mi)1/8, L(la, 1/8, 1/2)|
      clave = c-fa-4;
    }
  }
  partitura{
    clave = fa4;
    intensidad = mf;
    melodía{
      re'1|
      sol1|
      la'1|
    }
  }
}
```

Figura 6: Código de ejemplo de varias voces en µousiké

Ejemplo sencillo con repeticiones

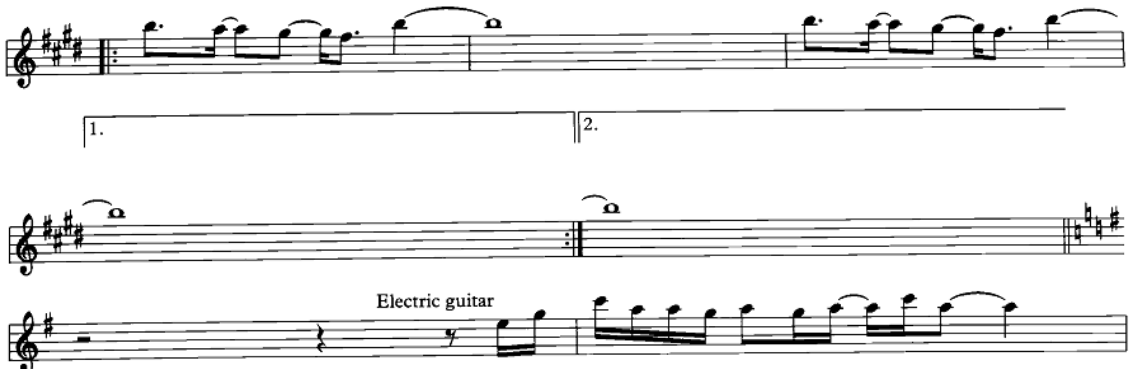


Figura 7: Ejemplo con repeticiones en µousiké

```
cabecera{
  titulo = "Sunjammer";
  tempo = (1/4, 86);
  compas = 4/4;
  tonalidad = miM;
  modo = grupo;
}
obra{
  partitura{
    intensidad = mf;
    melodía{
      ]:
        si'1/8., L(la', 1/16, 1/8), L(sol', 1/8, 1/16),
        re'1/8., L(si', 1/4 ,|)
        L(si', 1/1)| //parte común de la repetición con finales
        si'1/8., L(la', 1/16, 1/8), L(sol', 1/8,1/16),
        re'1/8., L(si', 1/4 ,|)
        1{
          L(si', 1/4, 1)| //primer final
        }
        2{
          L(si', 1/4, 1)|| // segundo final
          tonalidad = solM;
          intrumento = 30;
        }
      :[
        $1/2, $1/4, $1/8, (mi', sol')1/16|
        (do'', la', la', sol')1/16, la'1/8, sol'1/16,
        L(A'1/16, A'1/16), do''1/16,L(la'1/8, la'1/4)|
    }
  }
}
```

Figura 8: Código de ejemplo de repeticiones en µousiké

Palabras reservadas

Por último se muestra una tabla con las palabras reservadas del lenguaje

Palabra	*Valor*
Abrir corchete	[
Abrir llave	{
Abrir paréntesis	(
Abrir repetición compleja	]:
Accelerando	ace
Acorde	acorde
Alteraciones	# | x | b | bb | q
Apagar	off
Asignación	=
Cabecera	cabecera
Cambio compás	||
Cerrar corchete	]
Cerrar llave	}
Cerrar paréntesis	)
Cerrar repetición compleja	:[
Clave	clave
Coda	coda
Coma, separador	,
Comillas, para las cadenas	“ “
Compás	compas
Crescendo	cresc
Decrescendo	decresc
División	/
Doble repetición simple	%%
Encender	on
Etiqueta	FINE | CODA
Expresiones	! | > | _ | tr
Figuras	4/1 | 2/1 | 1/1 | 1/2 | 1/4 | 1/8 | 1/16 | 1/32 | 1/64 | 1/128
Fin declaración	;
Grupetos	dosillo | tresillo | cuatrillo | cinquillo | seisillo | septillo
Instrumento	instrumento
Intensidad	intensidad
Ligadura	L
Ligadura de expresión	lex
Melodía	melodía
Modo	modo
Motivo	motivo
Notas	do | re | mi | fa | sol | la | si | do’ | re’ | mi’ | fa’ | sol’ | la’ | si’ | do’’ | re’’ | mi" | fa" | sol" | la’’ | si" | k
Obra	obra
Partitura	partitura
Pedal	pedal
Puntillos	. | ..
Repetición simple	%

Palabra	*Valor*
Ritardando	rit
Separador compases	'|'
Sforzando	sfz
Silencio	$
Tempo	tempo
Tipo clave	c-sol | c-fa | c-fa-1 | c-fa-2 | c-fa-3 | c-fa-4 | c-sol+1 | c-sol+2 | c-sol+3 | c-percusion.
Tipo fin	D.C.fine | D.C.coda
Tipo intensidad	ppp | pp | p | mp | mf | f | ff | fff
Tipo modo	grupo | orquesta
Tipo tonalidad	doM | lam | solM | mim | reM | sim | laM | fa#m | miM | do#m | siM | sol#m | fa#M | re#m | do#M | la#m | faM | rem | sibM | solm | mibM | dom | labM | fam | rebM | sibm | dobM | labM.
Tipos grupetos	d | e | e1 | e2
Título	titulo
Tonalidad	tonalidad

Tabla 6 Palabras reservadas μousiké

Ventajas	Desventajas
Facilidad de escritura frente a MusicXML	Se trata de un lenguaje creado por el autor de estas líneas por lo que no cuenta con el soporte de ninguna plataforma de edición externa.
Permite especificar dinámicas para la expresividad que son compiladas a MIDI.	
Gramática más similar a una partitura	

Tabla 7: Ventajas y desventajas de μousiké

3.2.2 **Sistemas Expertos**

3.2.2.1 **CLIPS**

Clips [Cli] es una herramienta que provee un entorno de ejecución para Sistemas Expertos. Escrito en C y desarrollado por la NASA en la actualidad soporta Programación Lógica, Programación imperativa y Programación Orientada a Objetos.

Se hizo necesaria la creación de un lenguaje para Sistemas Expertos escrito en C dadas las dificultades existentes que ocasionaban los creados en LISP. Se basa en una ejecución no lineal construida sobre:

- Una base de hechos: que contiene un conjunto de literales o hechos que definen el estado en el que se encuentra la ejecución.
- Una base de reglas: que contiene un conjunto de reglas con un antecedente y un consecuente que se pueden disparar cuando el antecedente coincide con el estado actual de la ejecución.
- Una agenda: que contiene todas las reglas que cumplen sus antecedentes en el estado actual de la ejecución.

El orden de ejecución de las reglas viene definido por el motor de inferencia que se encarga de resolver el conjunto conflicto, que no es más que el conjunto de reglas equiparables en un instante de ejecución determinado.

Existen diversas estrategias para la resolución del conjunto conflicto, siendo las siguientes las que implementa CLIPS:

- Depth strategy: es la estrategia que implementa CLIPS por defecto. En ella las nuevas activaciones se sitúan por encima de las activaciones con igual prioridad.
- Breadh strategy: las nuevas activaciones se sitúan por debajo de las activaciones con igual prioridad.
- Simplicity strategy: las nuevas activaciones se sitúan por encima de las activaciones con mayor o igual especificidad.
- Complexity strategy: las nuevas activaciones se sitúan por debajo de las activaciones con mayor o igual especificidad.
- Random strategy: a cada activación se le asigna un número aleatorio para determinar su orden en la agenda. Siempre se le asigna el mismo número en diferentes ejecuciones.
- Lex strategy: ejecuta antes las reglas más nuevas.
- MEA strategy: Se aplica la misma estrategia que LEX pero mirando solo el primer patrón que equipara la regla. Si coincide se aplica LEX.

El motor de inferencia de CLIPS es de encadenamiento hacia adelante y procesa las reglas utilizando el algoritmo Rete.

```
(deftemplate plantilla
    (slot id
          (type INTEGER)
          (default 0)
    (slot atributo
          (type STRING)
          (default ?NONE))
)
```

Figura 9: Plantilla en CLIPS

La figura muestra la definición de un template en CLIPS. Para facilitar su comprensión, se puede considerar como el equivalente a una clase Java que únicamente contiene atributos. Define las características de un elemento que va a ser utilizado en el programa.

Los atributos pueden ser slot o multislot, dependiendo de si queremos que el atributo sea único o pueda aparecer de 0 a n veces. El tipo se define mediante la palabra reservada *type*. Además se puede limitar a un rango de valores un atributo mediante el uso de *allowed-values*, así como definir un valor por defecto con el uso de *default*.

```
(deffacts hecho
    (plantilla (id 1) (atributo "valor”))
)
```

Figura 10: Fact en CLIPS

La figura muestra un hecho en CLIPS. En este caso concreto se define un objeto del template plantilla dando valor a sus atributos.

```
(defrule ejemplo-regla
    (plantilla (id ?i) (atributo ?a))
    (test (< ?i 5))
    =>
    (assert (regla-anadida ?i))
)
```

Figura 11: Regla en CLIPS

En la figura se ve la definición de una regla como ejemplo. Está compuesta por dos partes: las condiciones que activan la regla, y la acción que ejecutan. En este caso la regla se activa si existe un hecho plantilla, cuyo id sea menor que 5, en cuyo caso añade un nuevo hecho.

Además de permitirnos definir templates, hechos y reglas, CLIPS nos da la posibilidad de crear reglas por medio de la palabra clave deffunction, así como la utilización de variables globales por medio de la palabra clave defglobal.

3.2.2.2 **CLIPSJNI**

CLIPSJNI [Cli] un interfaz para CLIPS escrito en Java que permite ejecutar comandos de la línea de comandos o *shell* de CLIPS, de esta forma, es posible ejecutar y modificar de forma sencilla código CLIPS desde Java, beneficiándonos del potencial que nos proporciona CLIPS. Esta herramienta ha permitido utilizar la potencia y estabilidad de CLIPS en un sistema Java de forma sencilla, evitando tener que recurrir a otras alternativas a priori de peor calidad.

Ventajas	Desventajas
Permite utilizar CLIPS en aplicaciones Java	Menos opciones que otros Sistemas Expertos en Java como Jess.
Una gran comunidad y soporte detrás	No existe una versión compilada para 64bits ni para sistemas Linux.
Gran rapidez de ejecución	
Software Libre	

Tabla 8: Ventajas y desventajas de CLIPSJNI

3.2.3 **Aprendizaje Basado en Instancias**

El Aprendizaje Basado en Instancias o IBL [IBL] es un método para la aproximación de funciones objetivo de valores continuos o discretos.

Se compone de dos fases, el entrenamiento y la generalización. El entrenamiento consiste en el almacenamiento de todo el conjunto de datos disponible mientras que la generalización consiste en la extracción de memoria de un conjunto de datos similares a un dato nuevo dado que son utilizados para clasificarlo, es por ello un sistema de Aprendizaje Perezoso (Lazy Learning).

La principal dificultad se encuentra en cómo definir el concepto de similitud, ya que varía según el problema y la representación de los datos. Otra dificultad reside en el coste de clasificación dado el tiempo necesario para realizar el cómputo de similitud con la base de ejemplos.

3.2.3.1 **K Nearest Neighbour**

Uno de los métodos de comparar la similitud entre las instancias es K Nearest Neighbour [Aha]. En este método todas las instancias se corresponden con puntos en un espacio de dimensión n (R^n). Los vecinos más cercanos de una instancia vienen dados en términos de la distancia euclídea de sus atributos:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (x_i[r] - x_j[r])^2}$$

Figura 12: Distancia euclídea

Si los atributos tienen diferentes rangos, se normalizan para poder tratarlos. En el caso de que los atributos tengan valores simbólicos hay que compararlos de alguna forma. El valor de K define el número de vecinos que se utilizan en la decisión.

Existen dos formas de elegir la salida dada una instancia x y sus k vecinos $x_1, \ldots, x_k$.

- Caso discreto: se asigna el valor más común de entre los k más cercanos: $f: R^n \rightarrow V, V = \{v_1, \ldots, v_s\}$

  tal que: $f(x) = arg_{v \epsilon V} máx \sum_{i=1}^{k} \delta(v, f(x_i))$

  donde: $\delta(a,b) = 1 \; si \; a = b, y \; \delta(a,b) = 0 \; en \; c.o.c$

- Caso continuo: se asigna el valor medio de entre los k más cercanos: $f: R^n \rightarrow R$

  tal que: $f(x) = \frac{\sum_{i=1}^{k} f(x_i)}{k}$

La elección del parámetro K se realiza por medio de prueba y error, ya que depende del problema y a priori no se conoce cuál funcionará mejor. Las regiones que se forman para k = 1 se conocen como regiones de Voronoi.

No siempre todos los atributos o características son igual de relevantes. Dependiendo del problema puede resultar interesante dar más valor a unas características que a otras. Para ello utilizamos la distancia euclídea ponderada:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} w_r (x_i[r] - x_j[r])^2}$$

Figura 13: Distancia euclídea ponderada

Decidir qué atributos son más relevantes que otros no es una tarea trivial. Su realización puede realizarse de forma manual por prueba y error pero resulta altamente ineficaz. Seguir una estrategia evolutiva es una mejor opción, aunque dependiendo del problema, encontrar una función de evaluación lo suficientemente buena puede resultar difícil.

La elección de esta técnica frente a otras viene dada por su similitud con el proceso de aprendizaje artístico del ser humano basado en aprender de la experiencia imitando o variando patrones percibidos en ellas. Se trata de un proceso sencillo de implementar donde la mayor dificultad aparece en ser capaces de calcular la similitud existente entre dos fragmentos musicales cualesquiera.

Ventajas	Desventajas
Facilidad de programación en entornos complejos.	Dificultad en la gestión de gran cantidad de datos.
Facilmente ampliable.	Difícil establecer similitud entre dos instancias según el problema.
No es necesaria supervisión.	Alta dependencia de la calidad de los datos.

Tabla 9: Ventajas y desventajas del Aprendizaje Automático basado en Instancias

Capítulo 4

Diseño

Este capítulo, dedicado al diseño del sistema de armonización, establece la arquitectura del sistema identificando sus componentes y el tránsito de información entre ellos.

4.1 Arquitectura del sistema

Antes de realizar una descripción detallada de los procesos involucrados en el sistema de armonización automática, se va a dar una visión general de la arquitectura del sistema, para pasar a describir en más profundidad cómo ha sido diseñado cada módulo del mismo.

El siguiente diagrama de bloques ilustra la arquitectura general del sistema desarrollado:

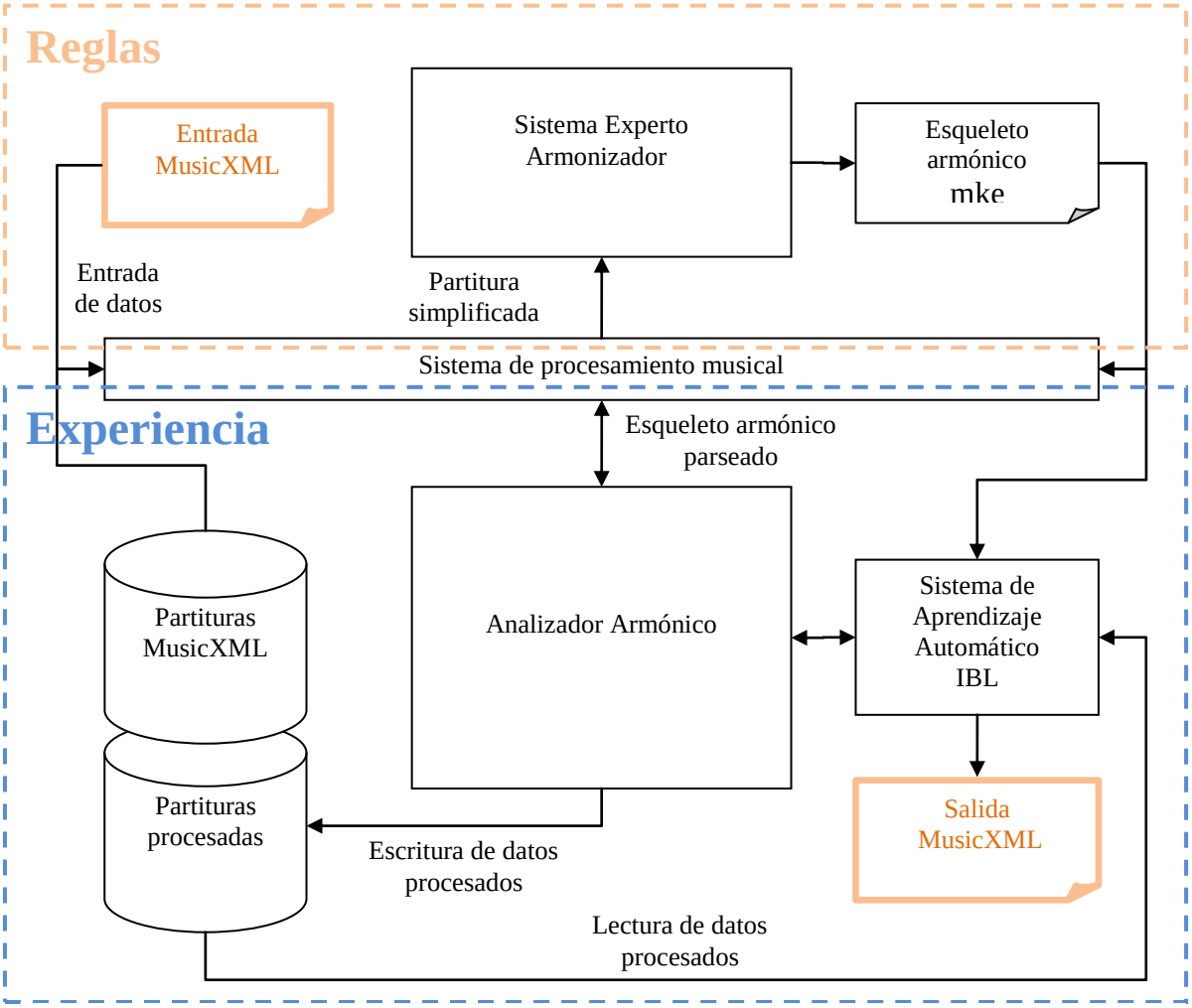


Figura 15: Arquitectura del sistema

Las partes fundamentales que componen esta arquitectura son:

- Los datos de entrada y de salida de los distintos módulos, representados con una esquina doblada. Estos son la entrada MusicXML del sistema, la salida *mke* del Sistema Experto y la salida final del sistema en MusicXML.
- A continuación se encuentran los datos utilizados en el proceso de análisis armónico y de aprendizaje automático. Se componen de un grueso de aproximadamente 400 partituras en MusicXML así como de dichos datos ya procesados para su uso inmediato.
- Ascendiendo en importancia se encuentra el sistema de procesamiento musical. Este sistema es el que contiene la distinta información musical para realizar la lectura y escritura de los distintos lenguajes de representación musical así como realizar una correcta manipulación de los elementos que componen las partituras.

Por último, los grandes bloques que forman el sistema son el Sistema Experto armonizador, encargado de generar el esqueleto armónico sobre la que se sustentará la obra valiéndose para ello del conocimiento musical adquirido, y el sistema de aprendizaje automático basado en instancias y compuesto por:

- El sistema de análisis armónico genera a partir de partituras en formato MusicXML una serie de datos en bruto y procesados que serán utilizados por el sistema de aprendizaje automático, valiéndose para ello técnicas matemáticas y conocimiento teórico musical.
- El sistema de aprendizaje automático basado en instancias compara cada parte del esqueleto armónico modificando cada línea melódica del acompañamiento con el fin de que el resultado final sea más natural y menos mecánico.

4.2 **Sistema de procesamiento musical**

Después de haber visto de forma general la arquitectura del sistema, se va a entrar con más detalle en cada uno de los módulos que lo forman. El primero de ellos es el sistema de procesamiento musical, encargado de la lectura y escritura de los distintos ficheros en instancias que el sistema pueda manejar, así como es el núcleo central que se comunica con cada uno de los módulos.

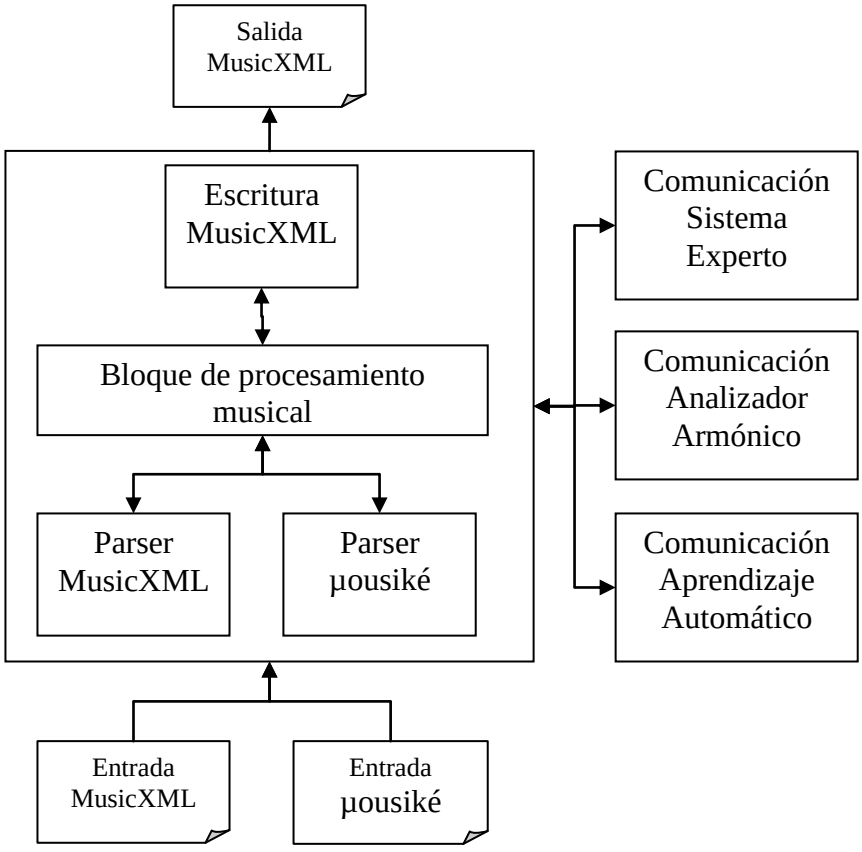


Figura 16: Arquitectura del sistema de procesamiento musical

El módulo cuenta con un sistema de parseo que convierte los ficheros MusicXML y µousiké en objetos *Score* que el bloque de procesamiento musical puede manejar para realizar diversos cálculos y operaciones musicales sobre ellos. Este sistema se comunica con los demás funcionando como un sistema de E/S del resto de módulos.

4.3 **Sistema Experto armonizador**

El Sistema Experto armonizador es el eje central del sistema, siendo el módulo más importante puesto que de él depende la calidad de la obra al ser el responsable de generar el esqueleto armónico de la misma.

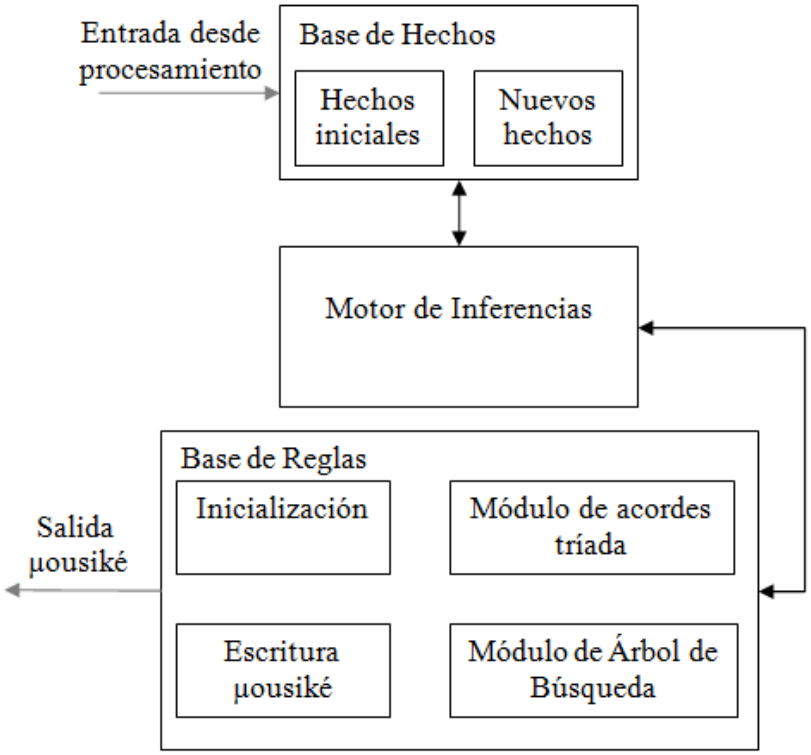


Figura 17: Arquitectura del Sistema Experto armonizador

Al tratarse de un Sistema Experto está formado por tres módulos: la base de hechos, la base de reglas y el motor de inferencias.

La base de hechos está formada por los hechos iniciales que serán utilizados por el sistema, para por medio de las reglas generar nuevos hechos que permitan alcanzar una solución.

La base de reglas ha sido dividida en módulos, haciendo el sistema modulable permitiendo añadir contenido en un futuro con mayor facilidad. Este sistema se compone principalmente por cuatro submódulos:

- Inicialización: encargado de inicializar el sistema, cargando los templates y realizando los cálculos previos para que el sistema pueda comenzar a funcionar.
- Módulo de acordes tríada: responsable de la generación de los distintos acordes que pueden ser parte de la obra final.
- Módulo de árbol de búsqueda: crea la estructura en forma de árbol de búsqueda para buscar la solución al problema.
- Escritura μousiké: escribe la solución en formato μousiké que será parseada por el sistema de procesamiento musical.

4.4 **Sistema de Aprendizaje Automático**

El Sistema de Aprendizaje Automático está formado por dos componentes: el analizador armónico y el algoritmo de aprendizaje automático basado en instancias. A continuación se detallan estos componentes.

4.4.1 **Analizador Armónico**

El siguiente módulo es el analizador armónico. Este sistema es el encargado de analizar las partituras de ejemplo para el sistema así como la partitura que se desea armonizar.

Para agilizar el proceso de armonización, el análisis de datos del banco de ejemplos es opcional, permitiendo generarlo únicamente cuando se inserten partituras nuevas.

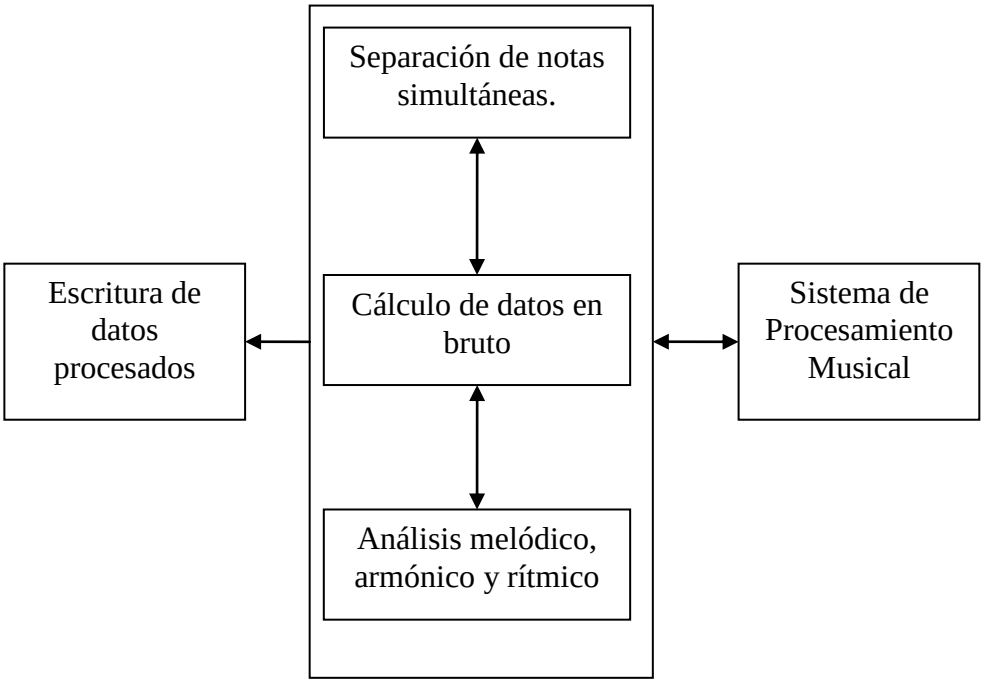


Figura 18: Arquitectura del analizador armónico

El analizador se compone de tres submódulos:

- Separación de notas simultáneas: encargado de procesar la partitura para extraer cada grupo de notas que suenen de forma simultánea.
- Cálculo de datos en bruto: realiza los cálculos del Set Theory Prime así como extrae información del contexto musical de cada grupo de notas pasadas por el módulo anterior.
- Análisis melódico, armónico y rítmico: procesa los datos en bruto generando patrones rítmicos, armónicos y rítmicos así como otro tipo de datos musicales basados en el contexto armónico y melódico.

El sistema cuenta además con un módulo encargado de la escritura de los datos analizados para que estén disponibles sin necesidad de volver a analizarlos.

Del mismo modo, el analizador se comunica con el sistema de procesamiento musical que hace las veces de comunicador con el exterior.

4.4.2 **Módulo de Aprendizaje Automático basado en Instancias**

El último módulo del sistema es el Sistema de Aprendizaje Automático. Es el encargado de realizar modificaciones en el esqueleto armónico utilizando para ello ejemplos previamente analizados como guía.

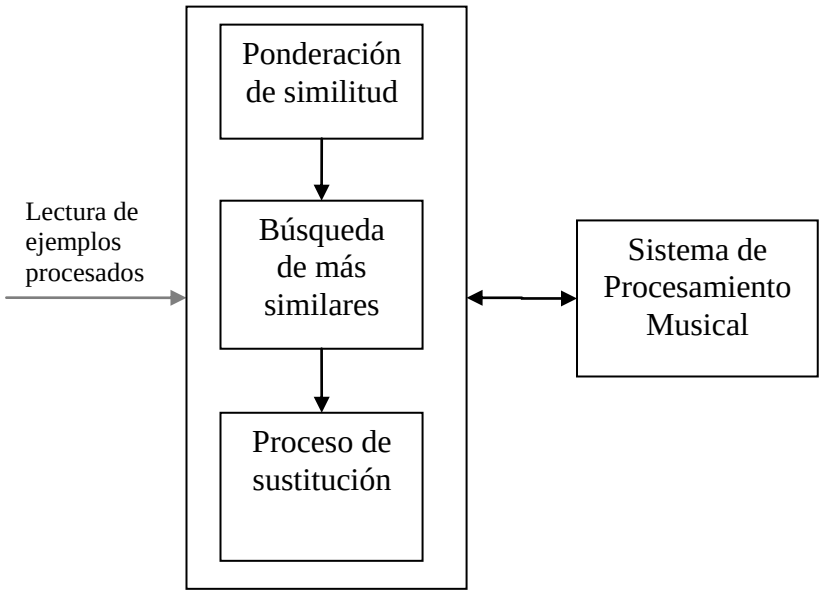


Figura 19: Arquitectura del sistema de Aprendizaje Automático

El núcleo central se comunica con el sistema de procesamiento musical y recibe por entrada los datos de ejemplos procesados según la técnica de agrupamiento a utilizar en el proceso de análisis. Se compone de tres módulos:

- Ponderación de similitud: encargado de ponderar las similitudes existentes entre dos fragmentos de partituras. Los atributos cuentan con distintos valores de ponderación dada su diferente importancia, y algunos únicamente se tienen en cuenta si la comparación es admisible, esto es, las notas encajan en la partitura sin afectar completamente a su armonía.
- Búsqueda de más similares: realiza una búsqueda de aquellos ejemplos que han obtenido la puntuación más alta almacenándolos en una estructura de datos.
- Proceso de sustitución: modifica la partitura utilizando la información contenida en el ejemplo mejor valorado. En caso de no haber ningún ejemplo similar no se realiza ninguna sustitución en ese fragmento con el objetivo de no empeorar el resultado.

4.5 **Diagrama de flujo del sistema**

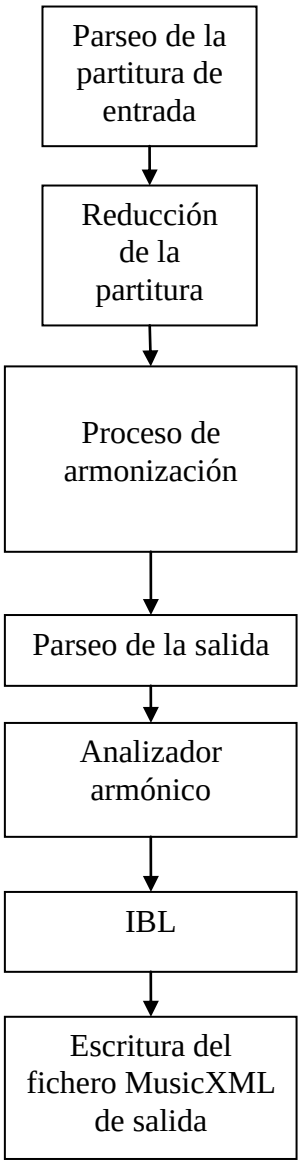


Figura 20: Flujo general del sistema

La figura muestra el flujo general del sistema, no entrando a describir en detalle el flujo de los módulos más importantes: armonizador, analizador armónico y aprendizaje automático, que serán explicados con más detalle más adelante. El resto de procesos son parsers y transformaciones musicales.

El flujo del sistema comienza con el parseo de la partitura de entrada en objetos que puedan ser manipulados por el sistema.

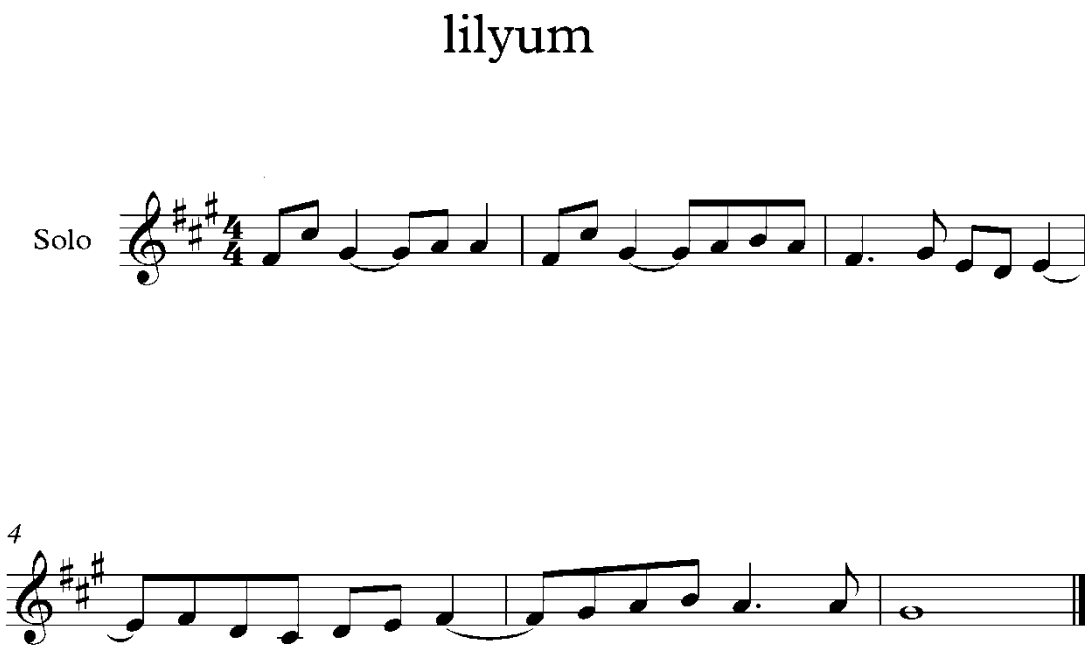


Figura 21: Entrada del sistema

Una vez se tiene instanciada la partitura en objetos se procede a realizar una simplificación de la misma que sirva como base para el esqueleto armónico. Esta simplificación se realiza creando una nueva melodía donde únicamente se muestras las notas que estaban en un ataque de pulso en la partitura original.

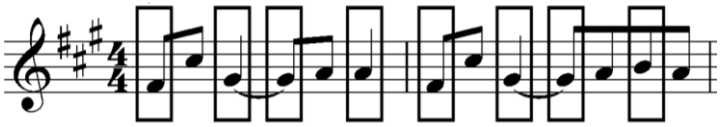


Figura 22: Fragmento melódico a simplificar

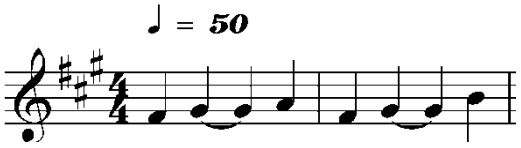


Figura 23: Melodía simplificada

Esta melodía es la entrada del proceso de armonización. A partir de esta simplificación se obtendrá un esqueleto armónico más sólido que si se intentase armonizar cada nota de la partitura.

4.5.1 **Flujo del Sistema Experto armonizador**

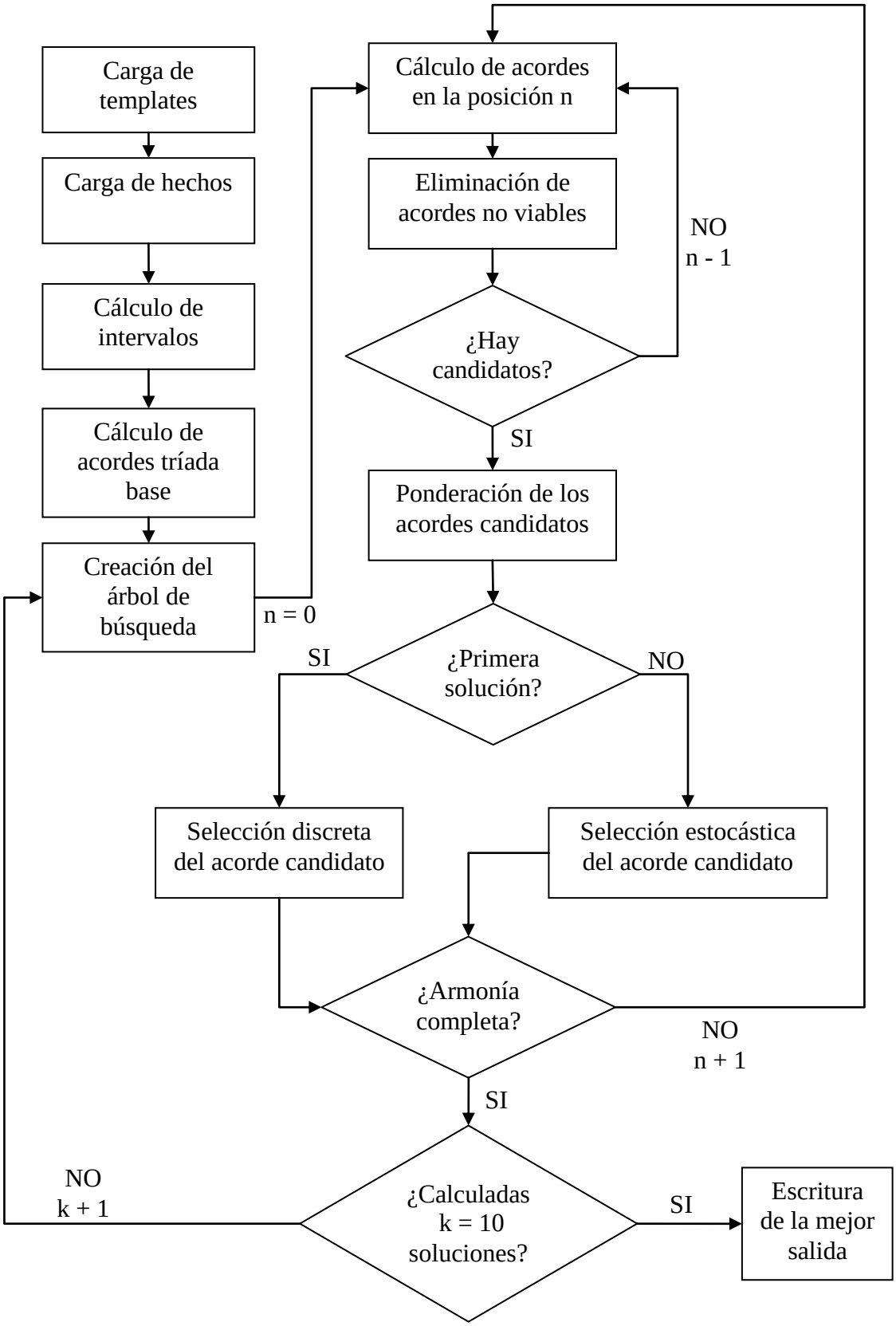


Figura 24: Diagrama de flujo del proceso de armonización

El proceso comienza con la carga del archivo de templates que contiene las plantillas necesarias para almacenar la información que necesita el sistema para generar el esqueleto armónico.

A continuación se definen los datos de entrada. Para que el sistema sea capaz de procesarla, la partitura tiene que ser definida en un conjunto de hechos como el compás, la tonalidad, el tempo, el título, los instrumentos asignados a cada una de las voces[1] y la propia melodía a armonizar junto con el número total de notas de la melodía.

El formato de representación de la melodía consiste en una tupla que contiene la posición que ocupa en la partitura en orden secuencial, el timbre sonoro y la duración de la nota. Así pues, dado el siguiente fragmento:

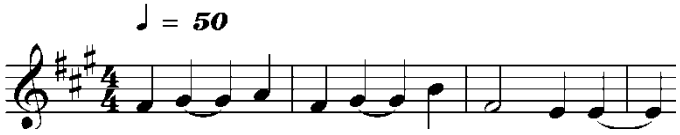


Figura 25: Fragmento de entrada al sistema

Los datos de entrada serían los siguientes:

Compás	
Numerador	Figura
4	1/4

Tabla 10: Datos de entrada SE - Compás

Tonalidad		
Nombre	Modo	Num-alteraciones
Fa♯	Menor	+3

Tabla 11: Datos de entrada SE - Tonalidad

Tempo	
Figura	Velocidad
¼	50

Tabla 12: Datos de entrada SE - Tempo

Título
Lilyum

Tabla 13: Datos de entrada SE - Título

Melodía			
Posición	Sonido	Octava	Figura
1	Fa♯	5	1/4
2	Sol♯	5	1/2
3	La	5	1/4
4	Fa♯	5	1/4
5	Sol♯	5	1/2

[1] Esto se debe a que el fichero de salida es un fichero μousiké que por su condición de compilador MIDI requiere de esta información.

Melodía			
Posición	Sonido	Octava	Figura
6	Si	5	1/4
7	Fa♯	5	1/2
8	Mi	5	1/4
9	Mi	5	1/2

Tabla 14: Datos de entrada SE - Melodía

Tras la inserción de los datos de entrada, se añaden en forma de hechos el conocimiento básico para empezar a calcular una armonía, esto es, la distancia que existe entre cada nota de forma natural para poder calcular los distintos tipos de intervalos, conocimiento necesario para la construcción de acordes.

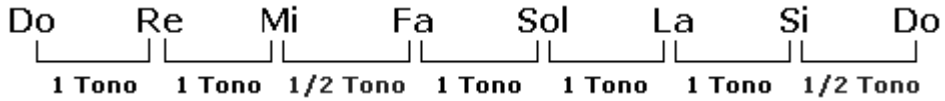


Figura 26: Distancia tonal entre notas naturales

Con esta información se calculan todos los intervalos para cada par de notas, información imprescindible para poder realizar cualquier cálculo armónico. Por último, dado que se conoce la tonalidad, se crea una relación unívoca entre grado tonal y nota perteneciente a la tonalidad de modo que se tenga el conocimiento de la función armónica de cada nota de la melodía.

Funciones armónicas	
Grado	Nota
I	Fa♯
II	Sol♯
III	La
IV	Si
V	Do♯
VI	Re
VII	Mi

Tabla 15: Funciones armónicas de Fa♯ menor

Esta información es utilizada para que el sistema sepa la función armónica de cada acorde durante la construcción del esqueleto armónico.

Una vez se han definido los distintos intervalos comienza el proceso de cálculo de acordes tríada base para cada posición (nota) de la partitura a armonizar.

El proceso es muy sencillo dado que sabemos que las notas que componen un acorde tríada tienen que estar en disposición de terceras entre ellas y que las notas que los forman deben pertenecer a la tonalidad de la obra.

Dada esta naturaleza del acorde tríada, por cada nota de soprano a armonizar puede haber un total de tres acordes tríada base en función de la posición que ocupe la nota: fundamental, tercera o quinta del acorde.

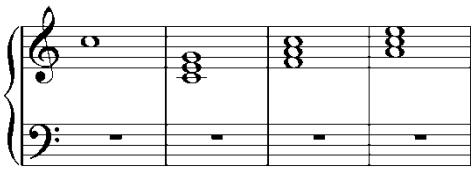


Figura 27: Obtención de los posibles acordes tríada base

Esto proporciona al sistema la información básica para poder empezar a construir el esqueleto armónico. Por cada nota de la melodía se calculan los posibles acordes tríada base y su posición en la partitura por lo que el número de acordes base viene dado por la fórmula.

$$numero\ Acordes = 3 * numero\ de\ notas$$

Una vez calculados todos los acordes tríada base de todas las notas que componen la melodía, el sistema procede a crear la estructura del árbol de búsqueda para empezar a calcular soluciones al problema.

El primer paso es calcular las construcciones de los acordes en su disposición coral asignando cada nota a una de las voces y realizando las duplicaciones necesarias.

Este proceso, pese a contar con las notas base del acorde tríada, posee una mayor complejidad dado el número de factores a tener en cuenta. Hay que destacar la dificultad que tiene procesar un lenguaje como es el musical dados los problemas que provocan las enarmonías tal y como se comenta en la explicación teórica musical.

Para cada acorde tríada hay que generar todos los posibles acordes en estado fundamental y en primera inversión.

- Acordes de quinta o tríada en estado fundamental: en esta posición usamos las tres notas del acorde tríada duplicando la fundamental. Tanto las voces de soprano como la voz del bajo van a ser fijas, la primera por restricción de la melodía, y el bajo por ser el que indica el tipo de acorde. Esto hace que solo queden dos posibles notas para cada una de las voces restantes (contralto y tenor). Dependiendo de si la distancia entre cada voz es la menor posible o si por el contrario la distancia entre cada voz no es la mínima se tendrán distintos tipos de acorde: cerrados o abiertos.

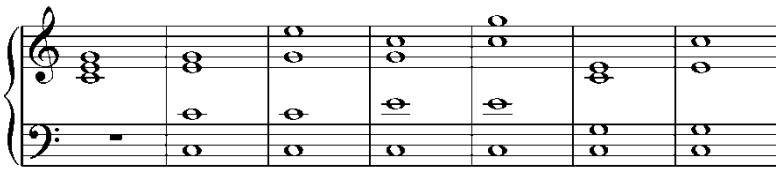


Figura 28: Acordes posibles a partir de un acorde tríada base

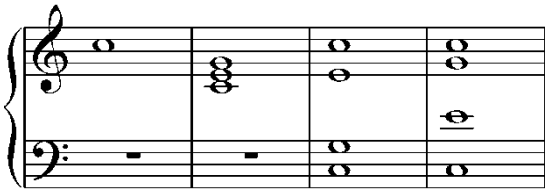


Figura 29: Generación de acordes en estado fundamental válidos

Como puede verse en la figura, a partir del acorde tríada base formado por las notas *do mi sol* y con la restricción de que la voz soprano debe ser un *do en tercer espacio* sólo se pueden obtener dos posiciones válidas que no incumplan ninguna regla armónica, mientras que si no se restringe la voz del soprano este número se multiplica por tres.

- Acordes de sexta o tríada en primera inversión: en este tipo de acorde se va a poder duplicar tanto la fundamental como la tercera y la quinta del acorde según la situación.

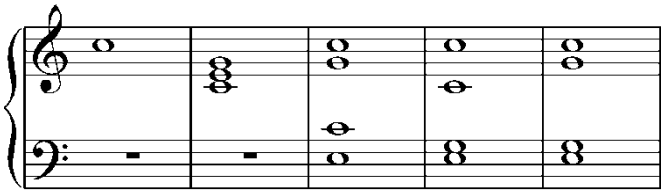


Figura 30: Generación de acordes de sexta válidos

Debido a un mayor número de duplicaciones posibles en los acordes dispuestos en primera inversión el número de resultados bien formado es mayor que en los acordes dispuestos en estado fundamental.

Tras haber calculado todos los posibles acordes para una posición determinada se procede a eliminar todos aquellos que incumplan alguna regla armónica o melódica con respecto al acorde inmediatamente anterior. Los acordes que sobrevivan a este proceso serán los acordes candidatos para ocupar esa posición en la solución final. En caso de que ninguno cumpla las restricciones del problema se vuelve a la posición inmediatamente anterior eligiéndose un acorde diferente.

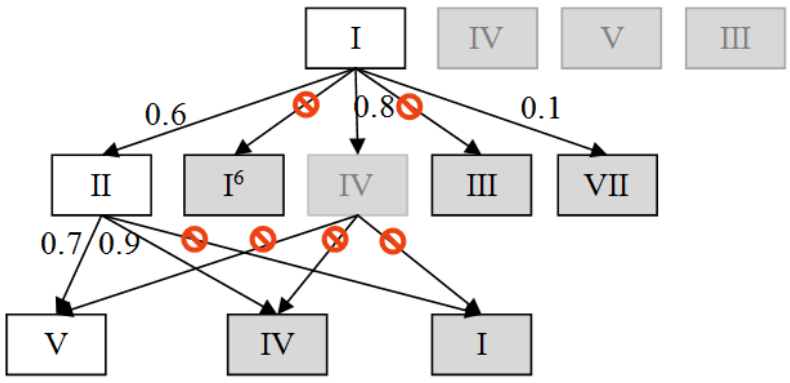


Figura 31: Representación gráfica de un fragmento del árbol de búsqueda

Si existen candidatos para la posición actual se procede a ponderar lo apropiados que son para dicha posición teniendo en cuenta diversos atributos como el tipo de progresión armónica que forma con el acorde anterior, su función tonal o el intervalo formado entre voces entre otros.

El candidato elegido dependerá del proceso de selección que se use. Existen dos tipos: discreto y estocástico. El primero de ellos es un proceso discreto que elige siempre el mejor candidato mientras que el segundo utiliza una ruleta en la que se introduce proporcionalmente a la nota los candidatos y se elige uno al azar, teniendo más posibilidades de salir aquel con una puntuación mayor. Este segundo proceso se realiza para dar cierta aleatoriedad al proceso, ya que elegir siempre la mejor opción no tiene por qué dar como resultado la mejor armonización para toda la obra. Esta es quizás la mayor dificultad del proceso de armonización, y es que no se cuenta con un método que garantice encontrar la mejor solución posible ni un método de ponderación que asegure que una solución sea objetivamente mejor que otra aunque se cuentan con métodos que dan una idea aproximada de cuánto mejor puede ser una solución con respecto a otra.

El proceso de armonización consta de un total de diez cálculos de soluciones, la primera de ellas discreta y nueve estocásticas. La solución elegida para ser escrita en formato *mke* es aquella que de una mejor ponderación global.

4.5.2 **Flujo del Sistema de Aprendizaje Automático**

4.5.2.1 **Analizador armónico**

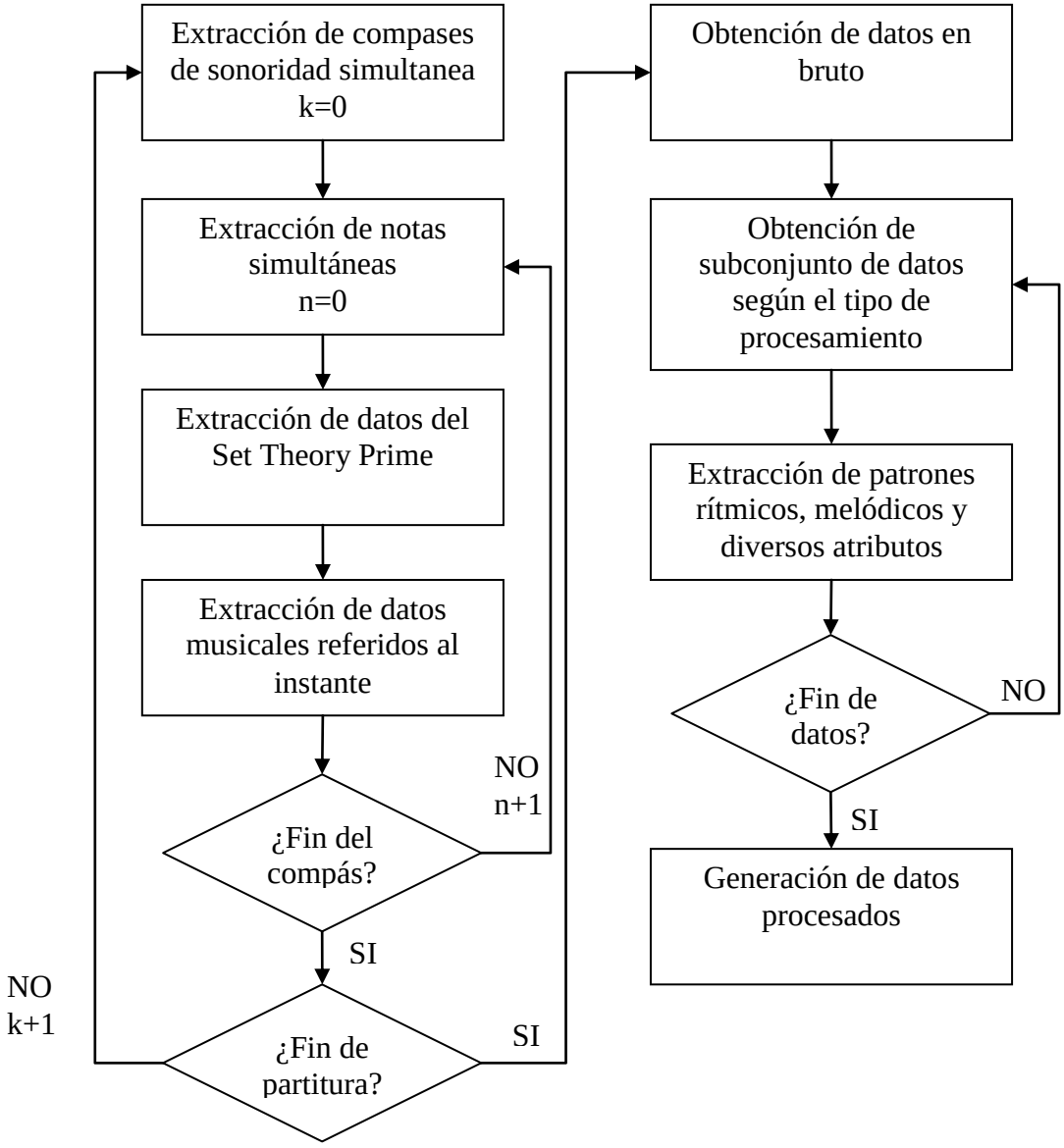


Figura 32: Diagrama de flujo del Proceso de análisis armónico

El análisis armónico comienza una vez se ha parseado la salida del proceso armónico sirviendo de entrada al proceso de análisis. Puesto que se trata de un análisis armónico es imprescindible analizar todos los sonidos que se produzcan de forma simultánea. Para ello la partitura se analiza con la extracción del compás actual de cada voz.

El siguiente paso es la extracción de notas simultáneas en ese fragmento. Se calculan todas las superposiciones de modo que queden cubiertos todos los cambios tal y como se muestra en la siguiente figura.

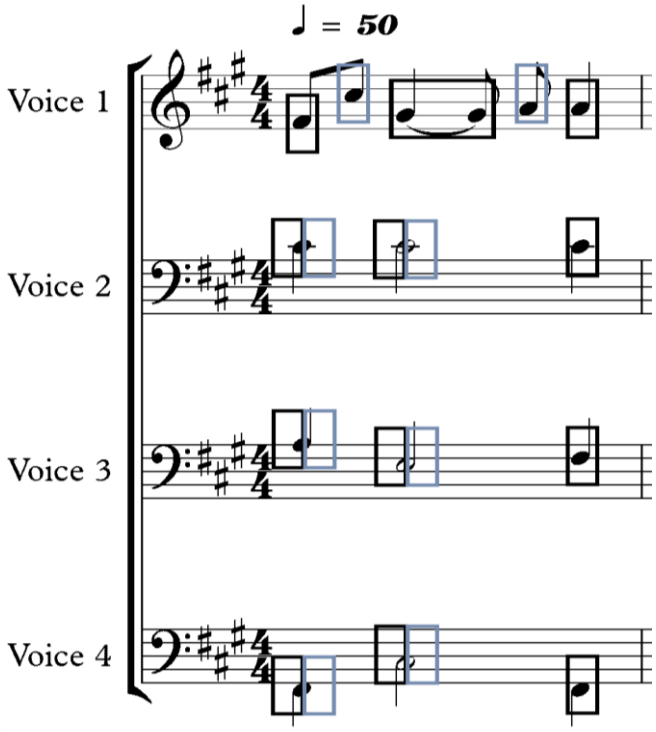


Figura 33: Obtención de notas simultáneas

Se puede observar cómo la nota de duración mínima es la que especifica la duración del conjunto. De este modo una misma nota puede repartir su duración entre varios conjuntos. Cada conjunto de notas es analizado y calculado su Set Theory Prime.

El *Set Theory Primer for Music* de Larry Solomon [Sol] es un método matemático que permite deducir el tipo de acorde que forman un conjunto de notas clasificándolo en su forma Prime[2]. El proceso de cálculo de la forma Prime de un acorde se divide en varias etapas:

- Se obtiene el valor numérico de las notas que componen el acorde según la siguiente tabla, teniendo en cuenta que dos notas enarmónicas tienen el mismo valor y que no se permiten duplicidades.

C	C♯	D	D♯	E	F	F♯	G	G♯	A	A♯	B
B♯	D ♭		E ♭	F ♭		G ♭		A ♭		B ♭	C ♭
0	1	2	3	4	5	6	7	8	9	10	11 ♭

Tabla 16: Valor de cada nota en el Set Theory Primer

[2] Forma mínima estable de un acorde que permite su clasificación.

Notas de entrada	E	G	B	D
Valor numérico	4	7	11	2

Tabla 17: Ejemplo de asignación de valores

- El siguiente paso consiste en ordenar ascendentemente el valor numérico obtenido al sustituir en la tabla, a esta secuencia se le conoce como *pins*.

pins	2	4	7	11

Tabla 18: Pins

- A continuación calculamos el *div*, que no es más que la secuencia formada por las restas de las notas en el índice siguiente menos la nota en el índice actual módulo 12.

pins	2	4	7	11
div	(4-2)	(7-4)	(11-7)	(2-11)
	2	3	4	3

Tabla 19: Cálculo del div a partir del pins

- La *normal-form* del acorde viene dada por la secuencia cíclica que comienza por la nota con el índice siguiente a la nota de más valor del *div* de la secuencia del *pins*. Esta forma representa el acorde en su forma más compacta.

pins	2	4	7	11
div	2	3	4	3
normal-form	11	2	4	7

Tabla 20: Cálculo de la normal-form de un acorde

- El último paso para calcular la forma prime del acorde consiste en ordenar el *div* de forma cíclica comenzando por la nota inmediatamente siguiente a la de valor más alto, para a continuación comenzando en 0, ir almacenando el acumulado de sumar a cada casilla la siguiente.

div	2	3	4	3
div ordenado	3	2	3	4
prime	0	0 + 3	(0 + 3) + 2	((0 + 3) + 2) + 3
	0	3	5	8

Tabla 21: Cálculo de la forma prime de un acorde

La forma prime del acorde le identifica independientemente de la tonalidad en la que se encuentre, lo que es de gran utilidad en un sistema como el que se plantea. Existe un gran número de formas prime, pero por tratarse de un proyecto a cuatro voces únicamente vamos a tener formas prime de hasta cuatro elementos.

Valores Prime		
Prime	Tipo de acorde	Fundamental
[0]	Unísono	primer elemento

Valores Prime		
Prime	Tipo de acorde	Fundamental
[0][1]	Segunda menor	primer elemento
[0][2]	Segunda mayor	primer elemento
[0][3]	Tercera menor	primer elemento
[0][4]	Tercera mayor	primer elemento
[0][5]	Cuarta justa	segundo elemento
[0][6]	Cuarta aumentada	segundo elemento
[0][1][2]	Acorde tríada cromático	primer elemento
[0][1][3]	Acorde tríada frigio	primer elemento
[0][1][4]	Acorde segunda-tercera menor	primer elemento
[0][1][5]	Acorde de séptima mayor incomplete	segundo elemento
[0][1][6]	Acorde de cuarta justa – cuarta aumentada	segundo elemento
[0][2][3]	Acorde de segundas mayors	primer elemento.
[0][2][4]	Acorde de segundas mayores.	primer elemento.
[0][2][5]	Acorde de séptima menor incompleto	segundo elemento.
[0][2][6]	Acorde de séptima de dominante incompleto | Sexta italiana	segundo elemento
[0][2][7]	Acorde tríada de cuartas.	segundo elemento
[0][3][4]	Acorde de tercera	primer elemento
[0][3][5]	Acorde de séptima menor incompleto	tercer elemento
[0][3][6]	Acorde disminuido	primer elemento
[0][3][7]	Acorde triada perfecto menor	primer elemento
[0][4][5]	Acorde de séptima mayor incompleto	tercer elemento
[0][4][6]	Acorde incompleto séptima half-dim	tercer elemento
[0][4][7]	Acorde triada perfecto mayor	primer elemento
[0][4][8]	Acorde aumentado	primer elemento
[0][5][6]	Acorde de cuarta aumentada, cuarta justa	tercer elemento
[0][1][3][4]	Tetra acorde espejado 2m,2M,2m	primer elemento
[0][1][3][5]	Tetra acorde frigio	primer elemento
[0][1][3][6]	Acorde disminuido con segunda menor	primer elemento
[0][1][3][7]	Acorde all-interval 3	primer elemento
[0][1][4][5]	Tetra acorde arábigo menor	primer elemento

Valores Prime		
Prime	Tipo de acorde	Fundamental
[0][1][4][8]	Tetra acorde menor aumentado	segundo elemento
[0][1][5][6]	Tetra acorde 2m,4j,2m	primer elemento
[0][1][5][8]	Acorde de séptima mayor	segundo elemento
[0][2][3][5]	Acorde espejado 2M, 2m, 2M	primer elemento
[0][2][3][6]	Tetra acorde armónico menor 2m,2M, 2A	primer elemento
[0][2][3][7]	Acorde menor con segunda menor	primer elemento
[0][2][4][5]	Tetra acorde mayor 2M,2M,2m	primer elemento
[0][2][4][6]	Tetra acorde 2M,2M,2M	primer elemento
[0][2][4][7]	Acorde mayor con segunda menor	primer elemento
[0][2][4][8]	Acorde de séptima aumentada	segundo elemento
[0][2][5][7]	Tetra acorde de cuartas	segundo elemento
[0][2][5][8]	Acorde de séptima medio disminuido	segundo elemento
[0][3][4][6]	Tetra acorde mayor disminuido	primer elemento
[0][3][4][8]	Tetra acorde mayor aumentado	tercer elemento
[0][3][5][6	Acorde disminuido con cuarta justa	primer elemento
[0][3][5][7]	Acorde menor con cuarta justa	primer elemento
[0][3][5][8]	Acorde de séptima menor	tercer elemento
[0][3][6][8]	Acorde de séptima de dominante	cuarto elemento
[0][3][6][9]	Acorde de séptima disminuida	primer elemento
[0][4][5][7]	Acorde mayor con cuarta justa	primer elemento
[0][4][6][7]	Acorde all-interval 4	primer elemento

Tabla 22: Valores de las formas prime

Los elementos que aparecen en la tabla son todas las posibles formas prime que pueden aparecer en una armonización como la que se pretende abarcar en este trabajo. Es, por tanto, una forma sencilla de realizar un análisis armónico que permita comparar de algún modo estructuras armónicas diferentes.

Para no perder la perspectiva del contexto se almacenan, a parte de los cálculos procedentes del *Set Theory Prime*, gran cantidad de datos referidos a lo que está ocurriendo en ese instante, desde la duración total de cada nota, si está siendo o no atacada, información sobre el compás o la tonalidad entre otros.

Este proceso se repite hasta finalizar la partitura obteniendo así los datos en bruto con toda la información de cierta relevancia para ser procesada. Antes de ser procesados, los datos deben ser agrupados siguiendo algún criterio de los disponibles:

- Por número de notas: este agrupamiento de datos es el más sencillo de todos los implementados. Simplemente se agrupan en conjuntos de *n* filas de datos en bruto.
- Por tiempo de pulso: se agrupan los datos en conjuntos de *n* pulsos para ser procesados.
- Por tiempo de pulso y ataque de las notas: se agrupan los datos en *n* conjuntos que contengan los datos del comienzo de pulso y el siguiente donde todas las notas sean atacadas en ese instante.

Los datos agrupados se procesan para hacerlos independientes de la tonalidad de modo que sean útiles para comparar con cualquier partitura independientemente de su tonalidad. La información que se extrae en este proceso son patrones rítmicos, armónicos y melódicos así como diversa información contextual sobre ataques de notas o cruzamiento de voces entre otros.

4.5.2.2 **Módulo de Aprendizaje Automático basado en Instancias**

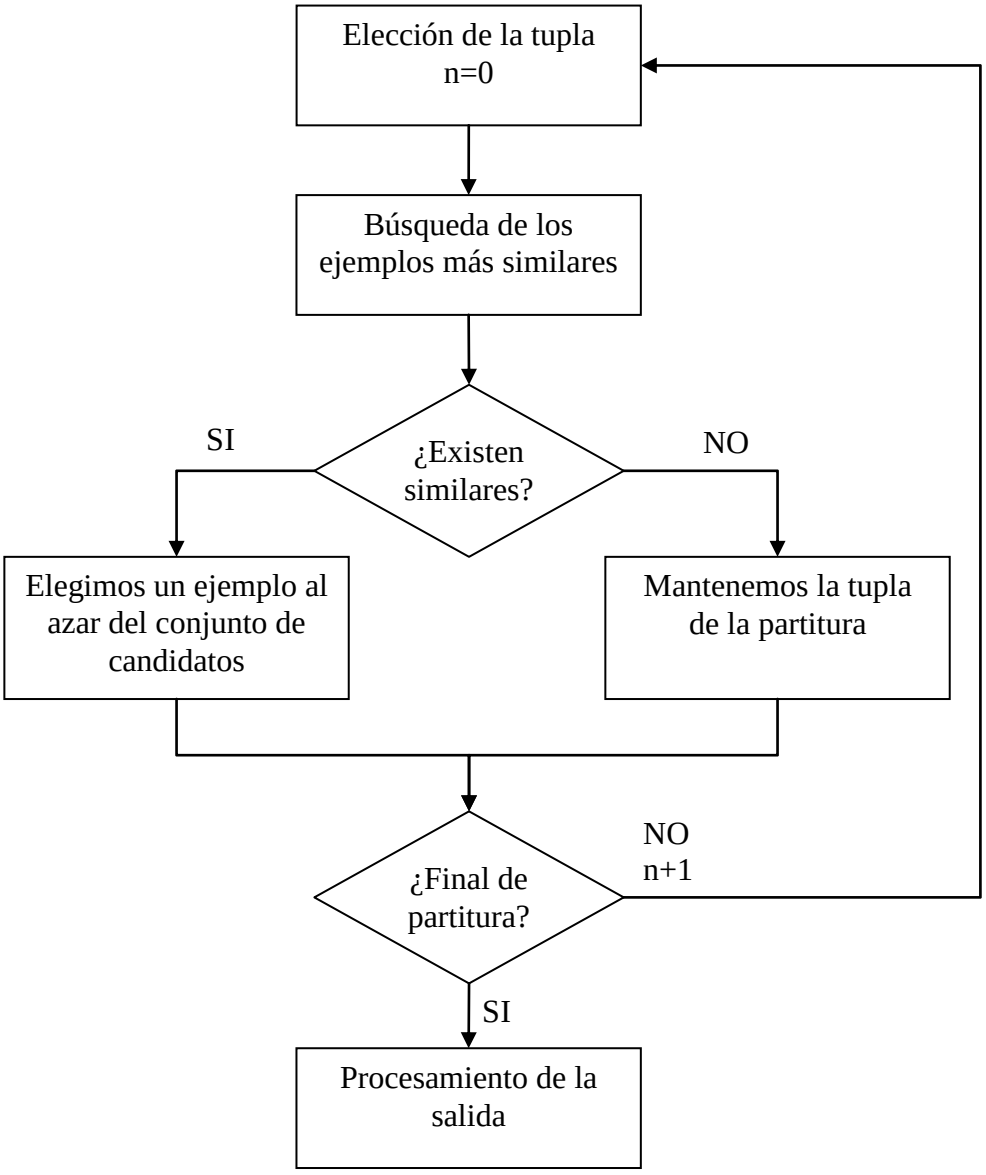


Figura 34: Diagrama de flujo del proceso de Aprendizaje Automático

El proceso de Aprendizaje Automático es el encargado de modificar el esqueleto armónico de la obra con el fin de hacer cada voz independiente de las demás sin afectar al conjunto de la armonía. El proceso consiste en buscar un ejemplo similar y utilizar su información para modificar la partitura. Para ello el sistema compara cada tupla del análisis armónico de la partitura con un conjunto de datos de partituras ya procesadas.

Independientemente del método elegido el proceso de Aprendizaje Automático basado en Instancias se divide en tres etapas:

- Ponderación del grado de similitud entre cada fragmento analizado de la partitura a modificar y el conjunto de datos de ejemplos.
- Selección de los fragmentos más similares.
- Modificación del fragmento original por el más similar.

4.5.2.3 **Ponderación de similitud**

La parte más importante del sistema de Aprendizaje Automático basado en Instancias se encuentra en la ponderación del grado de similitud entre dos instancias, un proceso cuya dificultad varía dependiendo del problema a tratar, siendo en este caso concreto un proceso de gran complejidad.

El planteamiento inicial es simple, se comparan los atributos de la instancia del conjunto de ejemplos con la instancia perteneciente a la partitura a modificar. Dado que en este trabajo los atributos con simbólicos se deben decidir alguna manera de ponderar el grado de similitud de forma numérica, en este caso se ha optado por asignar un valor de una unidad en caso de que dos atributos sean iguales y un valor de cero en caso contrario.

Dada la complejidad del dominio algunos atributos tienen la capacidad de descartar completamente una instancia como candidata otorgándole automáticamente un grado de similitud negativo. Este es el caso de las notas iniciales del acorde, donde para evitar disonancias en el ataque se obliga a que coincidan con la instancia de entrada, del mismo modo que se obliga a que ambas instancias tengan una misma duración total de modo que encaje en la partitura cuando se produzca la sustitución. Puesto que un *DataSet* contiene la información del conjunto y del *DataSet* anterior, para facilitar que las progresiones melódicas sean suaves, la ponderación total del grado de similitud comprendida entre [0,1] es repartida equitativamente entre ambas.

Del mismo modo se ha asignado un potenciador a cada atributo ya que la importancia de cada atributo en la comparación es diferente, por poner un ejemplo la importancia de la melodía o el ritmo es mayor que el cruzamiento de voces.

La comparación para el patrón actual se realiza únicamente sobre la línea melódica dada la complejidad para comparar una armonía completa. La armonía es ponderada únicamente utilizando los datos proporcionados por el *Set Theory Prime* dado que la comparación de los patrones rítmicos de cada una de las voces resultaría inútil. Por el contrario, la comparación con el patrón previo es más exhaustiva, realizando una comparación completa de todas las voces.

4.5.2.4 **Búsqueda de más similares**

Este proceso es realmente sencillo, simplemente se encarga de almacenar en una estructura de datos aquellas instancias cuyo grado de similitud sea mayor. La comparación se realiza dando distinto valor a cada atributo en función de su importancia.

Podemos encontrarnos dos casos, uno en el que nos encontremos un número n de instancias con un mismo grado de similitud o que no encontremos ninguna similar.

En el primer caso se elige una instancia al azar entre todas para ser utilizada en el proceso de sustitución. Por el contrario, en el segundo caso no se utiliza ninguna instancia en el proceso de sustitución manteniendo la instancia original.

4.5.2.5 **Proceso de sustitución**

El proceso de sustitución consiste en la modificación de las tres voces de acompañamiento generadas por el Sistema Experto armonizador utilizando para ello las instancias ponderadas cuyos grados de similitud son más elevados con el fragmento a modificar. Se han implementado dos métodos diferentes para llevar a cabo esta tarea:

- Sustitución rítmica: se obtienen los patrones rítmicos existentes en la instancia de ejemplo y se sustituye el acompañamiento armónico de la melodía en ese fragmento para mostrar dicho patrón rítmico, manteniendo el timbre de cada nota.

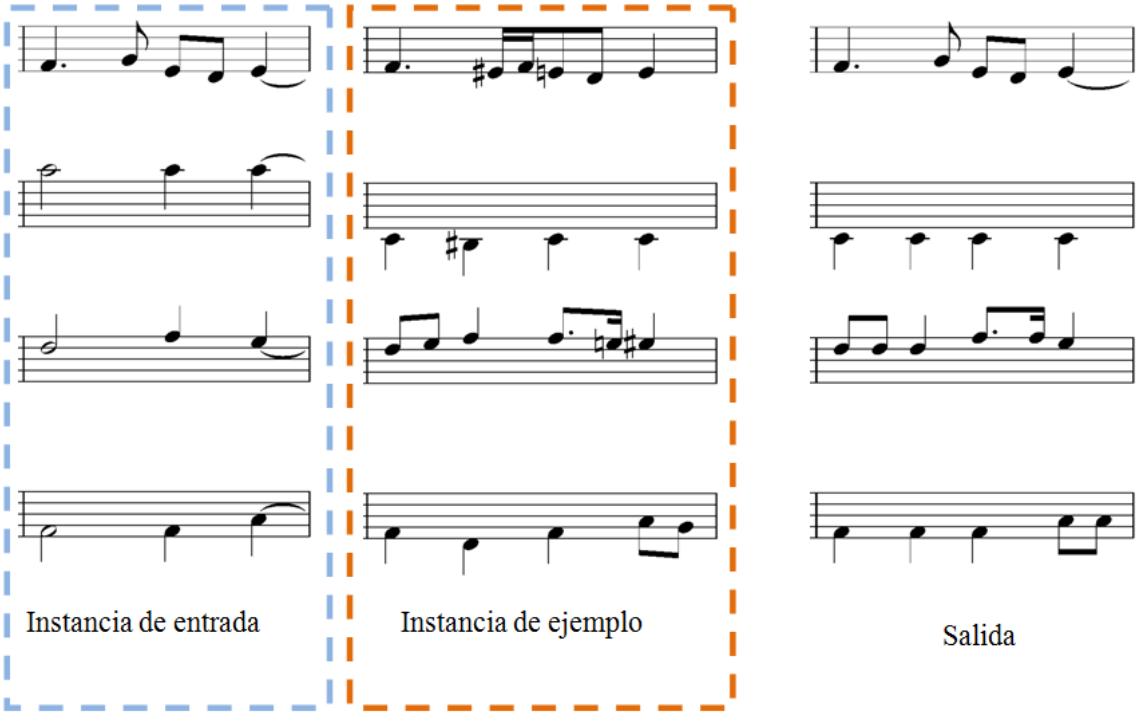


Figura 35: Sustitución rítmica

Tal y como se aprecia en la figura la salida es el resultado de la sustitución del patrón rítmico de la instancia de ejemplo seleccionada como la más similar manteniendo las notas de la instancia de entrada[3]. El resultado es un esqueleto armónico idéntico a la entrada pero que gracias a la sustitución del patrón rítmico adquiere una sonoridad más rica, donde cada voz funciona como un ente independiente.

[3] Destacar que el segundo pentagrama de Instancia de entrada está escrita en clave de fa, por lo que la nota que aparece es un do, la misma que aparece en la salida escrita en clave de sol.

- Sustitución melódico-rítmica: se modifica el acompañamiento armónico de modo que cada voz utilice el patrón rítmico y melódico existente en la instancia más similar al fragmento a modificar.

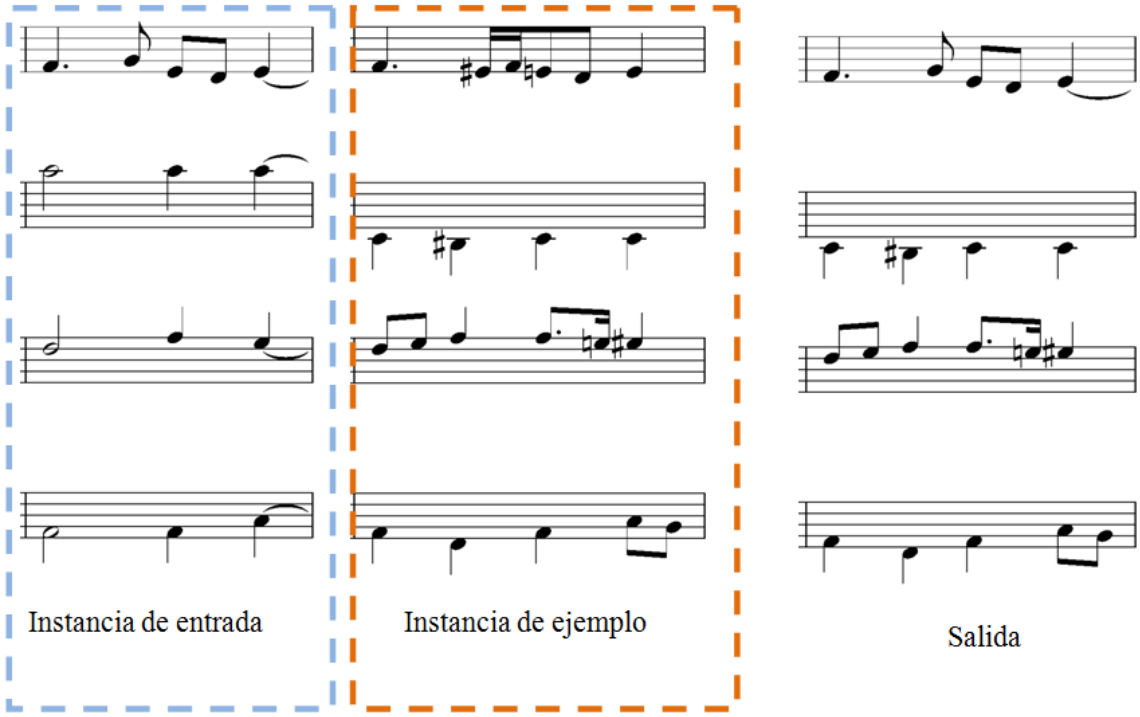


Figura 36: Sustitución rítmico-melódica

En este caso se puede apreciar como la sustitución es completa a excepción de la melodía original. Esto provoca que el esqueleto armónico sea modificado por lo que el resultado dependerá en gran medida de lo buena que sea la función de ponderación de similitud entre dos instancias. Pese a todo resulta de un gran interés dado que permite una mayor diferenciación entre cada una de las voces que componen la obra.

Tras haber modificado todos los fragmentos se procesa la información para transformarla nuevamente en un fichero MusicXML bien formateado.

Capítulo 5

Implementación

5.1 Introducción

En este capítulo se va a describir la implementación del sistema, entrando en detalle en las partes que lo forman. Se irá describiendo cada módulo descrito en el capítulo anterior individualmente, mostrando las dependencias entre módulos en aquellas partes que existan.

5.2 Sistema de procesamiento musical

Tal y como se comentó en el capítulo anterior, este módulo está formado por los distintos parsers, tanto para MusicXML como para μousiké, instanciándolos en objetos Java. Es por ello que la estructura que define una partitura cobre especial importancia en este apartado.

5.2.1 **Score**

Se ha optado por realizar un diseño en forma de árbol ya que es la forma más natural de representar una partitura, además de ser un diseño que facilita el proceso de escritura en MusicXML.

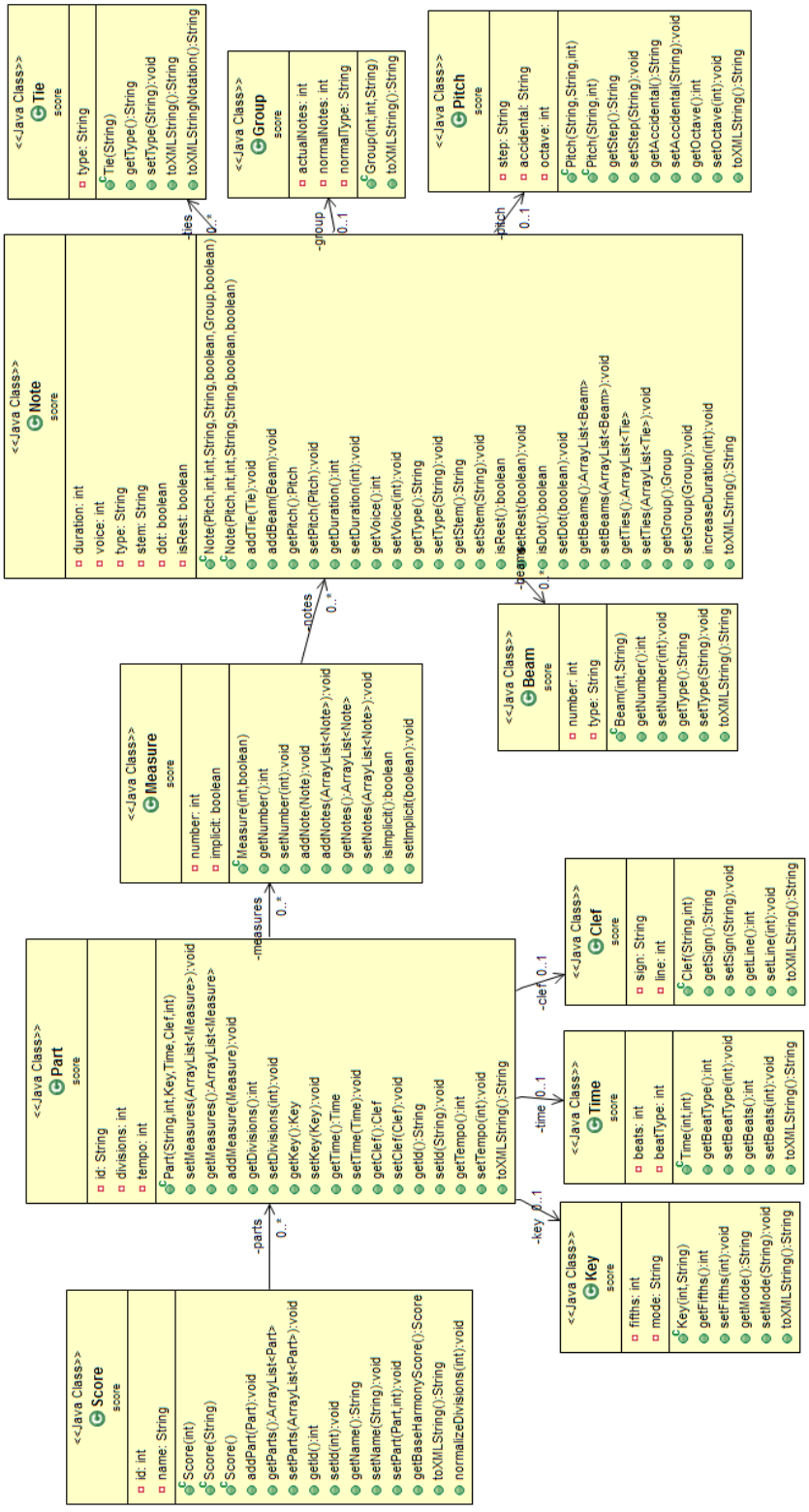


Figura 37: Diagrama de clases de una partitura

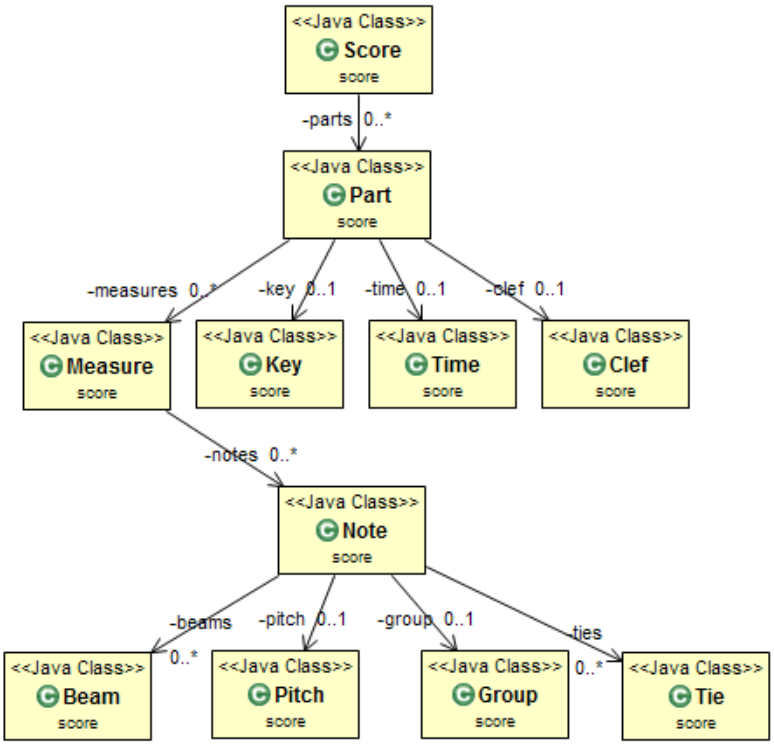


Figura 38: Estructura de una partitura

Puede observarse en el diagrama cómo una partitura, definida por el objeto *Score* está formada por varias partes llamadas *Part*, un identificador y un nombre. Un objeto *Part* está compuesto por varios compases definidos por la clase *Measure* así como por una serie de atributos como la tonalidad, el compás y la clave, definidos por las clases *Key*, *Time* y *Clef* respectivamente junto con un identificador y el número de divisiones que definen un pulso. En este proyecto el número máximo de partes que puede tener una partitura es de cuatro para representar cada una de las cuatro voces que forman una armonía.

Un compás a su vez está formado por una lista de notas representadas por la clase *Note*, así como por dos atributos que indica el número de compás y si este es implícito o no.

Cada una de las notas que componen un compás está definida por su duración, si es silencio o no, la dirección de su plica en caso de tenerla, el tipo de figura que la representa, la voz a la que pertenece, si cuenta con un puntillo así como su timbre, definido por la clase *Pitch*, si es un tipo de agrupación especificado por la clase *Group*, o si cuenta con ligaduras de duración o de expresión por medio de las clases *Tie* y *Beam* respectivamente.

Todas las clases que forman esta estructura cuentan con un método llamado *toXML*String que convierte en cadena de caracteres con formato MusicXML el contenido de cada clase y sus hijas, facilitando a posteriori la escritura de ficheros MusicXML.

5.2.2 **Parser**

El sistema es capaz de manejar dos tipos de lenguajes de representación musical: MusicXML y µousiké, siendo el primero el formato de entrada y salida del sistema al estar soportado por la mayoría de programas de edición existentes hoy día, y el segundo un lenguaje de representación intermedio dada su facilidad de escritura y lectura frente a MusicXML. Para su manejo se han creado dos parsers: *MusicXMLParser* y *MousikeParser* que instancian en un objeto tipo *Score* la partitura escrita en los ficheros.

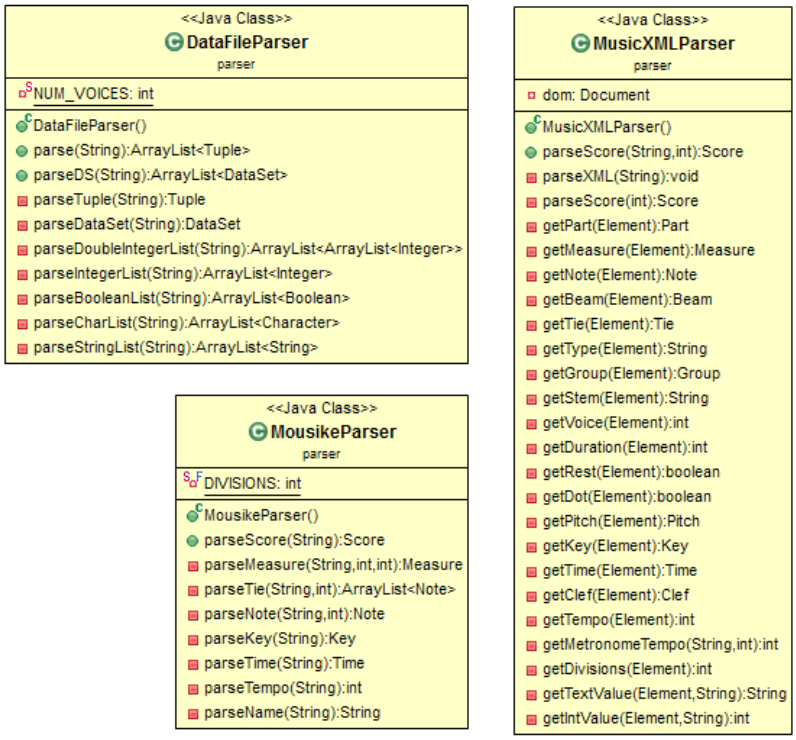


Figura 39: Diagrama de clases del paquete de parseadores

Existe además un tercer parseador, *DataFileParser* encargado de instanciar en objetos *Tuple* el contenido de los ficheros de datos resultantes del proceso de análisis armónico.

5.2.3 **Operators**

Dentro del paquete Operators encontramos la clase *Music*. Esta clase está formada únicamente por métodos estáticos para el manejo de operaciones musicales que resultan de gran utilidad en un proyecto de estas características ya que son operaciones que hay que repetir constantemente pues forman parte del propio lenguaje musical.

Así pues, se encuentran métodos para el cálculo de tonalidades, alteraciones por tonalidad, valor numérico de las notas, traducciones entre la forma internacional y la

española de representación musical o métodos para el manejo de enarmonías, cálculos de movimientos entre voces, cálculos armónicos y cálculos melódicos.

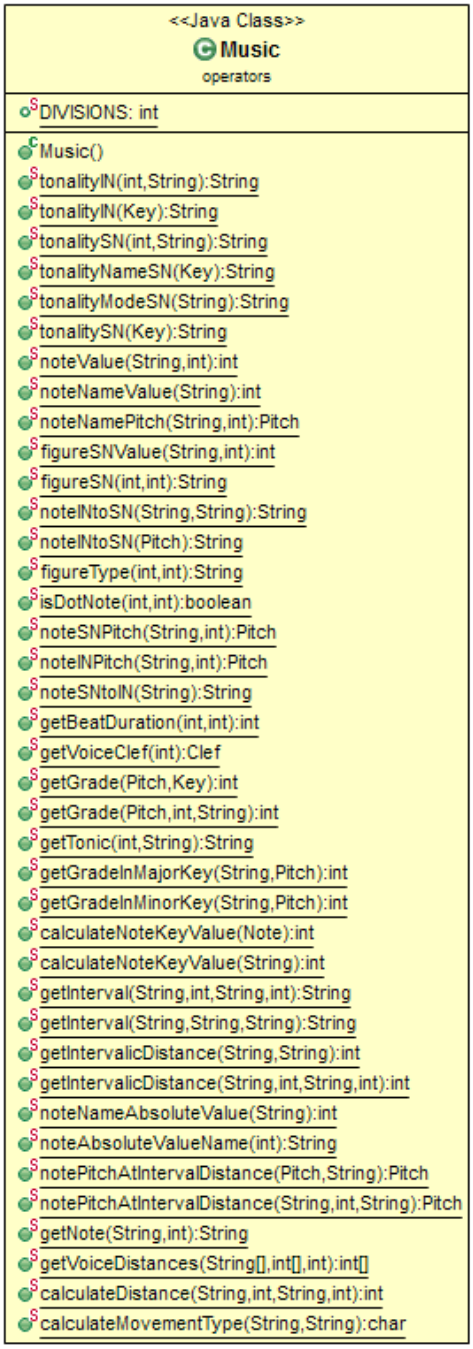


Figura 40: Diagrama de la clase Music

5.3 **Sistema Experto armonizador**

El Sistema Experto armonizador se compone de dos partes: un Sistema Experto escrito en CLIPS y una parte Java encargada de pasar los datos y comenzar el proceso mediante CLIPSJNI.

La clase Java encargada de la gestión del Sistema Experto en CLIPS es *HarmonizingSystem* que cuenta con la partitura de entrada como atributo. La clase posee un único método *createHarmony*, responsable de la ejecución del Sistema Experto que creará un fichero µousiké con el esqueleto armónico de la obra.

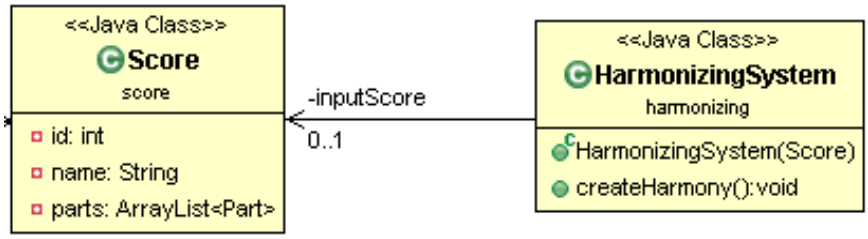


Figura 41: Diagrama de clases del sistema de armonización

El Sistema Experto CLIPS está compuesto por varios módulos para facilitar la ampliación y mantenibilidad del mismo. En este trabajo en concreto cuenta con seis archivos diferentes, cada uno con una función específica en el sistema:

- Templates: contiene la definición de las distintas plantillas de las que el sistema hará uso para almacenar y manipular la información.
- Melodía: este fichero es el único que se genera nuevo en cada ejecución ya que contiene la información de la partitura de entrada transformada en hechos que el sistema pueda manejar.
- Hechos iniciales: inserta en el sistema los primeros hechos y reglas necesarios para comenzar el proceso de armonización tales como la distancia interválica entre notas naturales, la generación de los distintos intervalos o la especificación de la función tonal de cada nota de la tonalidad entre otros.
- Acordes tríada: es el responsable de la generación de los distintos acordes tríada soportados por el sistema así como el encargado de verificar el cumplimiento de las restricciones armónicas en los enlaces producidos entre distintos acordes.
- Estructura armónica: implementa el árbol de búsqueda que permite encontrar la solución. Cuenta con diversas reglas y funciones para el manejo de la estructura en forma de árbol y toma de decisiones.
- Salida µousiké: contiene las reglas que permiten la escritura de la solución en un fichero µousiké.

Las partes más importantes para entender el sistema son la especificación de los distintos templates y las reglas que manipulan los hechos para alcanzar la solución.

5.3.1 **Templates**

En este apartado se muestran las distintas plantillas que se definen en el Sistema Experto para almacenar la información del proceso de armonización.

- Nota: representa

PLANTILLA NOTA		
ATRIBUTOS	**VALOR**	**Descripción**
Posición	Entero	Posición dentro de la obra
Sonido	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Sonido que representa una nota e indica su frecuencia fundamental
Octava	Entero entre 0 y 10	Octava a la que pertenece la nota
Figura	2/1, 2/1., 1/1, 1/1., 1/2, 1/2., 1/4, 1/4., 1/8, 1/8., 1/16, 1/16., 1/32, 1/32.	Duración de la nota

Tabla 23 Plantilla Nota

PLANTILLA ACORDE		
Atributos	**Valor**	**Descripción**
Id	Entero	Identificador unívoco del acorde
Posición	Entero	Posición de la obra a la que pertenece
Grado	I, II, III, IV, V, VI, VII	Grado del acorde tríada que define
Tipo-posicion	Abierto, cerrado	Tipo de colocación de las notas
Estado	Fundamental, inversión-1, inversión-2	Estado del acorde.
Sonido-soprano	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Sonido que representa una nota e indica su frecuencia fundamental
Octava-soprano	Entero entre 0 y 10	Octava a la que pertenece la nota
Figura-soprano	2/1, 2/1., 1/1, 1/1., 1/2, 1/2., 1/4, 1/4., 1/8, 1/8., 1/16, 1/16., 1/32, 1/32.	Duración de la nota
Sonido-contralto	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Sonido que representa una nota e indica su frecuencia fundamental
Octava-	Entero entre 0 y 10	Octava a la que pertenece la nota

PLANTILLA ACORDE		
Atributos	**Valor**	**Descripción**
contralto		
Figura-contralto	2/1, 2/1., 1/1, 1/1., 1/2, 1/2., 1/4, 1/4., 1/8, 1/8., 1/16, 1/16., 1/32, 1/32.	Duración de la nota
Sonido-tenor	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Sonido que representa una nota e indica su frecuencia fundamental
Octava-tenor	Entero entre 0 y 10	Octava a la que pertenece la nota
Figura-tenor	2/1, 2/1., 1/1, 1/1., 1/2, 1/2., 1/4, 1/4., 1/8, 1/8., 1/16, 1/16., 1/32, 1/32.	Duración de la nota
Sonido-bajo	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Sonido que representa una nota e indica su frecuencia fundamental
Octava-bajo	Entero entre 0 y 10	Octava a la que pertenece la nota
Figura-bajo	2/1, 2/1., 1/1, 1/1., 1/2, 1/2., 1/4, 1/4., 1/8, 1/8., 1/16, 1/16., 1/32, 1/32.	Duración de la nota

Tabla 24 Plantilla Acorde

PLANTILLA INTERVALO		
Atributos	**Valor**	**Descripción**
Nombre	Entero	Indica el nombre del intervalo: segunda, tercera, cuarta, quinta, sexta, séptima u octava.
Tipo	d, m, M, A, J	Indica el tipo del intervalo: disminuido, menor, mayor, aumentado o justo.
Nota-origen	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Nota desde la que se empieza a contar el intervalo.
Nota-destino	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Nota final del intervalo

Tabla 25 Plantilla Intervalo

PLANTILLA TONALIDAD		
Atributos	**Valor**	**Descripción**
Nombre	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Indica el nombre de la tonalidad
Modo	Mayor, menor	Indica el modo de la tonalidad que especifica la relación entre tonos y semitonos en los grados de la escala.

PLANTILLA TONALIDAD		
Atributos	**Valor**	**Descripción**
Num-alteraciones	Entero [-7, 7]	Numero positivo indica el número de sostenidos de la tonalidad. Número negativo indica el número de bemoles de la tonalidad.

Tabla 26 Plantilla Tonalidad

PLANTILLA COMPAS		
Atributos	**Valor**	**Descripción**
Numerador	Entero [2-21]	Indica la duración del compás en función del número de figuras.
Figura	1/1, 1/1., 1/2, 1/2., 1/4, 1/4., 1/8, 1/8., 1/16, 1/16., 1/32, 1/32.	Indica la figura base de ese compás.

Tabla 27 Plantilla Compas

PLANTILLA GRADO TONALIDAD		
Atributos	**Valor**	**Descripción**
Grado	I, II, III, IV, V, VI, VII	Indica el grado que ocupa la nota en la tonalida.
Función	Tónica, supertónica, mediante, subdominante, dominante, superdominante, sensible, subtónica.	Indica la función del grado.
Nota	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Indica la nota que tiene esa función en la tonalidad.

Tabla 28 Plantilla Grado Tonalidad

PLANTILLA VOZ		
Atributos	**Valor**	**Descripción**
Tipo	Soprano, contralto, tenor, bajo	Identifica la voz melódica.
Sonido-max	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Indica el sonido máximo permitido para la voz.
Octava-max	Entero [0-10]	Indica la octava del sonido máximo.
Sonido-min	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Indica el sonido mínimo permitido para la voz.
Octava-min	Entero [0-10]	Indica la octava del sonido mínimo.

Tabla 29 Plantilla Voz

PLANTILLA CADENCIA		
Atributos	**Valor**	**Descripción**
Tipo	Soprano, contralto, tenor, bajo	Identifica la voz melódica.
Sonido-max	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Indica el sonido máximo permitido para la voz.
Octava-max	Entero [0-10]	Indica la octava del sonido máximo.
Sonido-min	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Indica el sonido mínimo permitido para la voz.
Octava-min	Entero [0-10]	Indica la octava del sonido mínimo.

Tabla 30 Plantilla Cadencia

PLANTILLA NOTAS ACORDE TRÍADA		
Atributos	**Valor**	**Descripción**
Tipo	Soprano, contralto, tenor, bajo	Identifica la voz melódica.
Posición	Entero	Indica la posición que ocupa el acorde en la obra.
Grado	I, II, III, IV, V, VI, VII	Indica el grado del acorde.
Fundamental	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Indica el sonido fundamental del acorde
Tercera	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Indica el sonido de tercera del acorde
Quinta	dob, do, do#, reb, re, re#, mib, mi, mi#, fa, fa#, solb, sol, sol#, lab, la, la#, sib, si, si#	Indica el sonido de quinta del acorde

Tabla 31 Plantillas notas acorde tríada

Además han sido necesarias otras estructuras para poder realizar el árbol de búsqueda:

- Acordes candidatos**:** esta plantilla almacena todos los acordes candidatos para una posición concreta, siendo los nodos hoja con un mismo padre común, el acorde elegido en la posición anterior.

PLANTILLA ACORDES CANDIDATOS		
Atributos	**Valor**	**Descripción**
Posición	Entero	Indica la posición a la que pertenece el nodo del árbol

PLANTILLA ACORDES CANDIDATOS		
Atributos	**Valor**	**Descripción**
ID	Entero	Lista de identificadores de los acordes candidatos para esta posición.
Grado	I, II, III, IV, V, VI, VII	Lista de grados de los acordes candidatos para esta posición.
Tipo-posición	Abierto, cerrado	Lista de tipos de posiciones de los acordes candidatos para esta posición.
Estado	Fundamental. Inversión-1, inversión-2	Lista de estados de los acordes candidatos para esta posición.
Fitness	Number	Lista de los fitness de los acordes candidatos para esta posición.
Id-acorde-anterior	Entero	Identifica al acorde actual de la posición anterior. El nodo padre.
Grado-acorde-anterio	I, II, III, IV, V, VI, VI, VII	Indica el grado del acorde actual de la posición anterior. El nodo padre.
Estado-acorde-anterior	Fundamental, inversión-1, inversión-2	Indica el estado del acorde actual de la posición anterior. El nodo padre.

Tabla 32 Plantilla Acordes Candidatos

- Acorde actual: esta plantilla indica el acorde que ha sido elegido como mejor candidato para la posición actual de la obra de acuerdo a una evaluación por fitness en un proceso estocástico.

PLANTILLA ACORDE ACTUAL		
Atributos	**Valor**	**Descripción**
ID	Entero	Lista de identificadores de los acordes candidatos para esta posición.
Posición	Entero	Indica la posición del acorde elegido dentro de la obra así como el nivel de profundidad en el que se encuentra el árbol de búsqueda.
Grado	I, II, III, IV, V, VI, VII	Indica el grado del acorde actual.
Tipo-posición	Abierto, Cerrado	Indica el tipo de posición del acorde actual.
Estado	Fundamental. Inversión-1, inversión-2	Indica el estado del acorde actual.

Tabla 33 Plantilla Acorde Actual

5.3.2 **Base de Reglas**

La base de un sistema experto es la utilización de reglas que modifiquen el contenido de la base de hechos para poder alcanzar una solución.

En este apartado describiremos brevemente las reglas contempladas que agruparemos por bloques en función de su funcionalidad.

5.3.2.1 **Reglas de creación de conocimiento**

Para comenzar a ejecutar nuestro sistema es necesario tener una base de hechos iniciales con el conocimiento musical necesario para poder resolver la armonización. En este punto vamos a comentar las reglas necesarias para la creación de este conocimiento dado que hacían más sencilla la inserción de determinado tipo de conocimiento, al hacerlo de una forma más parecida a como lo haría un músico.

- Crear-intervalos: esta regla se basa en el conocimiento de la distancia tonal entre notas para crear hechos correspondientes a la plantilla de intervalo, para de este modo conocer qué intervalo existe entre cualquier par de notas.
- Calcular-grados-tonales-mayor: esta regla asigna a cada grado de la tonalidad la nota que le corresponde en función de la tonalidad mayor del problema a resolver.
- Calcular-grados-tonales-menor: esta regla asigna a cada grado de la tonalidad la nota que le corresponde en función de la tonalidad menor del problema a resolver.

5.3.2.2 **Reglas de creación de acordes**

En este apartado comentaremos las distintas reglas que se aplican para inferir qué acordes tríada se pueden crear a partir de una nota dada, para a partir de ellos construir la totalidad de acordes posibles para esa nota.

- Calcular-notas-acorde-triada1: esta regla inserta un hecho notas-acorde-triada en el que la fundamental del acorde es la nota de soprano dada. El grado del acorde se corresponde con la fundamental y la posición con la correspondiente a la nota dada.
- Calcular-notas-acorde-triada2: esta regla inserta un hecho notas-acorde-triada en el que la tercera del acorde es la nota de soprano dada. El grado del acorde se corresponde con la fundamental y la posición con la correspondiente a la nota dada.
- Calcular-notas-acorde-triada3: esta regla inserta un hecho notas-acorde-triada en el que la quinta del acorde es la nota de soprano dada. El grado del acorde se corresponde con la fundamental y la posición con la correspondiente a la nota dada.

A partir de este conocimiento base se construyen los acordes en estado fundamental y primera inversión en posición abierta y cerrada.

- Construir-acorde5-cerrada: esta regla se encarga de construir un acorde de quinta en posición cerrada, es decir, con la mínima distancia posible entre notas, a partir de un hecho notas-acorde-triada y la nota del soprano correspondiente. Se construirán dos acordes cuya diferencia radicará en que el bajo se posicionará a distancia de octava entre uno y otro, de este modo queremos ganar variabilidad.
- Construir-acorde5-abierta: esta regla se encarga de construir un acorde de quinta en posición abierta, es decir, con la máxima distancia posible entre notas, a partir de un hecho notas-acorde-triada y la nota del soprano correspondiente. Se construirán dos acordes cuya diferencia radicará en que el bajo se posicionará a distancia de octava entre uno y otro, de este modo queremos ganar variabilidad.
- Construir-acorde6-tercera-tonal-cerrada: esta regla crea un acorde triada en estado de primera inversión en posición cerrada donde la tercera del acorde se corresponde con un grado tonal, es decir, un I, IV o V grado. En este caso duplicaremos la tercera del acorde que se corresponde con el bajo al tratarse de la primera inversión. Se construirán dos acordes cuya diferencia radicará en que el bajo se posicionará a distancia de octava entre uno y otro, de este modo queremos ganar variabilidad.
- Construir-acorde6-tercera-tonal-abierta: esta regla crea un acorde triada en estado de primera inversión en posición abierta donde la tercera del acorde se corresponde con un grado tonal, es decir, un I, IV o V grado. En este caso duplicaremos la tercera del acorde que se corresponde con el bajo al tratarse de la primera inversión. Se construirán dos acordes cuya diferencia radicará en que el bajo se posicionará a distancia de octava entre uno y otro, de este modo queremos ganar variabilidad.
- Construir-acorde6-fundamental-tonal-cerrada: esta regla crea un acorde triada en estado de primera inversión en posición cerrada donde la fundamental del acorde se corresponde con un grado tonal, es decir, un I, IV o V grado. En este caso duplicaremos la fundamental del acorde. Se construirán dos acordes cuya diferencia radicará en que el bajo se posicionará a distancia de octava entre uno y otro, de este modo queremos ganar variabilidad.
- Construir-acorde6-fundamental-tonal-abierta: esta regla crea un acorde triada en estado de primera inversión en posición abierta donde la fundamental del acorde se corresponde con un grado tonal, es decir, un I, IV o V grado. En este caso duplicaremos la fundamental del acorde. Se construirán dos acordes cuya diferencia radicará en que el bajo se posicionará a distancia de octava entre uno y otro, de este modo queremos ganar variabilidad.
- Construir-acorde6-quinta-tonal-cerrada: esta regla crea un acorde triada en estado de primera inversión en posición cerrada donde la quinta del acorde se corresponde con un grado tonal, es decir, un I, IV o V grado. En este caso duplicaremos la quinta del acorde. Se construirán dos acordes cuya diferencia radicará en que el bajo se posicionará a distancia de octava entre uno y otro, de este modo queremos ganar variabilidad.
- Construir-acorde6-quinta-tonal-abierta: esta regla crea un acorde quinta en estado de primera inversión en posición abierta donde la tercera del acorde se corresponde con un grado tonal, es decir, un I, IV o V grado. En este caso

duplicaremos la quinta del acorde. Se construirán dos acordes cuya diferencia radicará en que el bajo se posicionará a distancia de octava entre uno y otro, de este modo queremos ganar variabilidad.

5.3.2.3 **Reglas de verificación de estructura armónica y melódica.**

Para garantizar que se cumplen las reglas de la armonía tradicional, es necesario crear una serie de reglas que verifiquen la correcta estructura melódica y armónica de la solución a generar. A continuación se detallan las reglas encargadas de este propósito.

Melódicas

- Eliminar-acorde-tesitura-voces: esta regla se encarga de eliminar de la base de hechos aquellos acordes que incumplan las restricciones tímbricas de alguna de sus voces. De este modo garantizamos que la solución se va a mover por los registros permitidos.
- Comprobar-movimiento-melodico-contralto: esta regla comprueba que desde la nota del acorde anterior en la posición de contralto al contralto del acorde candidato actual no se incumpla la restricción melódica, es decir, que no haya entre ellas un intervalo superior a la sexta, permitiendo únicamente la octava justa. En caso de que se incumpla esta regla el acorde es borrado de la lista de acordes candidatos para la posición actual del árbol de búsqueda.
- Comprobar-movimiento-melodico-tenor: esta regla comprueba que desde la nota del acorde anterior en la posición de tenor al tenor del acorde candidato actual no se incumpla la restricción melódica, es decir, que no haya entre ellas un intervalo superior a la sexta, permitiendo únicamente la octava justa. En caso de que se incumpla esta regla el acorde es borrado de la lista de acordes candidatos para la posición actual del árbol de búsqueda.
- Comprobar-movimiento-melodico-bajo: esta regla comprueba que desde la nota del acorde anterior en la posición de bajo al bajo del acorde candidato actual no se incumpla la restricción melódica, es decir, que no haya entre ellas un intervalo superior a la sexta, permitiendo únicamente la octava justa. En caso de que se incumpla esta regla el acorde es borrado de la lista de acordes candidatos para la posición actual del árbol de búsqueda.

Armónicas

Las principales restricciones armónicas existentes en la armonía tradicional son las quintas y octavas seguidas y directas que evitaremos con las siguientes reglas.

- Eliminar-5-seguida-soprano-contralto: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde entre cuyas voces de soprano y contralto se produzca un acorde de quinta siempre que en el acorde anterior se produjese este mismo fenómeno entre dichas voces. Sólo se permite si la quinta del acorde actual es disminuida.

- Eliminar-5-seguida-soprano-tenor: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde entre cuyas voces de soprano y tenor se produzca un acorde de quinta siempre que en el acorde anterior se produjese este mismo fenómeno entre dichas voces. Sólo se permite si la quinta del acorde actual es disminuida.
- Eliminar-5-seguida-soprano-bajo: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde entre cuyas voces de soprano y bajo se produzca un acorde de quinta siempre que en el acorde anterior se produjese este mismo fenómeno entre dichas voces. Sólo se permite si la quinta del acorde actual es disminuida.
- Eliminar-5-seguida-contralto-tenor: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde entre cuyas voces de contralto y tenor se produzca un acorde de quinta siempre que en el acorde anterior se produjese este mismo fenómeno entre dichas voces. Sólo se permite si la quinta del acorde actual es disminuida.
- Eliminar-5-seguida-contralto-bajo: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde entre cuyas voces de contralto y bajo se produzca un acorde de quinta siempre que en el acorde anterior se produjese este mismo fenómeno entre dichas voces. Sólo se permite si la quinta del acorde actual es disminuida.
- Eliminar-5-seguida-tenor-bajo: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde entre cuyas voces de tenor y bajo se produzca un acorde de quinta siempre que en el acorde anterior se produjese este mismo fenómeno entre dichas voces. Sólo se permite si la quinta del acorde actual es disminuida.
- Eliminar-5J-directa-soprano-contralto: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de soprano y contralto llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de quinta justa. Se permite si se trata del mismo acorde pero en distinta disposición.
- Eliminar-5J-directa-soprano-tenor: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de soprano y tenor llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de quinta justa. Se permite si se trata del mismo acorde pero en distinta disposición.
- Eliminar-5J-directa-soprano-descendente-bajo: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de soprano y bajo llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de quinta justa. Se permite si se trata del mismo acorde pero en distinta disposición o si el soprano se mueve por grados conjuntos en sentido descendente.
- Eliminar-5J-directa-soprano-ascendente-bajo: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de soprano y bajo llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de quinta justa. Se permite si se trata del mismo acorde pero en distinta disposición o si el soprano se mueve por grados conjuntos en sentido ascendente.

- Eliminar-5J-directa-contralto-descendente-tenor: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de contralto y tenor llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de quinta justa. Se permite si se trata del mismo acorde pero en distinta disposición o si el contralto se mueve por grados conjuntos en sentido descendente mientras el tenor se mueve por salto.
- Eliminar-5J-directa-contralto-ascendente-tenor: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de contralto y tenor llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de quinta justa. Se permite si se trata del mismo acorde pero en distinta disposición o si el contralto se mueve por grados conjuntos en sentido ascendente mientras el tenor se mueve por salto.
- Eliminar-5J-directa-contralto-tenor-descendente: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de contralto y tenor llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de quinta justa. Se permite si se trata del mismo acorde pero en distinta disposición o si el tenor se mueve por grados conjuntos en sentido descendente mientras el contralto se mueve por salto.
- Eliminar-5J-directa-contralto-tenor-ascendente: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de contralto y tenor llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de quinta justa. Se permite si se trata del mismo acorde pero en distinta disposición o si el tenor se mueve por grados conjuntos en sentido ascendente mientras el contralto se mueve por salto.
- Eliminar-5J-directa-contralto-bajo: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de contralto y bajo llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de quinta justa. Se permite si se trata del mismo acorde pero en distinta disposición.
- Eliminar-5J-directa-tenor-bajo: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de tenor y bajo llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de quinta justa. Se permite si se trata del mismo acorde pero en distinta disposición.
- Eliminar-8-seguida-soprano-contralto: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde entre cuyas voces de soprano y contralto se produzca un acorde de octava justa siempre que en el acorde anterior se produjese este mismo fenómeno entre dichas voces.
- Eliminar-8-seguida-soprano-tenor: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde entre cuyas voces de soprano y tenor se produzca un acorde de octava justa siempre que en el acorde anterior se produjese este mismo fenómeno entre dichas voces.

- Eliminar-8-seguida-soprano-bajo: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde entre cuyas voces de soprano y bajo se produzca un acorde de octava justa siempre que en el acorde anterior se produjese este mismo fenómeno entre dichas voces.
- Eliminar-8-seguida-contralto-tenor: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde entre cuyas voces de contralto y tenor se produzca un acorde de octava justa siempre que en el acorde anterior se produjese este mismo fenómeno entre dichas voces.
- Eliminar-8-seguida-contralto-bajo: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde entre cuyas voces de contralto y bajo se produzca un acorde de octava justa siempre que en el acorde anterior se produjese este mismo fenómeno entre dichas voces.
- Eliminar-8-seguida-tenor-bajo: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde entre cuyas voces de tenor y bajo se produzca un acorde de octava justa siempre que en el acorde anterior se produjese este mismo fenómeno entre dichas voces.
- Eliminar-8J-directa-soprano-contralto: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de soprano y contralto llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de octava justa. Se permite si se trata del mismo acorde pero en distinta disposición.
- Eliminar-8J-directa-soprano-tenor: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de soprano y tenor llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de octava justa. Se permite si se trata del mismo acorde pero en distinta disposición.
- Eliminar-8J-directa-soprano-descendente-bajo: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de soprano y bajo llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de octava justa. Se permite si se trata del mismo acorde pero en distinta disposición o si el soprano se mueve por grados conjuntos en sentido descendente.
- Eliminar-8J-directa-soprano-ascendente-bajo: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de soprano y bajo llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de octava justa. Se permite si se trata del mismo acorde pero en distinta disposición o si el soprano se mueve por grados conjuntos en sentido ascendente.
- Eliminar-8J-directa-contralto-descendente-tenor: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de contralto y tenor llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de octava justa. Se permite si se trata del mismo acorde pero en distinta disposición o si el contralto se mueve por grados conjuntos en sentido descendente mientras el tenor se mueve por salto.

- Eliminar-8J-directa-contralto-ascendente-tenor: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de contralto y tenor llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de octava justa. Se permite si se trata del mismo acorde pero en distinta disposición o si el contralto se mueve por grados conjuntos en sentido ascendente mientras el tenor se mueve por salto.
- Eliminar-8J-directa-contralto-tenor-descendente: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de contralto y tenor llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de octava justa. Se permite si se trata del mismo acorde pero en distinta disposición o si el tenor se mueve por grados conjuntos en sentido descendente mientras el contralto se mueve por salto.
- Eliminar-8J-directa-contralto-tenor-ascendente: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de contralto y tenor llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de octava justa. Se permite si se trata del mismo acorde pero en distinta disposición o si el tenor se mueve por grados conjuntos en sentido ascendente mientras el contralto se mueve por salto.
- Eliminar-8J-directa-contralto-bajo: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de contralto y bajo llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de octava justa. Se permite si se trata del mismo acorde pero en distinta disposición.
- Eliminar-8J-directa-tenor-bajo: esta regla se encarga de eliminar de la lista de acordes candidatos aquel acorde cuyas voces de tenor y bajo llegan a sus correspondientes voces del acorde actual por movimiento directo formando entre ellas un intervalo de octava justa. Se permite si se trata del mismo acorde pero en distinta disposición.

Reglas de creación y manipulación del árbol de búsqueda

Para poder alcanzar una solución se utilizará un árbol de búsqueda avaricioso que elegirá el mejor acorde para cada posición de forma estocástica en función de su fitness teniendo más probabilidades de ser elegido si es un mejor candidato.

Permitimos que la elección sea aleatoria para dar margen a la variabilidad de las soluciones, ya que soluciones a priori peores pueden ser globalmente mejores o más interesantes para el usuario.

- Inicialización-acordes-candidatos-pos-1: crea la estructura básica del primer nodo del árbol.
- Inicialización-acordes-candidatos-pos-n: crea la estructura básica de un nodo en la posición n del árbol, siendo n la profundidad del mismo. Referencia como padre el acorde elegido para la posición anterior.
- Crear-acordes-candidatos: esta regla se encarga de insertar en la estructura del árbol los acordes candidatos para la posición actual, es decir todos aquellos cuya

posición se corresponda con la profundidad actual del árbol. Inicialmente se insertan con un fitness de-1.

- Están-creados-acordes-candidatos: regla que verifica que todos los acordes candidatos para la posición actual han sido insertados en la estructura del árbol, insertando el hecho acordes-candidatos-creados que hace de barrera para sincronizar el proceso de búsqueda permitiendo eliminar candidatos no válidos.
- Volver-atrás: en el caso de que no haya ningún candidato válido para la posición actual se vuelve a la posición anterior, se elimina el candidato anteriormente elegido de la lista para la posición anterior y se vuelve a elegir un nuevo candidato de forma estocástica.
- Calcular-fitness-acorde-pos-1: esta regla es la encargada de calcular el fitness de un acorde candidato para la posición 1. Se realiza a parte ya que no proviene de ningún acorde anterior y por tanto no hay una progresión armónica, por tanto su fitness se calcula en base a otros criterios, más concretamente por un análisis de acordes frecuentes en la posición inicial.
- Calcular-fitness-acorde-pos-n: esta regla calcula el fitness de un acorde candidato en la posición n de la obra. Para ello tiene en cuenta la progresión armónica que genera, su fitness por cadencia, por el tipo de grado que es y por los gustos del usuario.
- Elegir-candidato: mediante esta regla se elige el acorde candidato de manera estocástica. Para ello se realiza el algoritmo de la ruleta sobre los acordes candidatos en función de su fitness, otorgando más probabilidad de ser elegido a aquel que posee un fitness mayor.

Reglas de escritura de fichero µousiké

Por último se comentan las reglas implementadas para realizar la escritura de la solución obtenida en el formato del compilador µousiké.

- Armonización-terminada: esta regla se encarga de comprobar que se ha alcanzado una solución al haber sido elegido un acorde de la última posición de la obra.
- Generar-cabecera: esta regla crea la cabecera del fichero de salida en función de los hechos a cerca de su tonalidad, compás, tempo y título.
- Completar-con-silencios: mediante esta regla completamos la obra con silencios al final de tal modo que quede correctamente escrita de acuerdo a las normas musicales que también exige el compilador µousiké.
- Fin-compas: comprueba si es fin de compás para insertar el símbolo de separación de compases.
- Generar-acorde: traduce el acorde elegido en la solución en una posición dada al lenguaje µousiké para cada una de las partituras que representan cada una de las cuatro voces.
- Generar-acorde-ligadura: traduce el acorde elegido en la solución en una posición dada al lenguaje µousiké para cada una de las partituras que representan cada una de las cuatro voces en caso de que no coja en el compás actual y sea necesario realizar una ligadura entre compases.

- Escribir-fichero: regla que se encarga de la escritura del fichero de salida en formato μousiké, realizando la escritura de una obra con cuatro partituras, una para cada voz. Además escribe temporal "estructura-obra" que contiene la información sobre la estructura de la obra elegida como solución y que será empleado en caso de que el usuario realice una valoración de la obra para la actualización de sus gustos.

5.3.3 **Motor de Inferencia**

El motor de Inferencia es el encargado de decidir qué reglas se ejecutan en cada momento dependiendo de las precondiciones que se cumplan. La estrategia de resolución de conflictos utilizada ha sido por profundidad eligiéndose la primera regla insertada en el sistema que cumple las precondiciones.

5.4 **Módulo de Aprendizaje Automático**

5.4.1 **Análisis armónico**

Este módulo es el encargado de realizar un análisis armónico de una partitura con el fin de poder compararla con otras para extraer así el grado de similitud existente entre ambas.

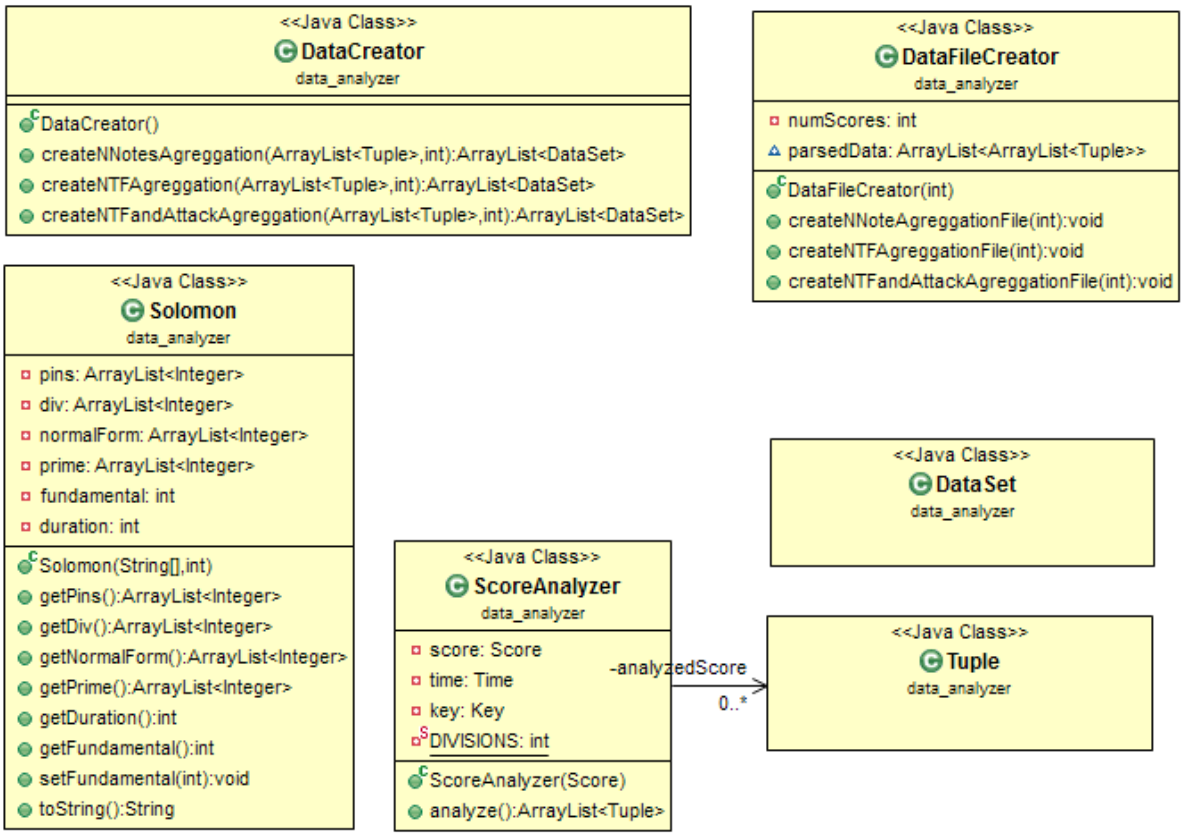


Figura 42: Diagrama de clases del analizador armónico

El sistema es capaz de generar datos en bruto y datos procesados, definidos por las clases *Tuple* y *DataSet* respectivamente. Tal y como se aprecia en el diagrama el analizador armónico implementado en la clase *ScoreAnalyzer* genera mediante el método *analyze()* los datos en bruto del análisis de la partitura que incluyen la información del Set Theory Prime, especificado en la clase *Solomon* que implementa el algoritmo especificado en el capítulo de diseño 4.5.2.1 Análisis armónico.

Los datos procesados son generados por la clase *DataCreator* la cual es capaz de generar *DataSet* de tres tipos diferentes por medio de tres métodos:

- *CreateNotesAgreggation*: genera un DataSet formado por un grupo de tuplas especificado por parámetro.
- *CreateTFAgreggation*: crea un DataSet formado por un grupo de tuplas que abarcan un número, especificado por parámetro, de tiempos de pulso en la línea melódica del soprano.
- *CreateTFandAttackAgreggation*: similar al método anterior, genera un DataSet formado por un grupo de tuplas que abarcan un número específico de tiempos de pulso donde todas las notas son atacadas a la vez.

El listado de *DataSet* es escrito en un fichero de datos por medio de la clase *DataFileCreator* para permitir ser utilizado en un futuro sin necesidad de repetir el análisis. Esta operación únicamente se realiza en el conjunto de partituras de ejemplo.

Para un mayor entendimiento de la información extraída en el proceso de análisis de detalla a continuación la información contenida tanto en *Tuple* como en un *DataSet*.

Tuple	
Atributo	Significado
ID	Identificador de la partitura.
Beats	Número de pulsos por compás.
Beats Type	Figura que especifica el pulso del compás.
Fifths	Alteraciones de la tonalidad.
mode	Modo de la tonalidad.
Divisions	Número de divisiones que especifican la unidad de pulso.
Measure	Número de compás donde aparece la tupla en la partitura.
Measure beat	En qué pulso del compás aparece.
Voice 1	Nota de la primera voz.
Voice 2	Nota de la segunda voz.
Voice 3	Nota de la tercera voz.
Voice 4	Nota de la cuarta voz.
Octave 1	Octava de la primera voz.
Octave 2	Octava de la segunda voz.
Octave 3	Octava de la tercera voz.
Octave 4	Octava de la cuarta voz.
Duration 1	Duración total de la primera nota.
Duration 2	Duración total de la segunda nota.
Duration 3	Duración total de la tercera nota.
Duration 4	Duración total de la cuarta nota.
Remaining Duration 1	Duración que queda por reproducir de la nota 1.
Remaining Duration 2	Duración que queda por reproducir de la nota 2.
Remaining Duration 3	Duración que queda por reproducir de la nota 3.
Remaining Duration 4	Duración que queda por reproducir de la nota 4.
H1-H2	Distancia existente entre la primera y la segunda voz.
H2-H3	Distancia existente entre la segunda y tercera voz
H3-H4	Distancia existente entre la tercera y la cuarta voz.
H1-H4	Distancia existente entre la primera y la cuarta voz.
In tone 1	Indica si la nota de la primera voz pertenece a la tonalidad.
In tone 2	Indica si la nota de la segunda voz pertenece a la tonalidad.
In Tone 3	Indica si la nota de la tercera voz pertenece a la tonalidad.

Tuple	
Atributo	Significado
In tone 4	Indica si la nota de la cuarta voz pertenece a la tonalidad.
Crossed 1	Si la primera voz se cruza con alguna otra.
Crossed 2	Si la segunda voz se cruza con alguna otra.
Crossed 3	Si la tercera voz se cruza con alguna otra.
Crossed 4	Si la cuarta voz se cruza con alguna otra.
Attacked 1	Si la primera nota está siendo atacada en ese instante.
Attacked 2	Si la segunda nota está siendo atacada en ese instante.
Attacked 3	Si la tercera nota está siendo atacada en ese instante.
Attacked 4	Si la cuarta nota está siendo atacada en ese instante.
Duplicate 1	Si se encuentra duplicada la primera nota.
Duplicate 2	Si se encuentra duplicada la segunda nota.
Duplicate 3	Si se encuentra duplicada la tercera nota.
Duplicate 4	Si se encuentra duplicada la cuarta nota.
Duplicate base	Si se encuentra duplicada la fundamental del acorde.
Tempo forte	Si es interpretado en un golpe de pulso o no.
Fundamental	Fundamental del acorde.
Grade	Grado del acorde.
Normal form	Normal form del acorde, es decir, su disposición en la que la distancia entre notas es la mínima posible.
Prime form	Forma Prime del acorde.
Chord Duration	Duración total del acorde.

Tabla 34: Especificación de los atributos que forman una tupla.

DataSet	
Atributos	Significado
ID	Identificador de la partitura.
Beats	Número de pulsos por compás.
Beats Type	Figura que especifica el pulso del compás.
mode	Modo de la tonalidad.
Divisions	Número de divisiones que especifican la unidad de pulso.
Duration	Duración total del DataSet.
Measure	Número de compás.
Measure beat	Pulso de compás.

DataSet	
Atributos	Significado
TF	Indica si comienza en un comienzo de pulso.
Chords Durations	Lista con la duración de los acordes que forman el DataSet.
Interval to Tonic 1	Distancia desde la nota de la primera voz a la tónica[4].
Interval to Tonic 2	Distancia desde la nota de la segunda voz a la tónica.
Interval to Tonic 3	Distancia desde la nota de la tercera voz a la tónica.
Interval to Tonic 4	Distancia desde la nota de la cuarta voz a la tónica.
In Tone 1	Indica si la primera nota del patrón pertenece a la tonalidad.
In Tone 2	Indica si la segunda nota del patrón pertenece a la tonalidad.
In Tone 3	Indica si la tercera nota del patrón pertenece a la tonalidad.
In Tone 4	Indica si la cuarta nota del patrón pertenece a la tonalidad.
Octave 1	Octava de la primera nota del patrón.
Octave 2	Octava de la segunda nota del patrón.
Octave 3	Octava de la tercera nota del patrón.
Octave 4	Octava de la cuarta nota del patrón.
Moves 1	Patrón de movimientos armónicos de la primera voz con la segunda.
Moves 2	Patrón de movimientos armónicos de la segunda voz con la tercera.
Moves 3	Patrón de movimientos armónicos de la tercera voz con la cuarta.
Moves 4	Patrón de movimientos armónicos de la primera voz con la cuarta.
Diffs 1-2	Patrón de diferencias interválicas entre las voces 1 y 2.
Diffs 2-3	Patrón de diferencias interválicas entre las voces 2 y 3.
Diffs 3-4	Patrón de diferencias interválicas entre las voces 3 y 4.
Diffs 1-4	Patrón de diferencias interválicas entre las voces 1 y 4.
Interval Pattern 1	Patrón de intervalos producido por la primera voz.
Interval Pattern 2	Patrón de intervalos producido por la segunda voz.

[4] Se indica por intervalos para hacer el DataSet independiente de la tonalidad y poder ser extrapolable a cualquier caso.

DataSet	
Atributos	Significado
Interval Pattern 3	Patrón de intervalos producido por la tercera voz.
Interval Pattern 4	Patrón de intervalos producido por la cuarta voz.
Duration Pattern 1	Patrón de duraciones de la primera voz.
Duration Pattern 2	Patrón de duraciones de la segunda voz.
Duration Pattern 3	Patrón de duraciones de la tercera voz.
Duration Pattern 4	Patrón de duraciones de la cuarta voz.
Remaining Durations 1	Patrón de duraciones restantes de la primera voz.
Remaining Durations 2	Patrón de duraciones restantes de la segunda voz.
Remaining Durations 3	Patrón de duraciones restantes de la tercera voz.
Remaining Durations 4	Patrón de duraciones restantes de la cuarta voz.
Grade Pattern	Patrón de los grados de cada conjunto de notas.
Attack Pattern 1	Patrón de ataques de la primera voz.
Attack Pattern 2	Patrón de ataques de la segunda voz.
Attack Pattern 3	Patrón de ataques de la tercera voz.
Attack Pattern 4	Patrón de ataques de la cuarta voz.
Crossed Pattern 1	Patrón de cruces de la primera voz con cualquier otra.
Crossed Pattern 2	Patrón de cruces de la segunda voz con cualquier otra.
Crossed Pattern 3	Patrón de cruces de la tercera voz con cualquier otra.
Crossed Pattern 4	Patrón de cruces de la cuarta voz con cualquier otra.
Prime Pattern	Patrón de las formas Prime de cada conjunto de notas que forma el DataSet.
previous Duration	Duración total del DataSet anterior
previous Measure	Número de compás total del DataSet anterior.
previous Measure beat	Pulso de compás total del DataSet anterior.
previous TF	Indica si el DataSet anterior comienza en un comienzo de pulso.
previous Chords Durations	Lista con la duración de los acordes que forman el DataSet anterior.

DataSet	
Atributos	Significado
previous Interval to Tonic 1	Distancia desde la nota de la primera voz a la tónica[5] del DataSet anterior.
previous Interval to Tonic 2	Distancia desde la nota de la segunda voz a la tónica total del DataSet anterior.
previous Interval to Tonic 3	Distancia desde la nota de la tercera voz a la tónica del DataSet anterior.
previous Interval to Tonic 4	Distancia desde la nota de la cuarta del DataSet anterior. voz a la tónica
previous In Tone 1	Indica si la primera nota del patrón del DataSet anterior pertenece a la tonalidad.
previous In Tone 2	Indica si la segunda nota del patrón del DataSet anterior pertenece a la tonalidad.
previous In Tone 3	Indica si la tercera nota del patrón del DataSet anterior pertenece a la tonalidad.
previous In Tone 4	Indica si la cuarta nota del patrón del DataSet anterior pertenece a la tonalidad.
previous Octave 1	Octava de la primera nota del patrón del DataSet anterior.
previous Octave 2	Octava de la segunda nota del patrón del DataSet anterior.
previous Octave 3	Octava de la tercera nota del patrón del DataSet anterior.
previous Octave 4	Octava de la cuarta nota del patrón del DataSet anterior.
previous Moves 1	Patrón de movimientos armónicos de la primera voz con la segunda del DataSet anterior.
previous Moves 2	Patrón de movimientos armónicos de la segunda voz con la tercera del DataSet anterior.
previous Moves 3	Patrón de movimientos armónicos de la tercera voz con la cuarta del DataSet anterior.
previous Moves 4	Patrón de movimientos armónicos de la primera voz con la cuarta.
previous Diffs 1-2	Patrón de diferencias interválicas entre las voces 1 y 2 del DataSet anterior.
previous Diffs 2-3	Patrón de diferencias interválicas entre las voces 2 y 3 del DataSet anterior.

[5] Se indica por intervalos para hacer el DataSet independiente de la tonalidad y poder ser extrapolable a cualquier caso.

DataSet	
Atributos	Significado
previous Diffs 3-4	Patrón de diferencias interválicas entre las voces 3 y 4 del DataSet anterior.
previous Diffs 1-4	Patrón de diferencias interválicas entre las voces 1 y 4 del DataSet anterior.
previous Interval Pattern 1	Patrón de intervalos producido por la primera voz del DataSet anterior.
previous Interval Pattern 2	Patrón de intervalos producido por la segunda voz del DataSet anterior.
previous Interval Pattern 3	Patrón de intervalos producido por la tercera voz del DataSet anterior.
previous Interval Pattern 4	Patrón de intervalos producido por la cuarta voz del DataSet anterior.
previous Duration Pattern 1	Patrón de duraciones de la primera voz del DataSet anterior.
previous Duration Pattern 2	Patrón de duraciones de la segunda voz del DataSet anterior.
previous Duration Pattern 3	Patrón de duraciones de la tercera voz del DataSet anterior.
previous Duration Pattern 4	Patrón de duraciones de la cuarta voz del DataSet anterior.
previous Remaining Durations 1	Patrón de duraciones restantes de la primera voz del DataSet anterior.
previous Remaining Durations 2	Patrón de duraciones restantes de la segunda voz del DataSet anterior.
previous Remaining Durations 3	Patrón de duraciones restantes de la tercera voz del DataSet anterior.
previous Remaining Durations 4	Patrón de duraciones restantes de la cuarta voz del DataSet anterior.
previous Grade Pattern	Patrón de los grados de cada conjunto de notas del DataSet anterior.
previous Attack Pattern 1	Patrón de ataques de la primera voz del DataSet anterior.
previous Attack Pattern 2	Patrón de ataques de la segunda voz del DataSet anterior.
previous Attack Pattern 3	Patrón de ataques de la tercera voz del DataSet anterior.
previous Attack Pattern 4	Patrón de ataques de la cuarta voz del DataSet anterior.
previous Crossed Pattern 1	Patrón de cruces de la primera voz con cualquier otra del DataSet anterior.
previous Crossed Pattern 2	Patrón de cruces de la segunda voz con cualquier otra del DataSet anterior.
previous Crossed Pattern 3	Patrón de cruces de la tercera voz con cualquier otra del DataSet anterior.
previous Crossed Pattern 4	Patrón de cruces de la cuarta voz con cualquier otra del DataSet anterior.

DataSet	
Atributos	Significado
previous Prime Pattern	Patrón de las formas Prime de cada conjunto de notas que forma el DataSet anterior.

Tabla 35: Especificación de los atributos que forman un DataSet

5.4.2 **Módulo de Aprendizaje Automático basado en Instancias**

El módulo de Aprendizaje Automático basado en Instancias se encarga de comparar el análisis de la partitura a armonizar con el conjunto de datos analizados de partituras de ejemplo. Esas partituras representan la experiencia del compositor siendo usadas para modificar el esqueleto armónico generado por el Sistema Experto armonizador.

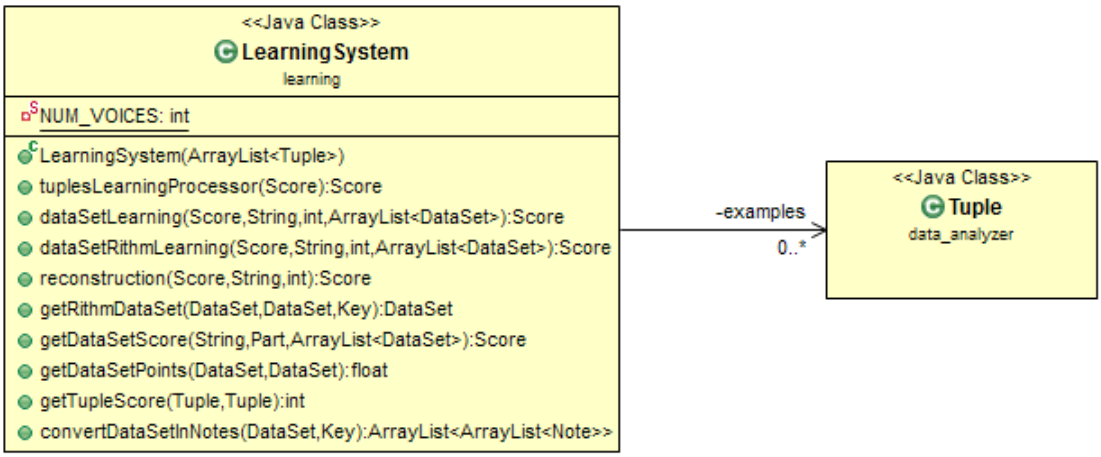


Figura 43: Diagrama de clases del Sistema de Aprendizaje Automático basado en Instancias

El proceso de aprendizaje se encuentra implementado en la clase *LearningSystem*. Esta clase implementa tres métodos de aprendizaje:

- tuplesLearningProcessor: este método se encarga de realizar un proceso de Aprendizaje Automático basado en Instancias sobre datos en bruto sin procesar.
- dataSetRithmLearning: realiza un proceso de Aprendizaje Automático basado en Instancias sobre datos previamente procesados, realizando únicamente una modificación de los patrones rítmicos de la obra.
- dataSetLearning: al igual que el método anterior realiza un proceso de Aprendizaje Automático basado en Instancias sobre datos previamente procesados, con la diferencia que el proceso de sustitución modifica tanto el ritmo como la melodía de la obra.

Capítulo 6

Experimentación

Esta sección describe el conjunto de experimentos realizados para evaluar la capacidad de armonización del software desarrollado en el proyecto. El conjunto de experimentos seleccionado consiste en la armonización de una melodía con las siguientes configuraciones del software:

- Sin sustituciones. La armonización resultante es directamente la salida del sistema experto sin ninguna intervención del módulo de aprendizaje automático.
- Con sustituciones rítmicas (tamaño de grupo 2).
- Con sustituciones rítmicas (tamaño de grupo 1).
- Con sustituciones rítmico-melódicas (tamaño de grupo 2) con partituras de estilos diferentes.

Una vez evaluado el comportamiento de distintas configuraciones del sistema mostraremos los resultados obtenidos al experimentar con la mejor de las configuraciones encontradas.

Para poder apreciar en mayor medida las diferencias se ha decidido realizar los experimentos iniciales sobre la misma línea melódica. Se ha optado por una melodía escrita en Re menor, compuesta por nueve compases entre los que aparecen agrupaciones de notas. Se quiere comprobar el comportamiento del sistema ante este tipo de agrupaciones.

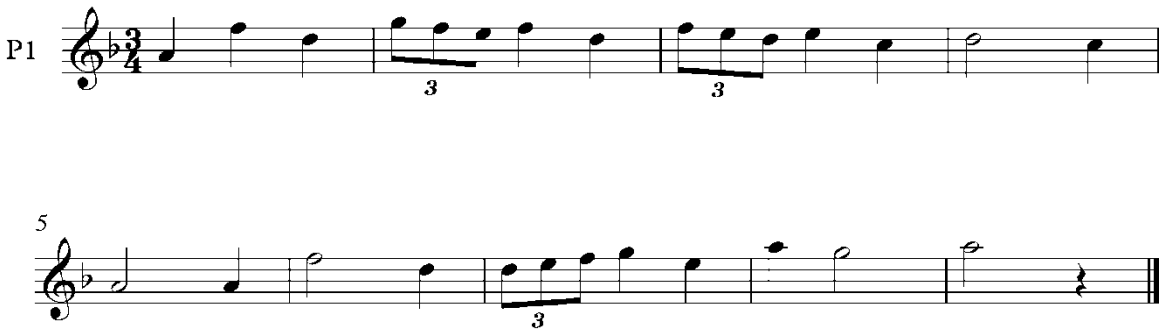


Figura 44: Experimento 1 - partitura de entrada

6.1 Experimento 0

6.1.1 Objetivo

El objetivo de este experimento es realizar una armonización sin sustituciones, plasmando como salida únicamente la armonización realizada por el Sistema Experto, para de este modo apreciar más adelante las diferencias existentes tras aplicar el módulo de Aprendizaje Automático.

6.1.2 Descripción

Se pretende obtener una composición armónica completa a cuatro voces utilizando únicamente el Sistema Experto armonizador que generará el esqueleto armónico de la obra.

6.1.3 **Resultados**

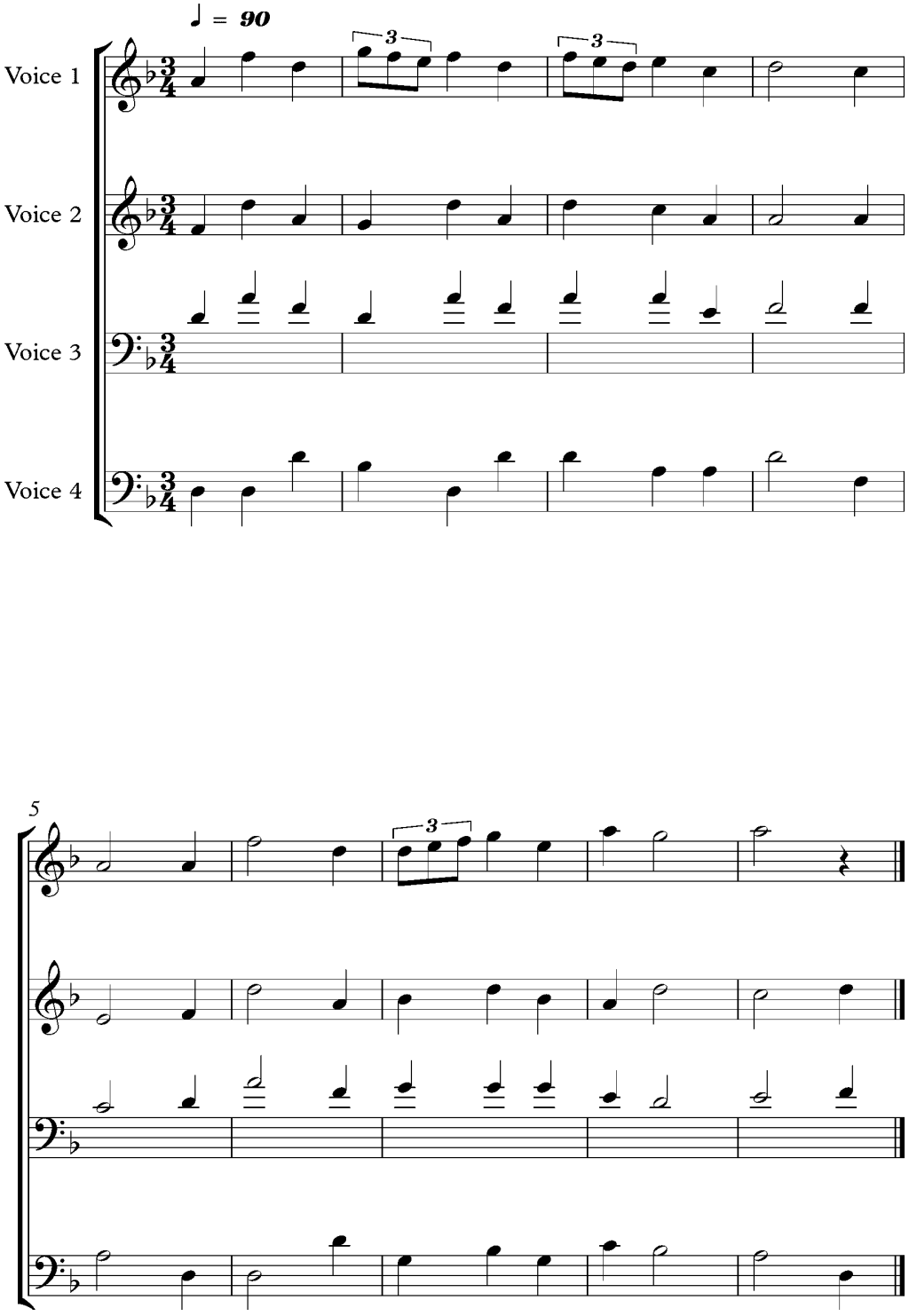


Figura 45: Experimento 0 – salida generada

6.1.4 **Análisis de los resultados**

La salida generada se corresponde con una armonía bien formada y de buena sonoridad. Al ser el acompañamiento únicamente el esqueleto armónico, se aprecia como las melodías solo funcionan en su conjunto, no comportándose como melodías independientes en ningún momento resultando una composición poco expresiva, con patrones rítmicos idénticos en todo el acompañamiento.

6.2 **Experimento 1**

6.2.1 **Objetivo**

El objetivo de este primer experimento es la armonización de una obra musical utilizando una sustitución de los patrones rítmicos de las voces generadas automáticamente.

6.2.2 **Descripción**

Se pretende obtener una composición armónica completa a cuatro voces que previamente será modificada utilizando para ello un banco de ejemplos de más de trescientas corales previamente analizadas.

Para el análisis armónico se utilizará una agrupación tipo *TFandAggregate*, donde los conjuntos están formados por las notas contenidas entre dos tiempos de ataque de pulso donde todas las voces sean atacadas en ese instante. El tamaño de grupo utilizado será de $n = 2$.

6.2.3 **Resultados**

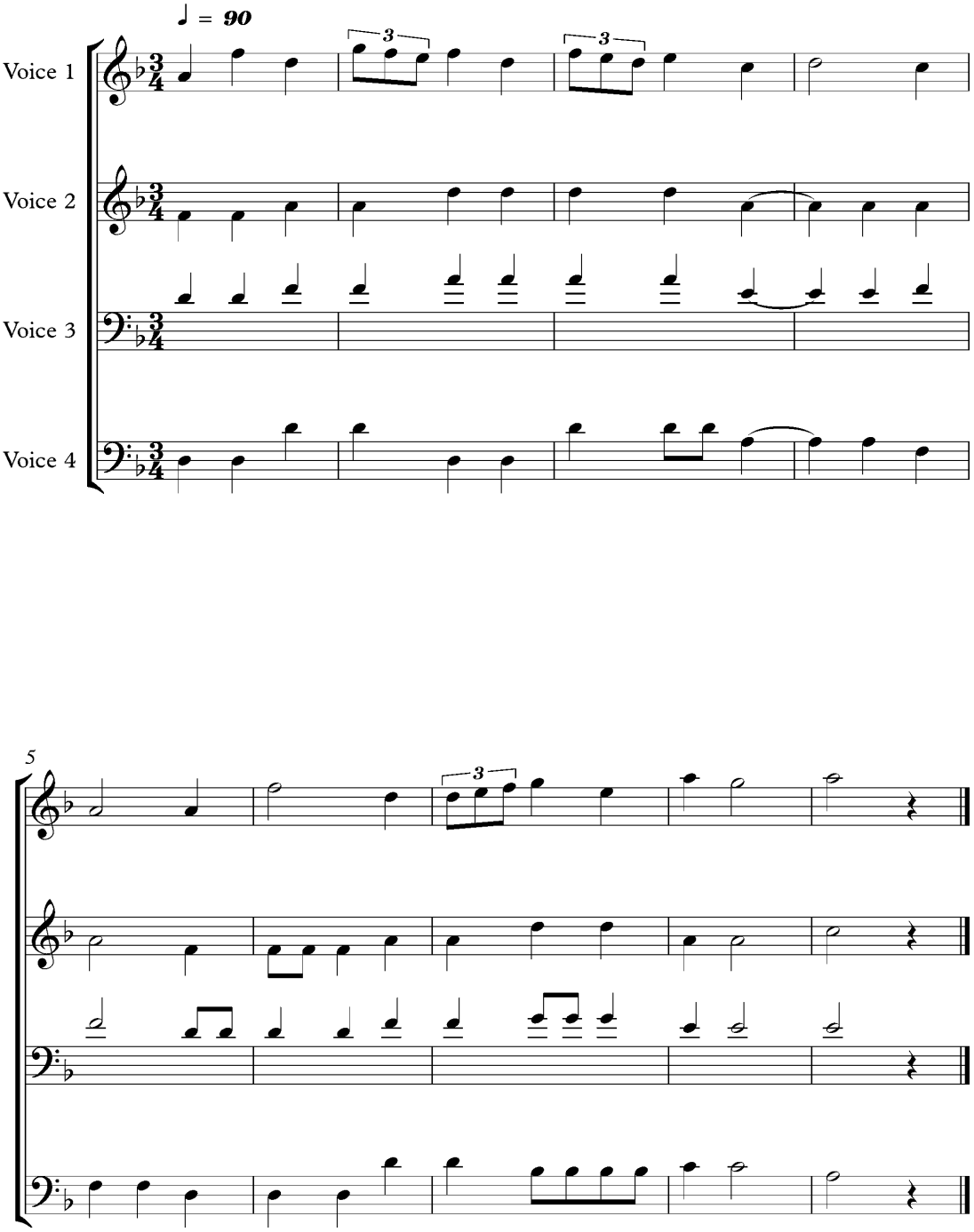


Figura 46: Experimento 1 – salida generada

6.2.4 **Análisis de los resultados**

El resultado a primera vista parece bastante satisfactorio, pero se puede observar como agrupaciones grandes en el análisis pueden provocar una pérdida del esqueleto armónico de la obra. Esto se debe a que el proceso de sustitución de patrones rítmicos puede encontrarse con situaciones en las que para varias notas del esqueleto armónico actual deban ser sustituidas por una única figura, lo que provoca pérdida de información armónica.

Por otro lado el uso de agrupaciones de un tamaño n superior a la unidad no proporciona ninguna ventaja clara en la sustitución de patrones rítmicos al no proporcionar mayor variabilidad rítmica en las melodías.

6.3 **Experimento 2**

6.3.1 **Objetivo**

El objetivo de este experimento será tratar de mejorar el experimento anterior haciendo uso de un tamaño de agrupación menor en el análisis armónico.

6.3.2 **Descripción**

Se pretende obtener una composición armónica completa a cuatro voces que mejore el resultado obtenido en el experimento anterior.

Para el análisis armónico se utilizará una agrupación tipo *TFandAggregate*, donde los conjuntos están formados por las notas contenidas entre dos tiempos de ataque de pulso donde todas las voces sean atacadas en ese instante. El tamaño de grupo utilizado será de $n = 1$.

6.3.3 **Resultados**

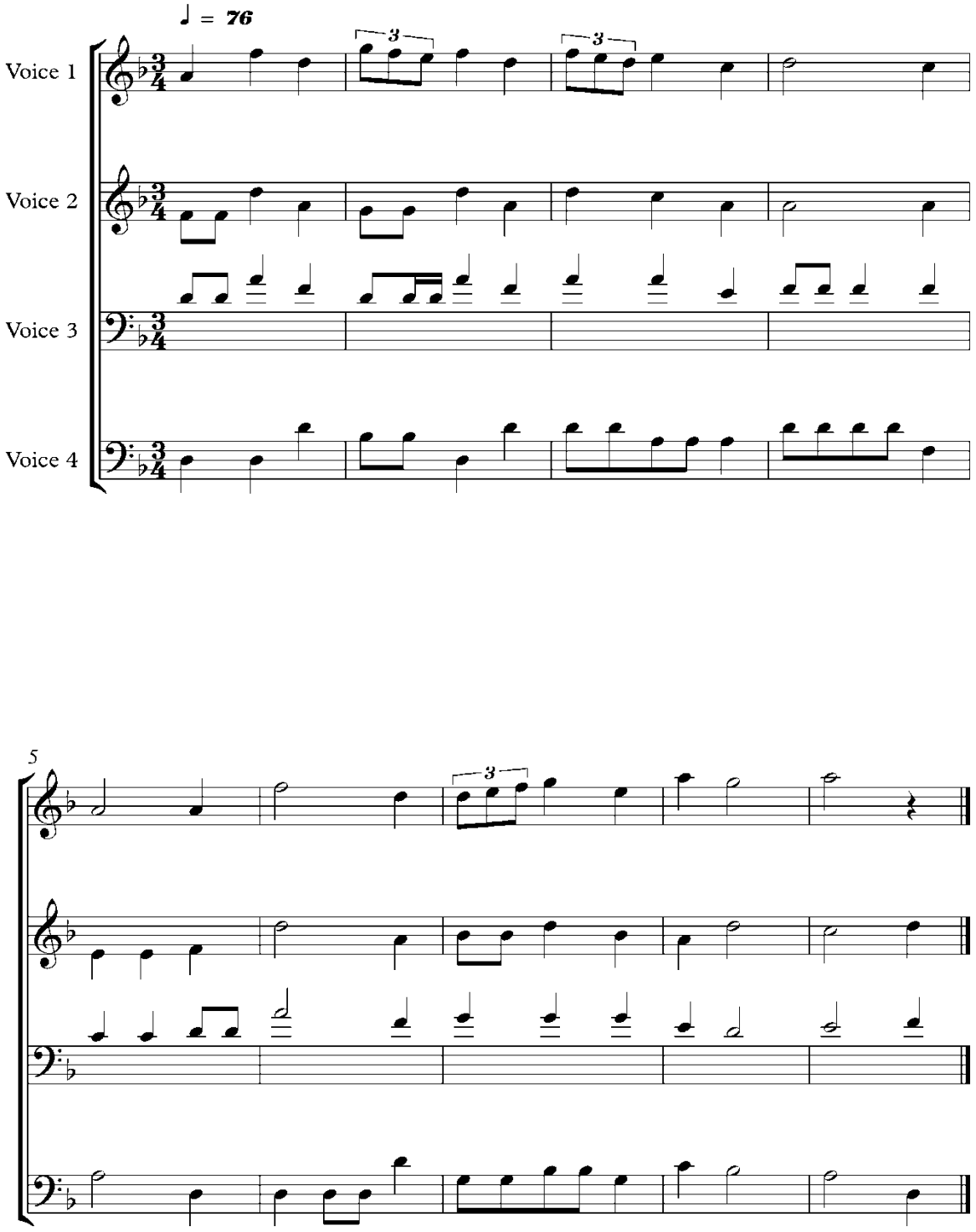


Figura 47: Experimento 2 – salida generada

6.3.4 **Análisis de los resultados**

El resultado obtenido en esta ocasión es significativamente mejor que el obtenido en el experimento anterior al respetar la estructura armónica de la obra. La disminución del tamaño de agrupación permite que no se pierda información del esqueleto armónico a la vez que proporciona mayor variabilidad rítmica a la obra.

Entre otras ventajas hay que destacar que dada la complejidad del proceso de ponderación de similitud, para agrupaciones pequeñas el sistema encuentra instancias más parecidas ya que al ser de menor duración se hace más fácil evitar los problemas asociados a las disonancias difíciles de tratar y evitar.

6.4 **Experimento 3**

6.4.1 **Objetivo**

El objetivo de este experimento será probar una sustitución rítmico-melódica en la misma melodía utilizada en los experimentos anteriores.

6.4.2 **Descripción**

En esta ocasión se va a realizar una composición armónica automática realizando una sustitución rítmico-melódica del esqueleto armónico generado por el Sistema Experto armonizador. La melodía utilizada nuevamente es la empleada en los experimentos anteriores para cuantificar las diferencias obtenidas.

Para el análisis armónico se utilizará una agrupación tipo *TFandAggregate* siendo el tamaño de grupo utilizado de $n = 2$.

6.4.3 **Resultado**

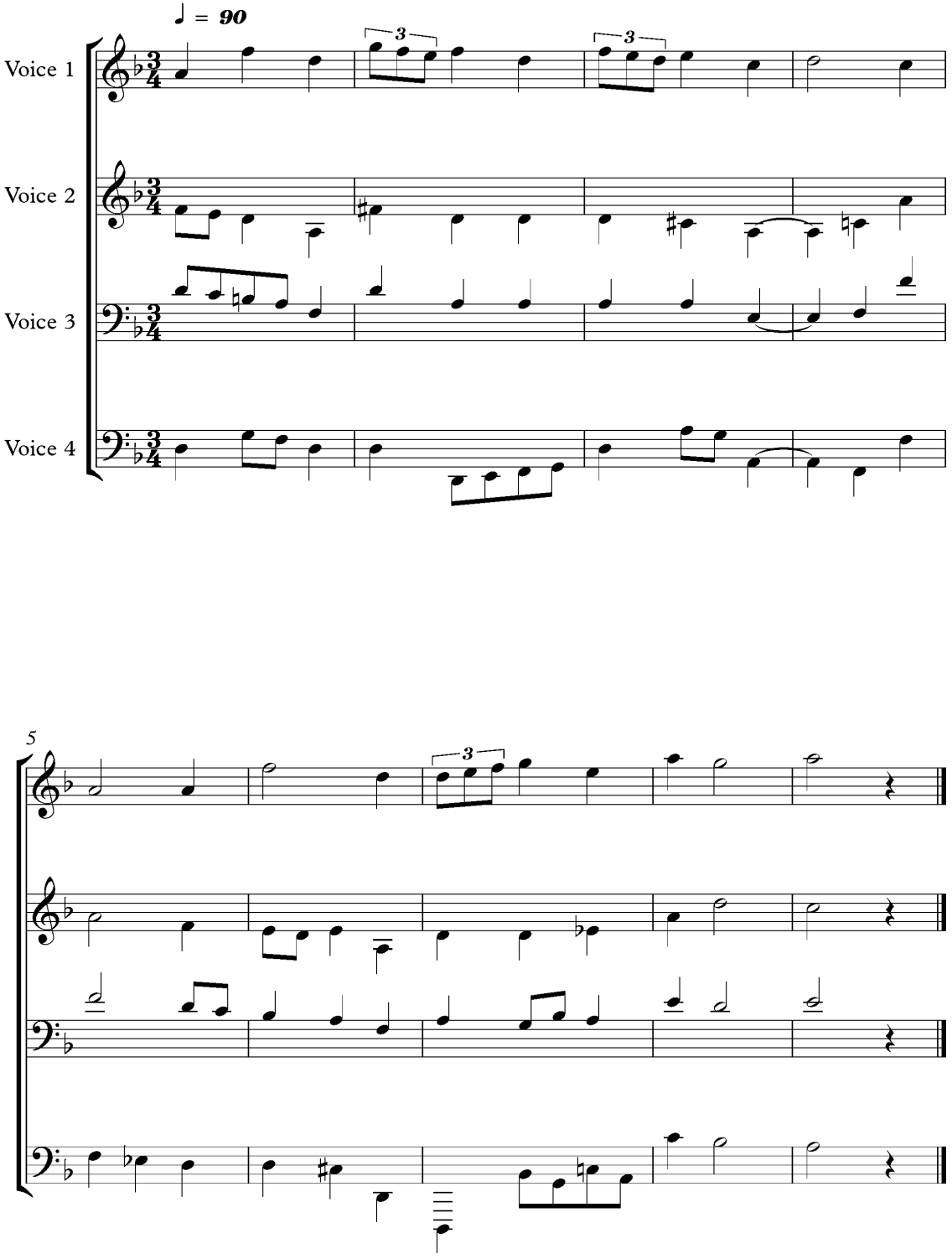


Figura 48: Experimento 3 – salida generada

6.4.4 **Análisis de los resultados**

En esta ocasión el resultado obtenido es musicalmente malo. Puede apreciarse por la aparición de disonancias producida por notas extrañas a la tonalidad, un mal encadenamiento melódico con numerosos saltos de gran tamaño interválico, y una pérdida grande del sentido armónico original creado por el Sistema Experto.

El motivo de este resultado no es el uso de una agrupación de tamaño 2 para el análisis armónico, ya que en la sustitución rítmico-melódica se pierde por completo el esqueleto armónico original que únicamente es utilizado para encontrar la instancia más similar.

El problema viene dado por varios factores, siendo el más importante el estilo musical de la obra y el estilo de los ejemplos utilizados. Pese a que con la sustitución de patrones rítmicos no tiene demasiada importancia el estilo de los ejemplos, cuando se produce una sustitución melódica sí que es un factor a considerar. Si se reproduce la obra se apreciará un fuerte carácter barroco en el acompañamiento debido a que es el estilo musical del conjunto de ejemplos. Esto provoca fuertes disonancias fruto de la unión de estilos totalmente opuestos.

Otro factor que influye en la calidad del resultado, es tal y como se ha comentado en experimentos anteriores, la dificultad de ponderar la similitud entre dos instancias donde pueden parecer más similares dos melodías que en realidad son disonantes entre sí. Este proceso que no afectaba a la sustitución rítmica, se hace muy notable en la sustitución melódica.

Se puede concluir que la sustitución rítmico-melódica con el conjunto de ejemplos actual está totalmente contraindicada.

6.5 **Experimento 4**

6.5.1 **Objetivo**

Se va a realizar una sustitución rítmico-melódica de estilo barroco para comprobar el funcionamiento del proceso de armonización y Aprendizaje Automático con una entrada acorde al banco de ejemplos.

6.5.2 **Descripción**

La composición armónica se realizará utilizando una sustitución rítmico-melódica con una agrupación para el análisis armónico *TFandAggregate* siendo el tamaño de grupo utilizado de $n = 2$.

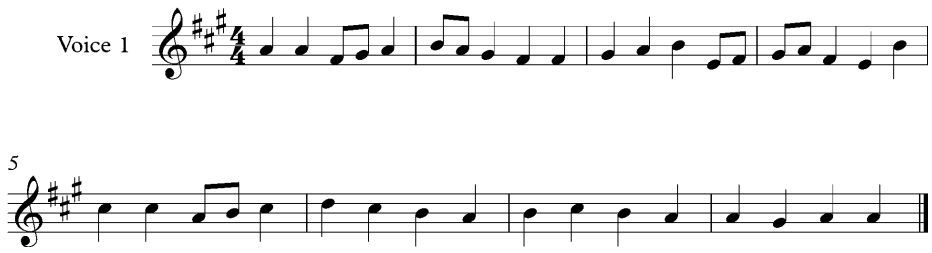


Figura 49: Experimento 4 – partitura de entrada

En esta ocasión la melodía utilizada tiene un estilo barroco al igual que el conjunto de partituras de ejemplo. Se quiere comprobar el comportamiento del sistema sin que interfiera una mala elección de los datos de ejemplo.

6.5.3 **Resultados**

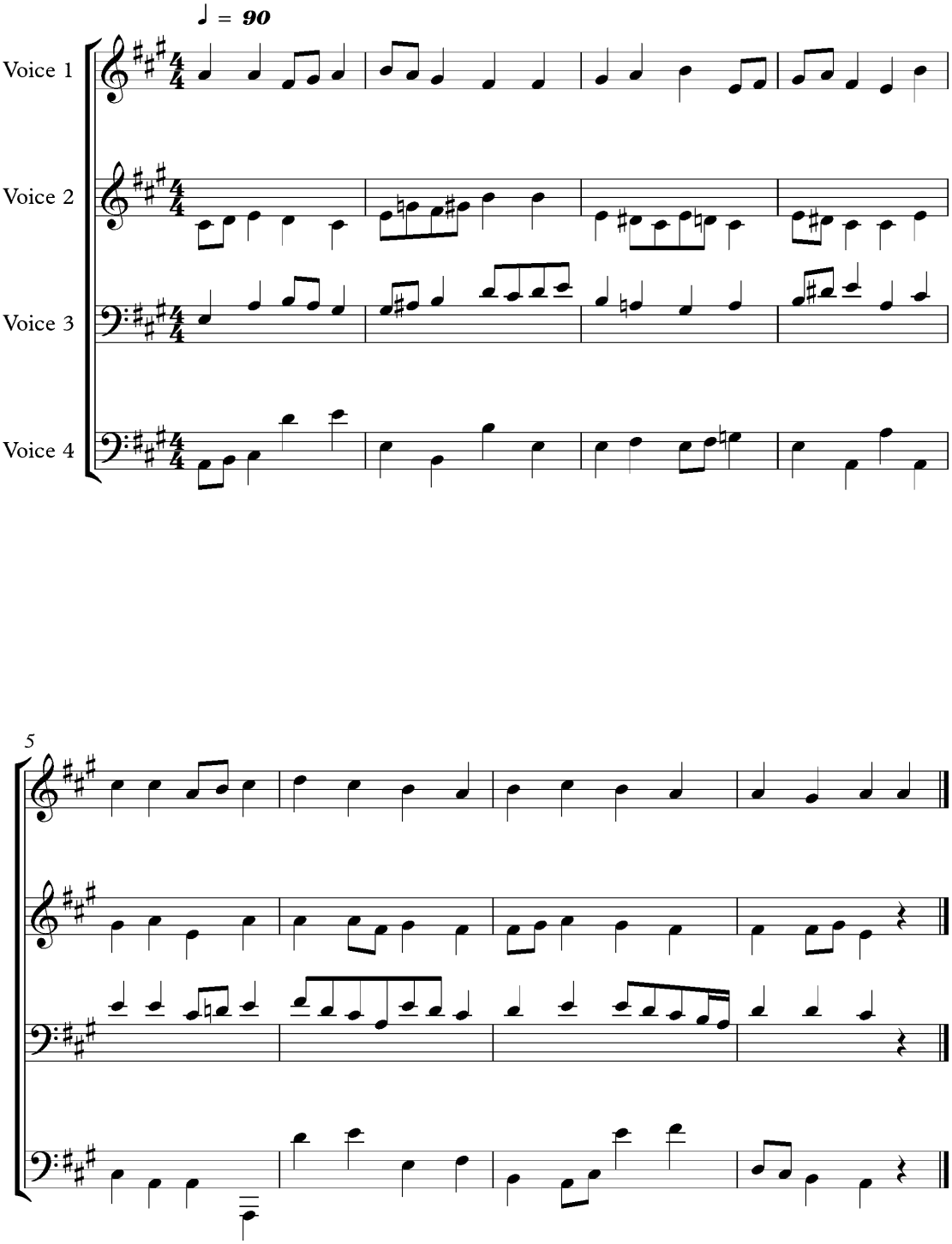


Figura 50: Experimento 4 – salida generada

6.5.4 **Análisis de los resultados**

El resultado obtenido en esta ocasión es claramente mejor que el obtenido en el experimento anterior. Nuevamente la sustitución melódica introduce elementos extraños en la armonía, pero en esta ocasión dado el estilo de la melodía el resultado no es tan disonante.

La armonía generada presenta una sonoridad satisfactoria teniendo en cuenta las limitaciones del sistema de Aprendizaje Automático implementado. Pese a la dificultad de encontrar un sistema de ponderación de similitud adecuado se aprecia que el proceso de sustitución utilizado muestra un comportamiento deseable, otorgando una riqueza melódica y rítmica a la composición.

Salida alternativa – Generación melódica automática completa

Durante el proceso de Aprendizaje Automático basado en Instancias con sustitución rítmico-melódica, se sustituye la melodía principal por la instancia más parecida encontrada por el sistema. Esto provoca que se maneje de forma temporal una partitura completamente nueva generada a partir de extractos de otras obras unidos por similitud con el esqueleto armónico generado y la melodía de entrada.

El resultado llama la atención por dos motivos, el primero es que la melodía principal resultante no son notas al azar sino que presenta cierto sentido melódico; el segundo es que el acompañamiento produce un mejor resultado al no presentar más disonancias que las ya existentes con la melodía principal.

Obsérvese la siguiente composición:

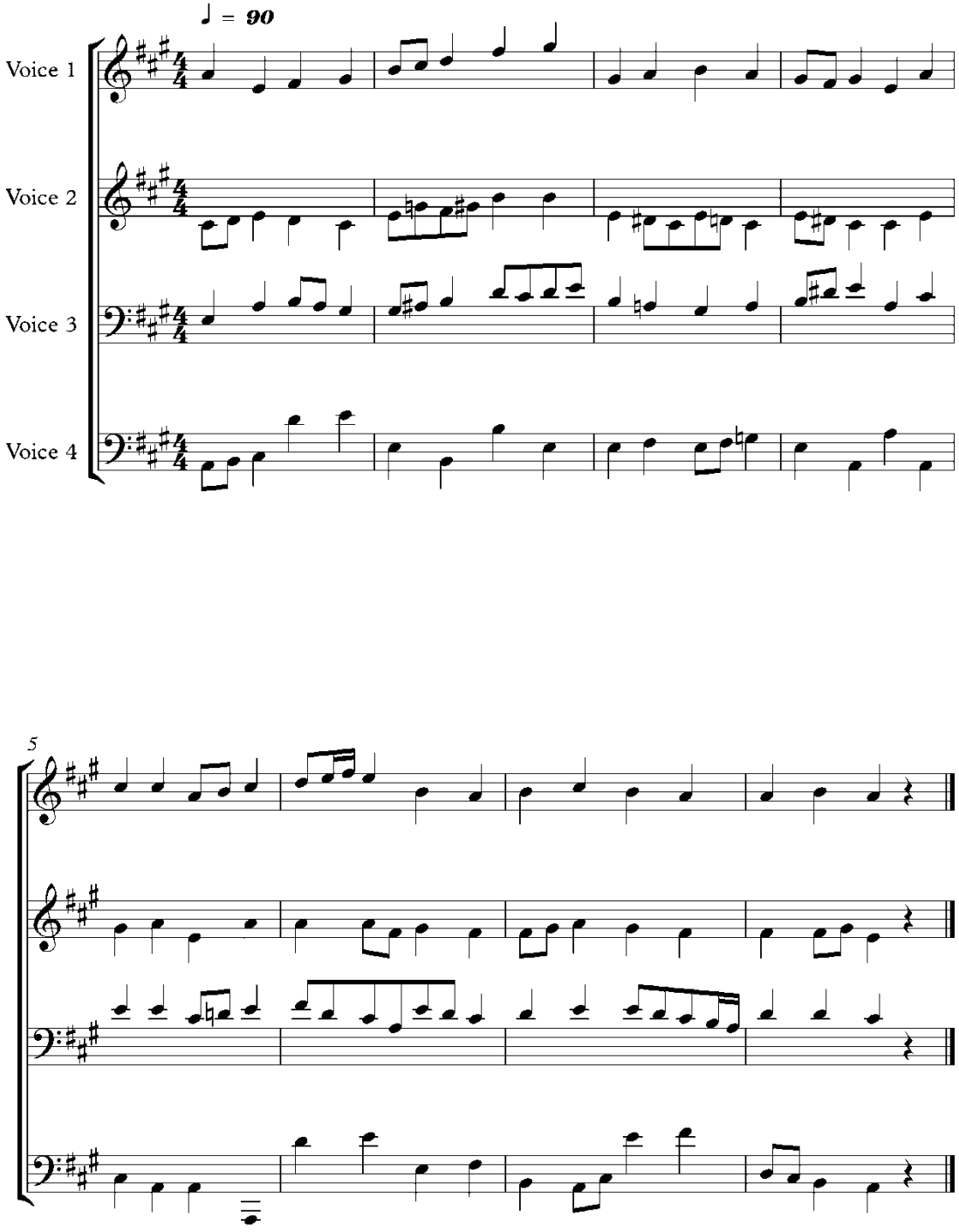


Figura 51: Anexo II – Composición automática completa generada

Si bien es cierto el resultado no está dotado de la misma calidad armónica y melódica que los obtenidos con el programa principal, presenta patrones que lo hacen una línea de investigación plausible para la generación de obras completamente nuevas.

La principal ventaja es que se parte de un esqueleto armónico de una obra similar lo que sirve de sustento para la nueva melodía. Los resultados se ven ensuciados nuevamente por la dificultad de obtener una ponderación de similitud capaz de expresar toda la complejidad musical armónica, por lo que es de suponer que perfeccionando esa parte del proceso mejorarían los resultados.

6.6 **Experimento 5**

6.6.1 **Objetivo**

El objetivo de este experimento es mostrar una serie de melodías armonizadas utilizando la configuración que mejores resultados ha dado, sirviendo de muestra de la calidad compositiva del sistema.

6.6.2 **Descripción**

Una vez analizadas distintas experimentaciones se han podido apreciar los pros y los contras de cada configuración. En este experimento se va a mostrar una serie de melodías armonizadas por el sistema con el objetivo de obtener el mejor resultado posible.

Para ello se va a utilizar un agrupamiento *TFandAAggregate* con un tamaño de grupo $n = 1$ con una modificación de los patrones rítmicos del esqueleto armónico generado por el Sistema Experto. A continuación se muestran las dos melodías a armonizar.

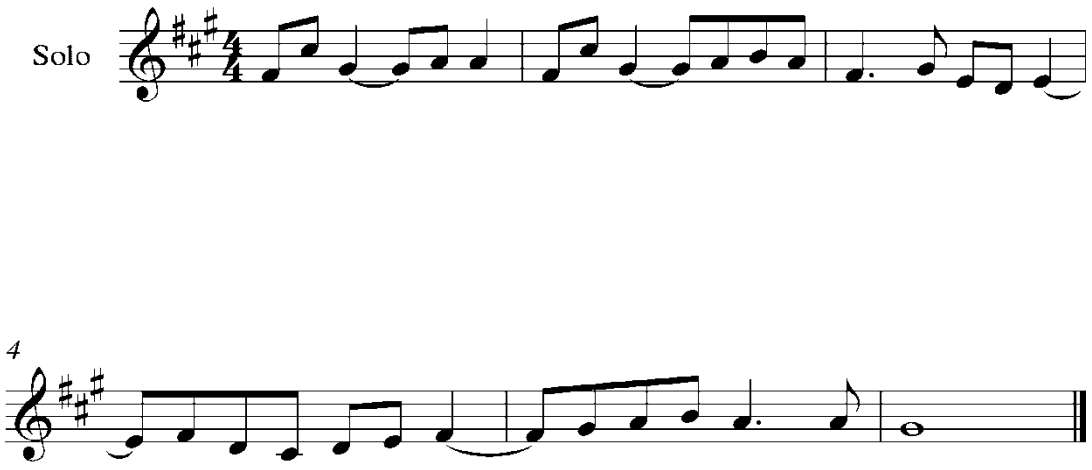


Figura 52: Experimento 5 – melodía de entrada 1

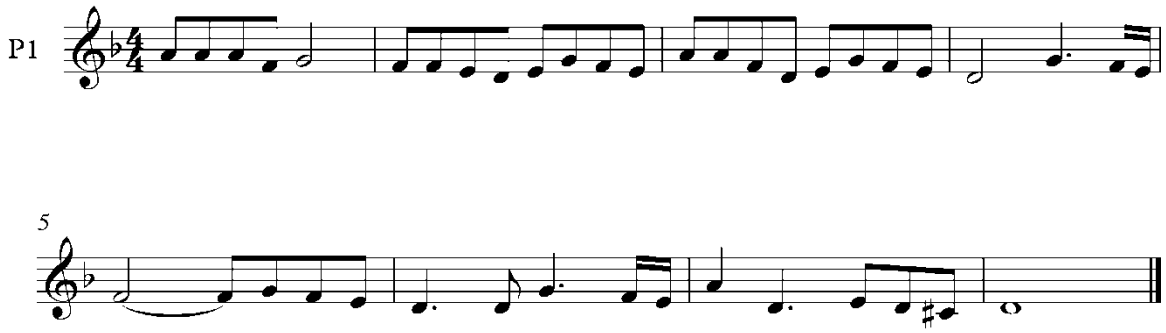


Figura 53: Experimento 5 – melodía de entrada 2

La primera de las melodías se encuentra escrita en Fa♯ menor mientras que la segunda está escrita en Re menor.

6.6.3 **Resultados**

Las melodías generadas por el sistema se presentan en el mismo orden que se presentaron en la descripción del experimento.

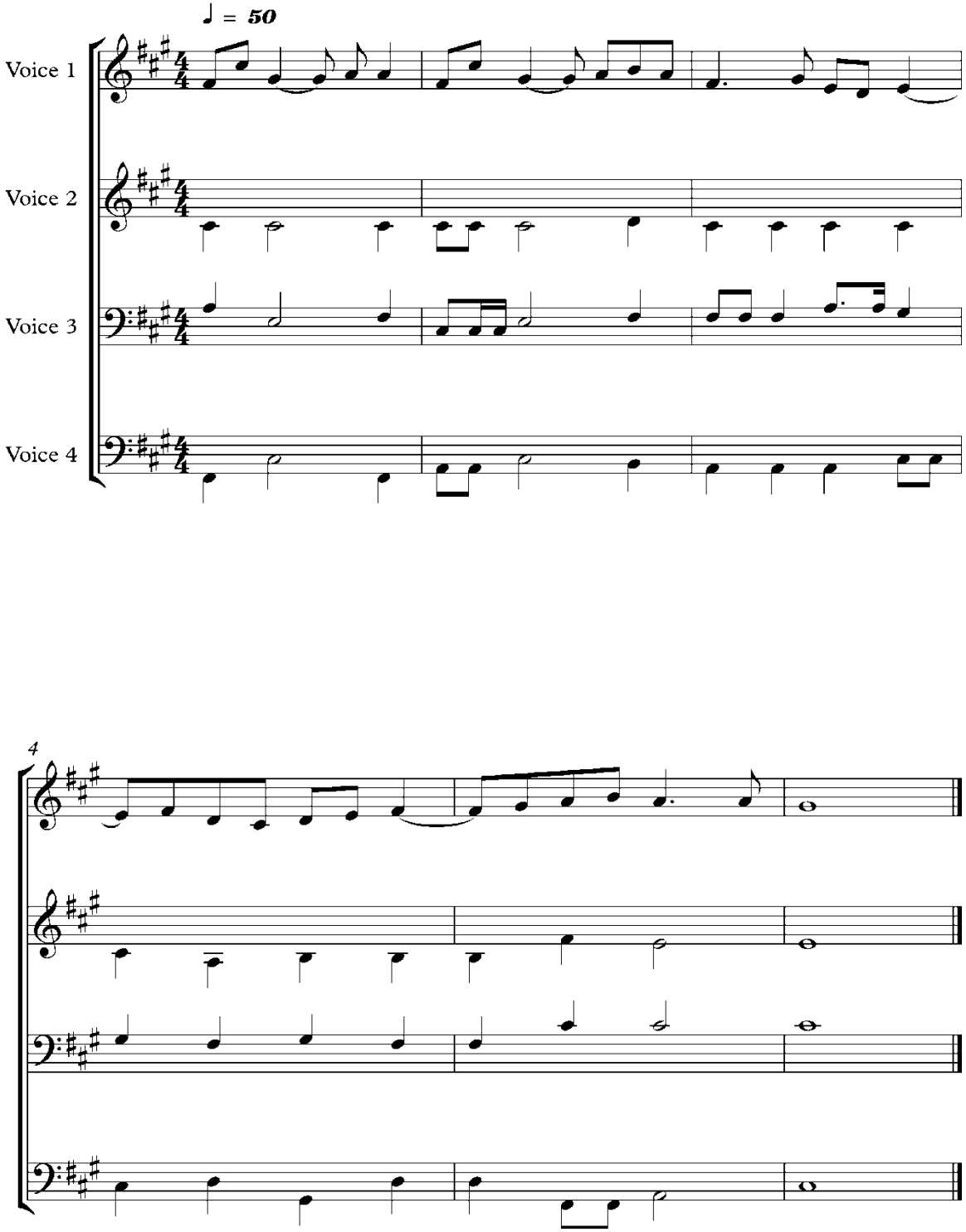


Figura 54: Experimento 5 – salida generada 1

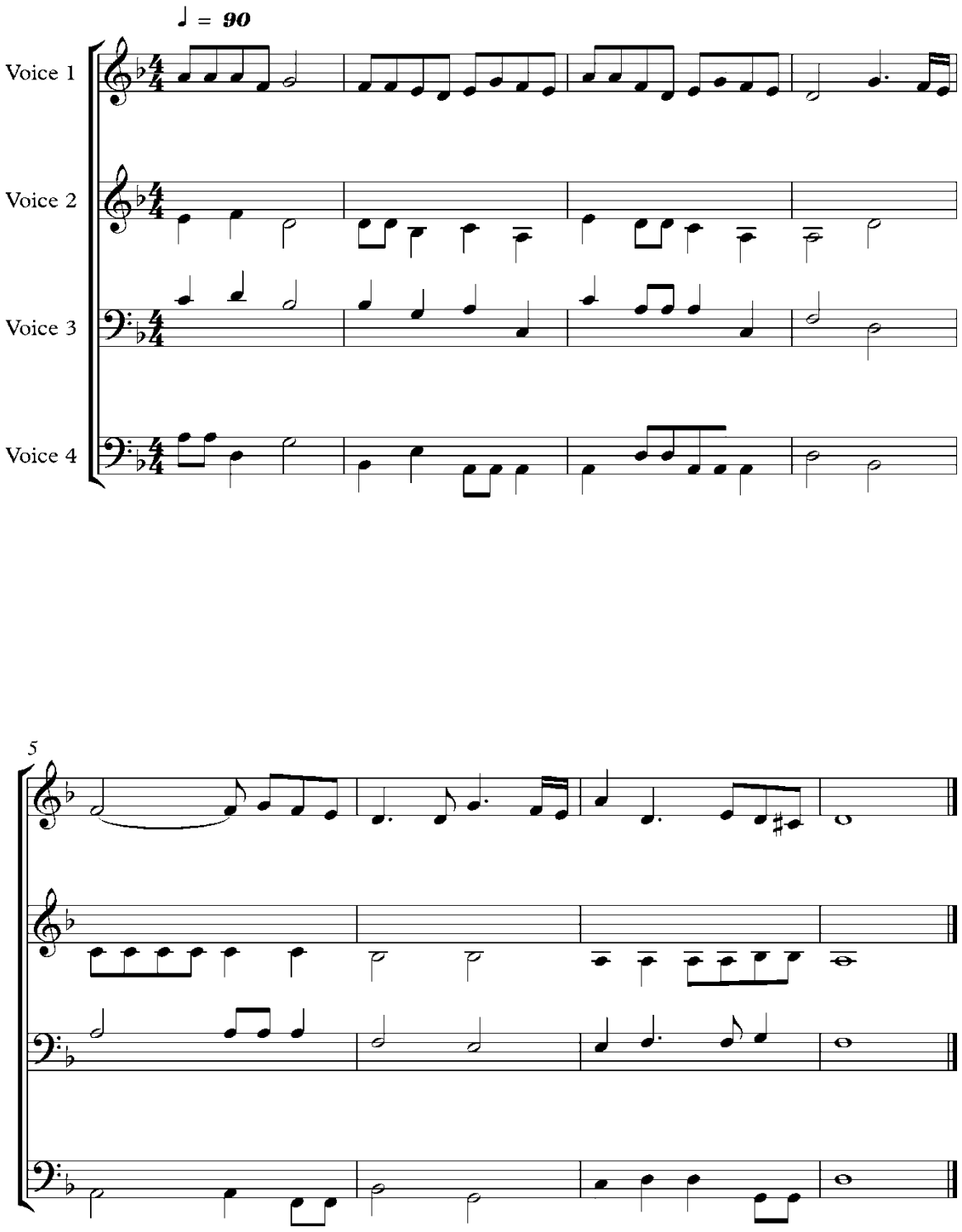


Figura 55: Experimento 5 – salida generada 2

6.6.4 **Análisis de los resultados**

Los resultados obtenidos gozan de una gran calidad. Pueden apreciarse detalles que los hacen destacar como los movimientos continuos en las voces intermedias mientras que el bajo se mueve por saltos, la variedad rítmica de cada voz y su independencia con respecto a las demás voces sin dejar de ser parte del conjunto.

Al utilizar un tamaño de agrupamiento de tamaño la unidad de pulso el esqueleto armónico de la obra generado por el Sistema Experto no se ha visto alterado lo que garantiza que la obra no incumple ninguna restricción armónica, siendo el resultado a parte de sonoramente agradable bien construido.

Capítulo 7

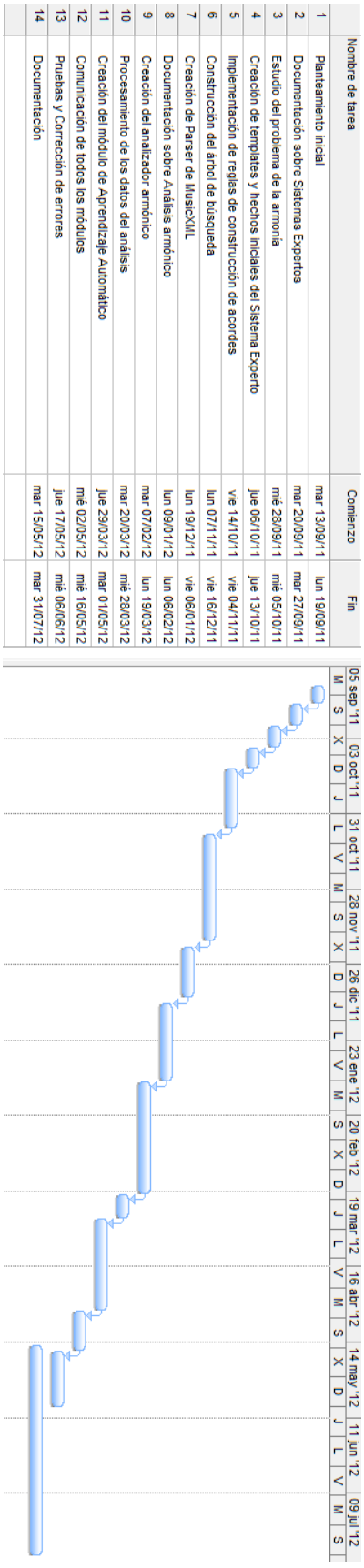
Gestión del Proyecto

7.1 Planificación del proyecto

Para la realización de este proyecto ha sido necesaria una planificación previa. El objetivo de esta planificación no es otro que dar una idea general de los recursos necesarios para realizar cada tarea, pudiendo observar las dependencias entre ellas junto con el tiempo estimado para su cumplimiento de modo que faciliten la optimización de recursos.

En esta sección se van a mostrar dos planificaciones: una planificación inicial con la estimación planteada al comienzo del proyecto y una planificación final realizada a mitad de proyecto ajustada para tener en cuenta una serie de retrasos producidos en la primera etapa, quedando esta última planificación como la seguida hasta el final del proyecto.

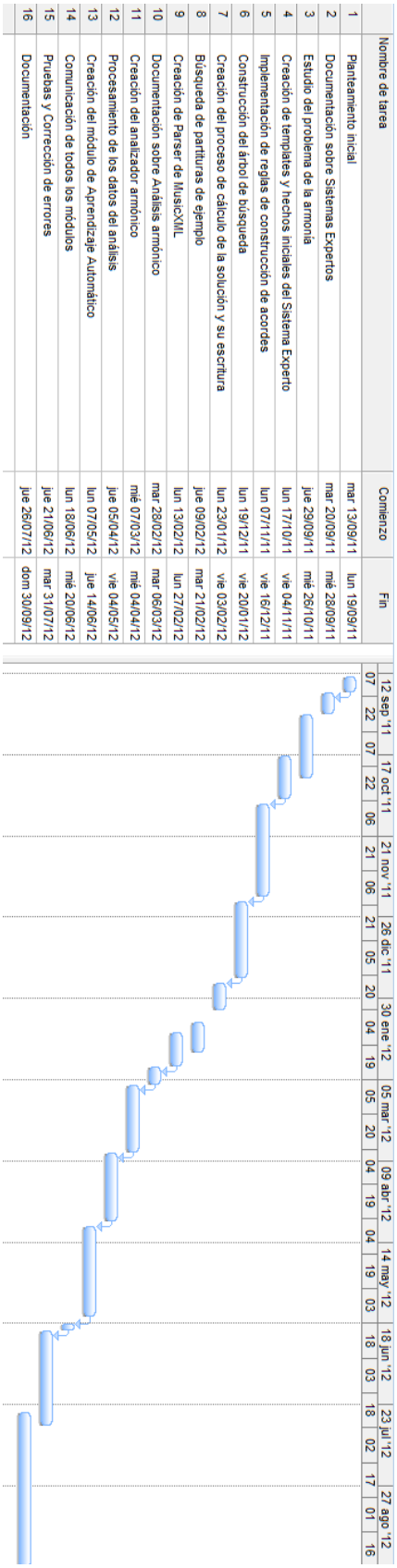
7.1.1 **Planificación inicial**



	Nombre de tarea	Comienzo	Fin
1	Planteamiento inicial	mar 13/09/11	lun 19/09/11
2	Documentación sobre Sistemas Expertos	mar 20/09/11	mar 27/09/11
3	Estudio del problema de la armonía	mié 28/09/11	mié 05/10/11
4	Creación de templates y hechos iniciales del Sistema Experto	jue 06/10/11	jue 13/10/11
5	Implementación de reglas de construcción de acordes	vie 14/10/11	vie 04/11/11
6	Construcción del árbol de búsqueda	lun 07/11/11	vie 16/12/11
7	Creación de Parser de MusicXML	lun 19/12/11	vie 06/01/12
8	Documentación sobre Análisis armónico	lun 09/01/12	lun 06/02/12
9	Creación del analizador armónico	mar 07/02/12	lun 19/03/12
10	Procesamiento de los datos del análisis	mar 20/03/12	mié 28/03/12
11	Creación del módulo de Aprendizaje Automático	jue 29/03/12	mar 01/05/12
12	Comunicación de todos los módulos	mié 02/05/12	mié 16/05/12
13	Pruebas y Corrección de errores	jue 17/05/12	mié 06/06/12
14	Documentación	mar 15/05/12	mar 31/07/12

Figura 56: Planificación inicial

7.1.2 **Planificación final**



	Nombre de tarea	Comienzo	Fin
1	Planteamiento inicial	mar 13/09/11	lun 19/09/11
2	Documentación sobre Sistemas Expertos	mar 20/09/11	mié 28/09/11
3	Estudio del problema de la armonía	jue 29/09/11	mié 26/10/11
4	Creación de templates y hechos iniciales del Sistema Experto	lun 17/10/11	vie 04/11/11
5	Implementación de reglas de construcción de acordes	lun 07/11/11	vie 16/12/11
6	Construcción del árbol de búsqueda	lun 19/12/11	vie 20/01/12
7	Creación del proceso de cálculo de la solución y su escritura	lun 23/01/12	vie 03/02/12
8	Búsqueda de partituras de ejemplo	jue 09/02/12	mar 21/02/12
9	Creación de Parser de MusicXML	lun 13/02/12	lun 27/02/12
10	Documentación sobre Análisis armónico	mar 28/02/12	mar 06/03/12
11	Creación del analizador armónico	mié 07/03/12	mié 04/04/12
12	Procesamiento de los datos del análisis	jue 05/04/12	vie 04/05/12
13	Creación del módulo de Aprendizaje Automático	lun 07/05/12	jue 14/06/12
14	Comunicación de todos los módulos	lun 18/06/12	mié 20/06/12
15	Pruebas y Corrección de errores	jue 21/06/12	mar 31/07/12
16	Documentación	jue 26/07/12	dom 30/09/12

Figura 57: Planificación final

La diferencia existente entre la planificación inicial y la final se debe principalmente a una subestimación en la complejidad de ciertas tareas al inicio del proyecto, y a la búsqueda de partituras que sirvieran de ejemplo para el Sistema de Aprendizaje Automático basado en Instancias.

La planificación final tiene en cuenta estos aspectos y recalcula el fin del proyecto en base a las deficiencias vistas en la planificación inicial.

7.2 **Medios técnicos empleados**

7.2.1 **Hardware**

Para la realización de este proyecto ha sido necesaria la utilización de equipo informático, concretamente un ordenador portátil Packard Bell TS44-HR 244SP valorado en 544€.

7.2.2 **Software**

Con el objetivo de alcanzar el mejor resultado posible se ha utilizado diverso software musical. Para la lectura y escritura de ficheros MusicXML se ha utilizado el editor musical *Sibelius 7*, que ha permitido comprobar que las salidas del sistema estaban generadas correctamente.

Para mejorar la experiencia auditiva, acercando el resultado de la reproducción musical al que se percibiría escuchando una interpretación humana con el fin de que el resultado no se viese afectado por la calidad de reproducción por defecto del sistema, se han utilizado distintos bancos de sonidos así como un secuenciador:

- *EWQSL SO Gold*: es un banco de sonidos de alta calidad de los distintos instrumentos que componen una orquesta.
- *EWQSL Symphony Choirs*: recrea las voces que componen un coro así como múltiples efectos vocales en un banco de sonidos de alta calidad.
- *Kontakt 5*: un secuenciador compatible con *Sibelius 7* que permite gestionar la salida de los canales MIDI para que utilicen los bancos de sonido anteriormente citados.

7.3 **Análisis económico**

7.3.1 **Presupuesto**

En esta sección se detalla el presupuesto realizado para este proyecto

1.- Autor:

José Ángel Canabal Delgado

2.- Departamento:

Departamento de Informática - UC3M

3.- Descripción del Proyecto:

- Titulo **Sistema Inteligente de composición armónica**
- Duración (meses) **12**
- Tasa de costes Indirectos: **20%**

4.- Presupuesto total del Proyecto (valores en Euros):

32.766,93 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Sergio Jiménez Celorrio		Ingeniero Senior	0,5	4.289,54	2.144,77	
Tomás de la Rosa Turbides		Ingeniero Senior	0,5	4.289,54	2.144,77	
José Ángel Canabal Delgado		Ingeniero	8,3	2.694,39	22.363,44	
		Hombres mes	**9,3**	**Total**	26.652,98	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d)]
Equipo Informático	540,00	100	12	60	108,00
Licencia Sibelius	505,00	100	12	60	101,00
Licencia Kontakt 5	379,00	100	12	60	75,80
Licencia EWQSL SO Gold	495,00	100	12	60	99,00
Licencia EWQSL S Choirs	495,00	100	12	60	99,00
					0,00
				Total	482,80

[d)] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} xCxD$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado

B = periodo de depreciación (60 meses)

C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
No procede		
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e)]

Descripción	Empresa	Costes imputable
Libro de armonía Walter Piston		40,00
Imprenta		100,00
CD/DVD		30,00
	Total	170,00

[e)] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,…

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales (€)**
Personal	26.653
Amortización	483
Subcontratación de tareas	0
Costes de funcionamiento	170
Costes Indirectos	5.461
Total (€)	32.767

El presupuesto total de este proyecto asciende a la cantidad de 32767€ + IVA

Leganés a 1 de Octubre de 2012
El ingeniero proyectista

Fdo: José Ángel Canabal Delgado

Capítulo 8

Conclusiones

A lo largo del presente proyecto se ha desarrollado un sistema capaz de generar una armonía completa a cuatro voces a partir de una melodía dada en formato MusicXML utilizando para ello un Sistema Experto que hace uso de los tratados y reglas enseñados en la teoría de la armonización musical. Además el sistema es capaz de modificar satisfactoriamente el acompañamiento generado mediante un sistema de Aprendizaje Automático basado en Instancias.

Se ha logrado de forma satisfactoria gestionar la entrada y salida del sistema, pudiendo leer y escribir una partitura en MusicXML.

El Sistema Experto es capaz de generar esqueletos armónicos completos en cualquier tonalidad y compás, con unos resultados muy superiores a otras técnicas de armonización al hacer un uso exhaustivo del conocimiento del dominio, soportando acordes tríada en estado fundamental y primera inversión.

Además el Sistema Experto ha sido creado por módulos, escritos en CLIPS para facilitar la ampliación de la funcionalidad en un futuro, pudiendo soportar nuevas reglas y acordes correspondientes a estilos distintos de la armonía tradicional, pudiendo centrarse si así se desea en el dodecafonismo tan frecuente hoy día.

Se ha conseguido realizar una modificación rítmica del esqueleto armónico de la obra proporcionando una riqueza musical significativa, haciendo cada voz independiente sin perder la cohesión armónica.

Se ha podido observar que existen dificultades en el proceso de ponderación de la similitud entre dos obras, lo que hace que la sustitución melódico-rítmica no obtenga un resultado de la misma calidad que la sustitución rítmica al producirse disonancias inesperadas. Del mismo modo el sistema es muy sensible a las partituras de ejemplo usadas por lo que es recomendable dedicar suficiente tiempo a elegir un buen repertorio.

El único requisito para que una partitura sea utilizada como parte de la base de ejemplos es que esta sea un fichero MusicXML bien formado a cuatro voces, lo que facilita su elección e incluso posibilita que un autor incluya sus propias obras como ejemplos.

Es destacable también la aproximación realizada a la modificación melódica de una armonía, siendo una vía de investigación con mucho potencial para conseguir resultados más naturales y similares a las composiciones humanas.

Por lo tanto, se han conseguido cumplir todos los objetivos propuestos en un inicio, obteniendo unos resultados que hacen ver que la música generada por ordenador es tan válida como la que escribiría una persona.

En mi opinión, este trabajo se une a muchos otros que antes han intentado abordar este problema, aportando una idea original que sirva como base para seguir avanzando por los caminos que se van abriendo.

Capítulo 9

Líneas Futuras

Pese a que se han cumplido los objetivos y el resultado obtenido es de gran calidad, yendo un paso por delante de muchos trabajos de este tipo, existe un buen número de mejoras que no han podido implementarse por limitaciones temporales y presupuestarias que caracterizan a un proyecto de este tipo.

El principal objetivo a corto plazo es continuar mejorando la composición para hacerla más similar a la que realizaría una persona, para ello se proponen algunas mejoras:

- Utilizar técnicas de aprendizaje automático supervisado para inferir de forma más certera las distintas progresiones armónicas en el sistema experto.
- Permitir a un usuario puntuar cada fragmento de la obra y ésta en su conjunto de modo que el sistema pueda adaptarse a los gustos del usuario en las composiciones.
- Mejorar el árbol de búsqueda del Sistema Experto armonizador. Sería interesante explorar varios niveles del árbol antes de decidir qué acorde es válido en lugar de basarnos en el acorde inmediatamente anterior.
- Tratar de obtener un método más adecuado para cuantificar la calidad de una partitura de forma computacional sin requerir de un usuario experto para ello.

- Mejorar el sistema de aprendizaje automático basado en instancias. Actualmente es muy limitado, funcionando de forma satisfactoria únicamente con los patrones rítmicos. Esto obedece a la dificultad de comparar la equivalencia entre dos melodías para que no se produzcan disonancias.

- Ampliar el número de acordes que el Sistema Experto puede crear permitiendo al usuario elegir qué tipos de acorde desea que se utilicen en su armonización.
- Crear un analizador musical que infiera el fraseo de una obra, teniendo en cuenta esa información para mejorar el proceso de armonización (aparición de cadencias en momentos clave).
- Crear una interfaz de usuario que permita visualizar las partituras, crearlas y armonizarlas de forma sencilla para el usuario. En su defecto investigar la posibilidad de crear un plugin para algún editor musical.

Referencias

[Hil] *L. Hiller, L. Isaacson, Musical Composition with a High-Speed Digital Computer, 1993, from Machine Models of Music by S. M. Schwanauer, D. A. Levitt, MIT Press, 9-21*

[Moo] *J. A. Moorer, Music and Computer Compostion, 1993, from Machine Models of Music by S. M. Schwanauer, D. A. Levitt, MIT Press, 167-186*

[Lev] *D. A. Levitt, A Representation for Musical Dialects, 1993, from Machine Models of Music by S. M. Schwanauer, D. A. Levitt, MIT Press, 455-469*

[Rad] R. López de Mántaras, J. L. Arcos, AI and Music. From Composition to Expressive Performance,2006, AI Magazine,2-2

[Cli] *Clips JNI, uso de Clips desde Java,* Disponible [Internet] <http://clipsrules.sourceforge.net/CLIPSJNIBeta.html> <http://clipsrules.sourceforge.net/WhatIsCLIPS.html> [14 de octubre de 2012]

[KMM] *Karsten Verbeurgt, Michael Dinolfo, and Mikhail Faye*, Extracting Patterns in Music for Composition via Markov Chains, *Lecture notes in computer science Volme 3029, 2004*

[ALM] *ARCOS, J. L., DEMANTARAS, R. L., AND SERRA, X. 1997. SaxEx: A Case-based reasoning system for generating expressive musical performances. In Proceedings of 1997 International Computer Music Conference, Thessalonikia, Greece, September 1997, P. R. Cook, Ed. ICMA, San Francisco, CA, 329–336*

[Ban] *Band in a Box, software de composición musical,* Disponible [Internet] <http://www.pgmusic.com/> [14 de octubre de 2012]

[Har] *Harmony Builder, software de composición musical,* Disponible [Internet] <http://www.harmonybuilder.com/> [14 de octubre de 2012]

[Ton] *Tonica fugata, software de composición musical,* Disponible [Internet] <www.capella-software.com/tonica.cfm> [14 de octubre de 2012]

[Rot] *R. López de Mántaras, J. L. Arcos, AI and Music. From Composition to Expressive Performance*,2006, AI Magazine,3-3

[Bah] *R. López de Mántaras, J. L. Arcos, AI and Music. From Composition to Expressive Performance*,2006, AI Magazine,3-3

[Cop] *R. López de Mántaras, J. L. Arcos, AI and Music. From Composition to Expressive Performance,*2006, AI Magazine,4-4

[Ebc] *R. López de Mántaras, J. L. Arcos, AI and Music. From Composition to Expressive Performance*,2006, AI Magazine,3-3

[Mus] *MusicXML, lenguaje de notación musical,* Disponible [Internet] <http://www.makemusic.com/musicxml> [14 de octubre de 2012]

[IBL] *Aprendizaje Automático Basado en Instancias, Tom Mitchell. Machine Learning, capítulo 8,* McGraw-Hill 1997. *David Aha. Lazy Learning*. Kluwer Academic Publishers, 1997.

[AA] *Aprendizaje Automáico, Mitchell, T. (1997). Machine Learning,* McGraw Hill. ISBN 0-07-042807-7

[Sol] *Set Theory Data Primer for Music, Larry Solomon,* Disponible [Internet] <http://solomonsmusic.net/setheory.htm> [14 de octubre de 2012]

[Bar] *Curso de Armonía,* Michel Baron, Disponible [Internet] <http://michelbaron.phpnet.us/armonia.htm> [14 de octubre de 2012]

[Pis] *Armonía, Walter Piston. Machine Learning, capítulo 8,* Labor 1998. ISBN: 1-58045-935-8

[For] *Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", Charles Forgy,* Artificial Intelligence, 19, pp 17-37, 1982

[CA] *Sistemas Expertos,* E. Castillo y E. Alvarez, Ed. Paraninfo, S.A., Madrid, 1989

[Aha] *Lazy Learning. Kluwer, David Aha,* Academic Publishers, 1997

Anexo I

Teoría Musical

AI.1 Introducción

En este apartado se comentarán los conocimientos musicales básicos que el sistema debe conocer. Toda la información ha sido contrastada y verificada en distintas fuentes [Bar][Pis]

AI.2 Conceptos básicos

AI.2.1 **Nota**

Una nota es un sonido determinado por una vibración cuya frecuencia fundamental es constante. Para su representación en el pentagrama se utilizan las figuras.

Su notación varía dependiendo del país, siendo las utilizadas en este trabajo la notación tradicional en español y la notación internacional

Do Re Mi Fa Sol La Si

C D E F G A B

Figura 58: Notación tradicional (arriba) e internacional (abajo)

Alteraciones

Una nota puede ver alterado su timbre ascendente o descendentemente un semitono. Para ello se hace uso de las alteraciones, un signo que modifica la altura tonal de una nota. Existen distintas alteraciones siendo las más comunes las siguientes:

- Bemol (♭): disminuye el tono de la nota un semitono.
- Doble bemol (𝄫): disminuye el tono de la nota un tono.
- Sostenido (♯): aumenta el tono de la nota un semitono.
- Doble sostenido (𝄪): aumenta el tono de la nota un tono.
- Becuadro (♮): anula el efecto de cualquier alteración sobre la nota.

Las alteraciones provocan la aparición de enarmonías, representación gráfica diferente para dos sonidos idénticos, así por ejemplo un si♯ suena igual que un do. Tal y como se verá más adelante la enarmonía juega un papel muy importante en la armonía y su grafía.

Figura musical

Una figura musical es un signo que representa gráficamente la duración del sonido de una nota.

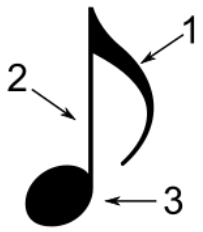


Figura 59: Partes de una nota: 1-corchete, 2-plica, 3-cabeza

La duración relativa de una nota viene dada por la forma de la figura que variará sus partes dependiendo de su duración. Las figuras más utilizadas son siete: redonda, blanca, negra, corchea, semicorchea, fusa y semifusa.

Nombre	Figura	Silencio	Valor
Redonda			1

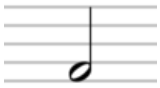
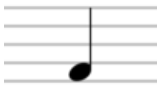
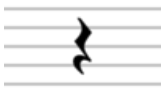
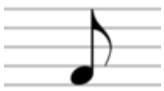
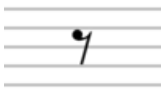
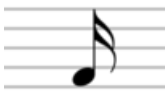
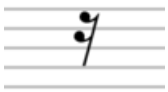
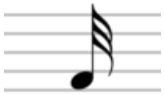
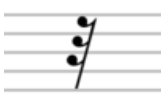
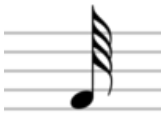
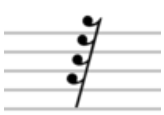
Nombre	Figura	Silencio	Valor
Blanca			$\frac{1}{2}$
Negra			$\frac{1}{4}$
Corchea			$\frac{1}{8}$
Semicorchea			$\frac{1}{16}$
Fusa			$\frac{1}{32}$
Semifusa			$\frac{1}{64}$

Tabla 36: Correspondencia figura - valor

Como se aprecia en la tabla el valor de cada nota es el doble de la siguiente, de este modo se consigue poder expresar cualquier duración como una combinación de otro tipo de figuras de duración menor.

Cada figura por su parte puede estar acompañada de uno o más puntillos que prolongan su duración. El caso más usado es el uso de un único puntillo que prolonga la duración de la figura en la mitad.

Figura 60: Negra con puntillo. Duración 1/4 + 1/8

AI.2.2 **Compás**

Entidad métrica musical compuesta por varias unidades de tiempo. Se delimita por líneas verticales que cortan el pentagrama. Se indica al principio del pentagrama con una fracción que indica la duración que tiene el compás. El denominador especifica el tipo de

figura que nos da la duración base, y el numerador el número de figuras que forman la duración total del mismo.

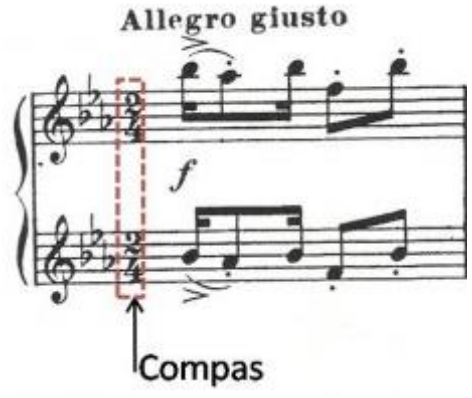


Figura 61: Ejemplo de compás de dos por cuatro

AI.2.3 **Tonalidad**

Indica cual es la nota fundamental en torno a la cual "giran" las frases y progresiones musicales especificando una organización jerárquica de las relaciones entre las diferentes alturas en función de la consonancia sonora con respecto a dicha nota fundamental o centro tonal.

Cada una de las siete notas o intervalos de una escala diatónica (escala mayor o menor) tienen una relación entre ellas. Como se ha indicado antes, la nota fundamental en torno a la que giran las demás es la tónica, que a su vez da nombre a la tonalidad. Cada nota recibe un nombre o grado según su posición con respecto a la tónica.

- I (primer grado): tónica
- II (segundo grado): supertónica
- III (tercer grado): mediante o modal
- IV (cuarto grado): subdominante
- V (quinto grado): dominante
- VI (sexto grado): superdominante o submediante
- VII (séptimo grado): sensible si se encuentra a una distancia de semitono de la tónica, o subtónica si se encuentra a una distancia de un tono de la tónica.

Los grados de mayor importancia armónica son la dominante, encargada de crear tensión, y la tónica con un carácter resolutivo.

Existen múltiples tonalidades, pero nos centraremos en aquellas pertenecientes a escalas diatónicas, las mayores y menores.

- Una tonalidad se considera mayor cuando existe una distancia de semitono entre el III y IV grado, y entre el VII y la tónica o I grado.

T T ST T T T ST

Figura 62: Disposición tonal en el modo mayor

- Una tonalidad se considera menor cuando existe una distancia de semitono entre el II y III grado, y entre V y VI grado.

T ST T T ST T T

Figura 63: Disposición tonal en el modo menor

Dependiendo de cuál sea el valor de tónica, el cual da nombre a la tonalidad, deberán existir una serie de alteraciones para cumplir dichas restricciones que formarán parte de la armadura[6] de la tonalidad.

A continuación se muestran las posibles tonalidades en modo mayor y menor que podemos encontrarnos.

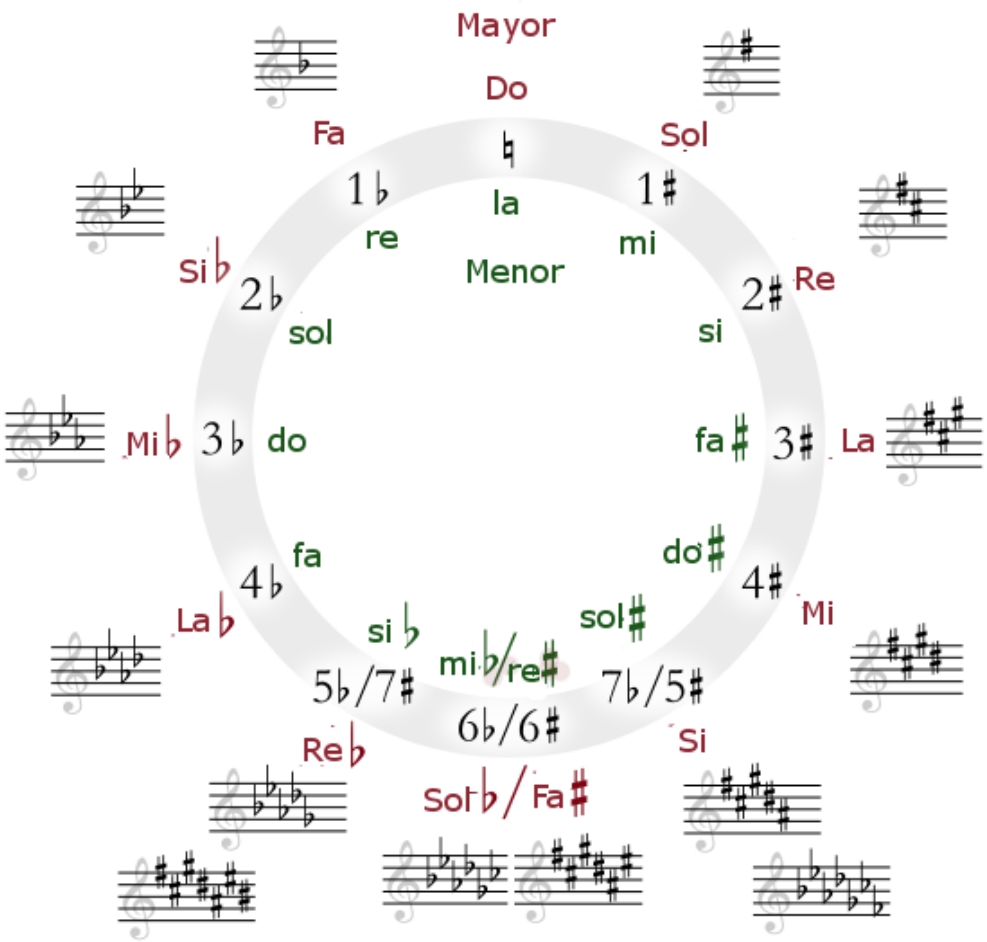


Figura 64: Círculo de quintas

[6] La armadura no es más que el conjunto de alteraciones propias que se escriben al principio del pentagrama indicando la tonalidad a la que pertenece. Todas las notas que aparecen en la armadura se interpretan alteradas a no ser que se especifique lo contrario haciendo uso de otra alteración en la obra.

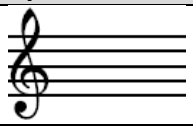
Tonalidad Mayor	Relativo menor	Armadura	Representación
doM	lam	0	

Tabla 37: Tonalidades sin alteraciones

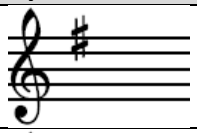
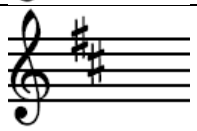
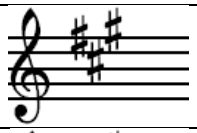
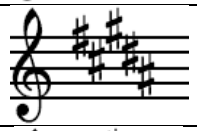
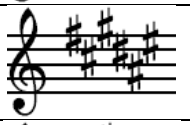
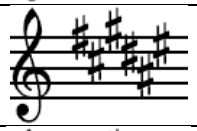
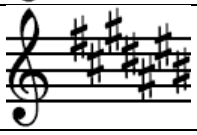
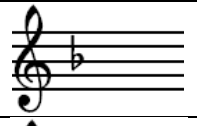
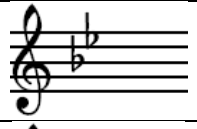
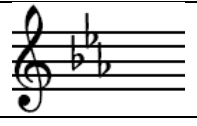
Tonalidad Mayor	Relativo menor	Armadura	Representación
solM	mim	1♯ Fa	
reM	sim	2♯ Fa, Do	
laM	fa#m	3♯ Fa, Do, Sol	
miM	do#m	4♯ Fa, Do, Sol, Re	
siM	sol#m	5♯ Fa, Do, Sol, Re, La	
fa#M	re#m	6♯ Fa, Do, Sol, Re, La, Mi	
do#M	la#m	7♯ Fa, Do, Sol, Re, La, Mi, Si	

Tabla 38: Tonalidades con sostenidos

Tonalidad Mayor	Relativo menor	Armadura	Representación
faM	rem	1 ♭ Si	
sibM	solm	2 ♭ Si, Mi	
mibM	dom	3 ♭ Si, Mi, La	

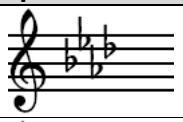
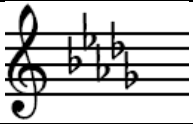
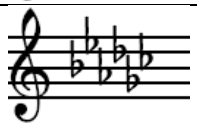
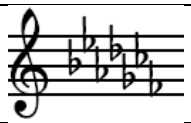
Tonalidad Mayor	Relativo menor	Armadura	Representación
labM	fam	4 ♭ Si, Mi, La, Re	
rebM	sibm	5 ♭ Si, Mi, La, Re, Sol	
solbM	mibm	6 ♭ Si, Mi, La, Re, Sol, Do	
dobM	labm	7 ♭ Si, Mi, La, Re, Sol, Do, Fa	

Tabla 39: Tonalidades con bemoles

Se puede observar como cada tonalidad mayor tiene una relativa menor con la misma armadura tonal. La diferencia entre una y otra radica en la función tonal de cada nota. Poder distinguir en qué tonalidad está escrita una obra es realmente importante a la hora de realizar un análisis o una armonización sobre ella.

AI.2.4 **Intervalo**

En música, un intervalo es la diferencia de altura (frecuencia) entre dos notas musicales, medida cuantitativamente en grados y cualitativamente en tonos o semitonos.

Según la teoría musical, existen dos grandes grupos de intervalos, los tonales y los modales. Los intervalos tonales son los de primera o únisono, cuarta, quinta y octava, mientras que los intervalos modales son los de segunda, tercera, sexta y séptima.

Los intervalos tonales tienen un solo valor, justo, mientras que los modales pueden tener un valor mayor y otro menor. Todos ellos pueden ser además aumentados o disminuidos.

Es muy importante tener en cuenta que dos intervalos sonoramente idénticos no tienen por qué ser el mismo intervalo dado que la enarmonía afecta a su notación, así pese que al intervalo do-mi y do-fa ♭ son sonoramente idénticos el primero es una tercera menor mientras que el segundo es una cuarta disminuida.

El listado completo de los tipos de intervalos según el número de grados afectado y su distancia en semitonos puede consultarse en la siguiente tabla.

Tipo	Número de grados afectados	Distancia en semitonos
Unísono / Primera Justa	1	0
Primera Aumentada / Semitono Cromático	1	1
Segunda Disminuida	2	0
Segunda Menor	2	1
Segunda Mayor	2	2
Segunda Aumentada	2	3

Tipo	Número de grados afectados	Distancia en semitonos
Tercera Disminuida	3	2
Tercera Menor	3	3
Tercera Mayor	3	4
Tercera Aumentada	3	5
Cuarta Disminuida	4	4
Cuarta Justa	4	5
Cuarta Aumentada / Tritono	4	6
Quinta Disminuida / Quinta Falsa	5	6
Quinta Justa	5	7
Quinta Aumentada	5	8
Sexta Disminuida	6	7
Sexta Menor	6	8
Sexta Mayor	6	9
Sexta Aumentada	6	10
Séptima Disminuida	7	9
Séptima Menor	7	10
Séptima Mayor	7	11
Séptima Aumentada	7	12
Octava Justa	8	12

Tabla 40: Tipos de intervalos

Dependiendo de su disposición en la partitura pueden ser de dos tipos, intervalos melódicos o intervalos armónicos. Los primeros, tal y como su nombre indica son aquellos en los que las notas son interpretadas de forma consecutiva formando parte de una melodía, mientras que el intervalo armónico es aquel que se produce en un único instante de tiempo.

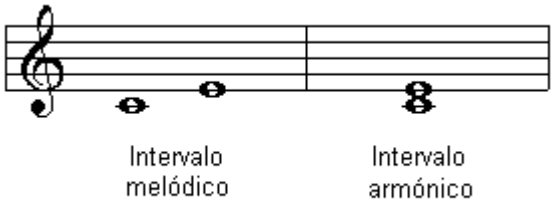


Figura 65: Intervalo melódico e Intervalo armónico.

AI.3 Armonía

En música, la armonía es la disciplina que estudia la percepción del sonido en forma «vertical» o «simultánea» en forma de acordes y la relación que se establece con los de su entorno próximo.

En este trabajo en concreto trataremos la armonía tradicional a cuatro voces.

Voz melódica

Una voz es un conjunto de notas que no se superponen, sino que suenan de forma sucesiva formando una melodía. En la armonía clásica tenemos 4 voces: soprano, contralto, tenor y bajo. Cada una de ellas tiene una tesitura[7] diferente.

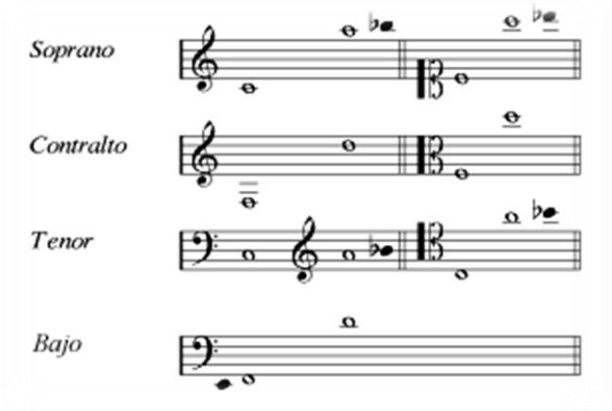


Figura 66: Tesitura de cada voz. En blanco el rango recomendable y en negro el permitido pero no deseable.

Movimientos entre las voces

Existen diversas maneras de espaciar entre ellas las cuatro voces. Siendo las más frecuentes las siguientes:

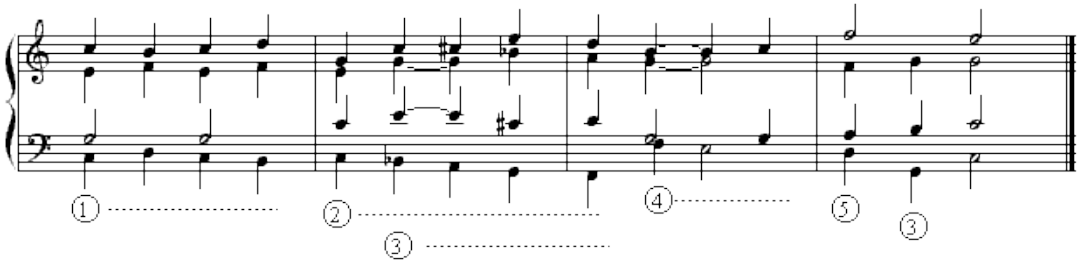


Figura 67: Movimiento entre las voces

1. Intervalos prácticamente iguales entre las diferentes partes. Esta es una disposición conocida como posición de cuarteto. Es una posición ideal que suena muy bien.

2. En este caso vemos que el bajo queda aislado mientras que las otras tres voces superiores quedan agrupadas. A veces esta posición es conocida como la posición de piano ya que el bajo queda para la mano izquierda y el resto para la mano derecha.

[7] Altura propia de cada voz o instrumento.

3. Destacar que no se debe sobrepasar en ningún momento el intervalo de octava entre ninguna de las tres voces superiores y su vecina, únicamente está permitido entre tenor y bajo.

4. Si se alcanza la octava entre dos partes intermedias las dos voces inferiores deben formar un intervalo que no sobrepase la tercera para evitar que el acorde suene hueco.

5. Las dos voces superiores pueden distanciarse hasta la octava y si el equilibrio armónico lo justifica incluso superarla.

Movimientos armónicos

Un movimiento armónico es aquel que se produce entre dos voces diferentes. Hay cuatro tipos de movimientos.

- Movimiento paralelo: las dos voces suben o bajan juntas conservando el mismo intervalo entre ellas. Lo más frecuente es un movimiento por terceras o sextas.

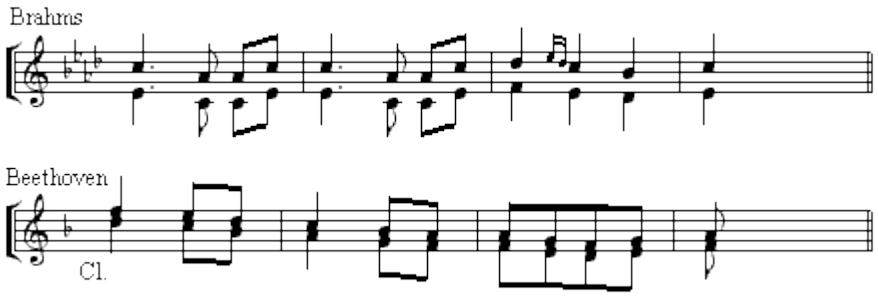


Figura 68: Movimiento paralelo

Este movimiento está prohibido en dos casos: octavas consecutivas diferentes por movimiento paralelo y en quintas consecutivas diferentes por movimiento paralelo dado que resultan muy vulgares y demasiado duros.

- Movimiento contrario: en este movimiento una de las voces sube y la otra baja o viceversa. Es un movimiento bastante frecuente ya que es bastante equilibrado y no presenta inconvenientes.

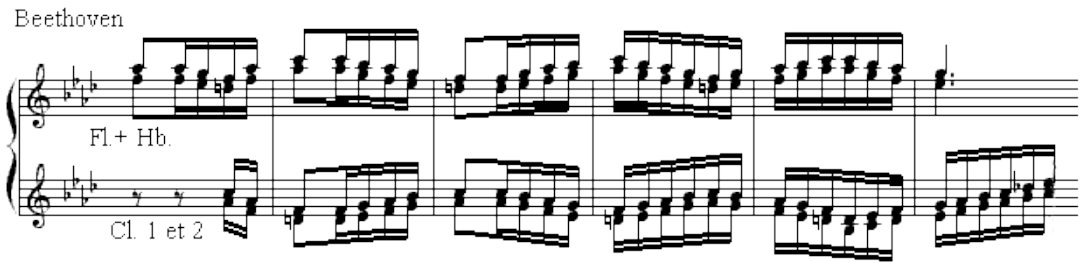


Figura 69: Movimiento contrario

Nuevamente hay que tener cuidado de que no se generen quintas y octavas consecutivas por los motivos mencionados anteriormente.

- Movimiento oblicuo: una voz se mantiene mientras que otra se aleja o se acerca. Lo único que hay que evitar es llegar a un unísono por parte de la moz que se mueve. Por el contrario resulta interesante salir de un unísono por movimiento oblicuo.

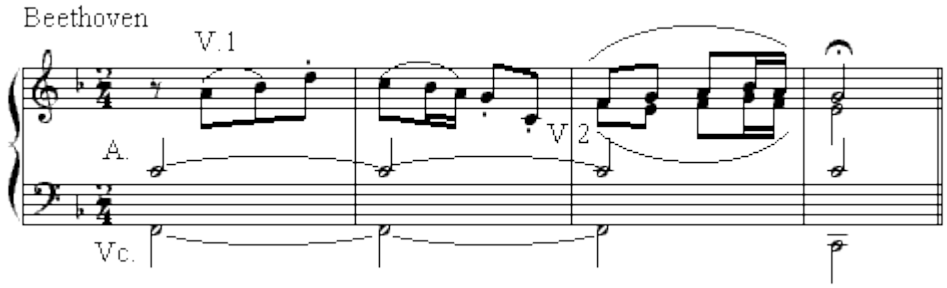


Figura 70: Movimiento oblicuo

- Movimiento directo: en este movimiento las dos partes ascienden o descienden juntas pero a diferencia del movimiento paralelo, donde se mantenía el intervalo entre ellas, se acercan o se alejan la una de la otra.

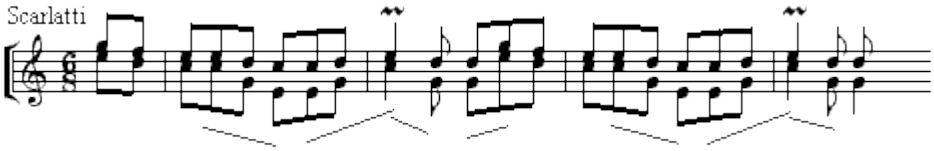


Figura 71: Movimiento directo

Este movimiento cuenta con una serie de restricciones si finaliza en una quinta o en una octava.

- Quinta u octava directa entre voces extremas: la voz soprano debe proceder por movimiento conjunto[8].
- Quinta u octava directa entre otro par de voces: una de las voces (preferiblemente la superior) debe proceder por movimiento conjunto, de no ser posible una de las notas del intervalo debe ser oída en el acorde anterior.
- Unísono directo: siempre prohibido.

AI.3.1 **Acorde**

Llamamos acorde al conjunto armónico formado por al menos tres sonidos distintos. En la música de la práctica común, además, los sonidos de un acorde deben poderse ordenar por terceras superpuestas.

[8] El intervalo formado por una nota y la anterior debe ser de segunda.

Existen varios tipos de acorde dependiendo del número de sonidos que lo formen:

- Tríada: acorde formado por tres sonidos.
- Tríada sin quinta: acorde tríada al que se le elimina la quinta.
- Séptima de dominante: acorde de cuatro sonidos cuya fundamental es la dominante de la tonalidad.
- Séptima de dominante sin fundamental: acorde de séptima de dominante donde omitimos la fundamental.
- Séptima de especies: acorde de cuatro sonidos.
- Novenas de dominante mayor: acorde mayor de cinco sonidos cuya fundamental es la dominante de la tonalidad.
- Novena de dominante mayor sin fundamental: acorde de Novena de dominante mayor donde omitimos la fundamental.
- Novenas de dominante menor: acorde menor de cinco sonidos cuya fundamental es la dominante de la tonalidad.
- Novena de dominante menor sin fundamental: acorde de Novena de dominante menor donde omitimos la fundamental.

Cada acorde cuenta con distintas inversiones[9], el número variará dependiendo del número de sonidos que lo formen, definidos por la nota del acorde colocada en el Bajo. Además cada acorde cuenta con un cifrado, que no es más que una notación que especifica su especie.

Dada la inmensidad del dominio, en este trabajo se ha decidido abarcar únicamente los acordes tríada, más concretamente en su estado fundamental y primera inversión, esto es acordes de quinta y de sexta.

- Acorde de quinta: es el acorde tríada en estado fundamental, colocado por terceras las notas del acorde están a distancia de tercera y de quinta con respecto a la nota fundamental del acorde.

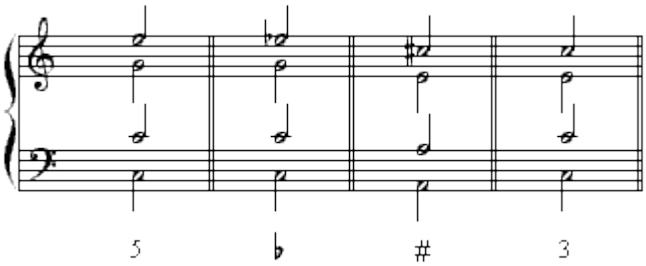


Figura 72: Acordes de quinta

Dependiendo de los intervalos que lo formen puede ser de distintos tipos:

- Mayor: formado por una tercera mayor y una quinta justa.
- Menor: formado por una tercera menor y una quinta justa.
- Aumentado: formado por una tercera mayor y una quinta aumentada.

[9] La inversión de un acorde tríada no es más que el cambio de la nota fundamental en el bajo por su tríada o su quinta.

- Disminuído: formado por una tercera menor y una quinta disminuída.

- Acorde de sexta: es la primera inversión del acorde de quinta. En otras palabras, el bajo del acorde de sexta es la tercera del mismo acorde en estado fundamental.

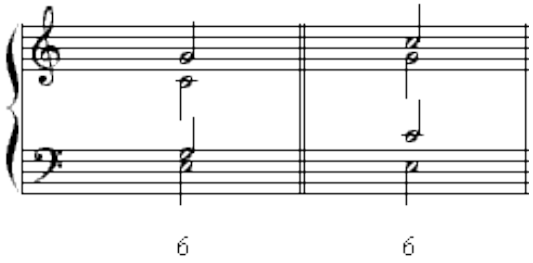


Figura 73: Acordes de sexta

AI.3.2 **Cadencias**

Las cadencias son encadenamientos de grados o funciones armónicas muy específicos que puntúan la frase musical de manera más o menos suspensiva o conclusiva.

Semicadencia

Se trata de un encadenamiento de dos acordes donde el segundo es un V grado en estado fundamental, siendo una cadencia de carácter suspensivo. Normalmente va precedida de los grados IV o II en estado fundamental o primera inversión. A veces puede ir precedido del VI o incluso del I.

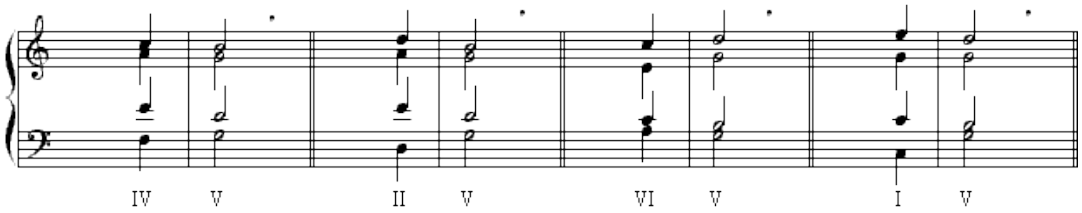


Figura 74: Semicadencias

Cadencia perfecta

La cadencia perfecta es la sucesión V-I en estado fundamental, siendo una cadencia de carácter conclusivo. Pese a que no es necesario que la voz de soprano haga la sucesión sensible tónica sí que es una posición que refuerza su carácter conclusivo.

Esta cadencia suele ir precedida de los grados IV o II en estado fundamental o primera inversión. A veces puede ir precedida del sexto y raramente del III.

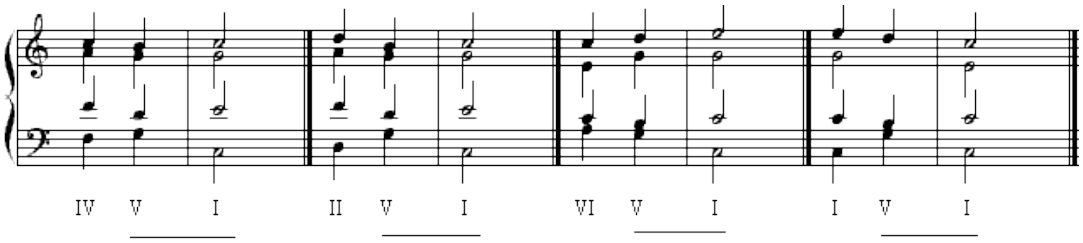


Figura 75: Cadencias perfectas

Cadencia rota

Sucesión V-VI en estado fundamental con carácter suspensivo. No hay ningún soprano obligado aunque la sucesión sensible tónica da el mejor efecto. Suele ir precedida de los mismos grados que la cadencia perfecta.

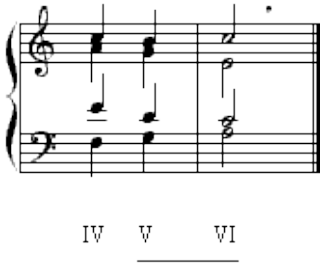


Figura 76: Cadencia rota

Cadencia plagal

La cadencia plagal, también conocida como cadencia de iglesia[10], debe su nombre a los antiguos modos plagales. Se trata de la sucesión de los grados IV- I en estado fundamental y tiene un carácter muy conclusivo. Normalmente se encuentra precedida del VI grado en estado fundamental o del I en estado fundamental o primera inversión. Puede ir precedida de una cadencia rota o cadencia perfecta reforzando así su carácter conclusivo.

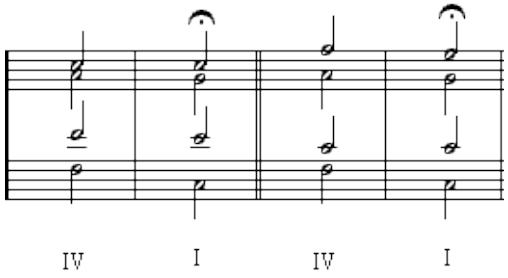


Figura 77: Cadencias plagales

Cadencias imperfectas

Una cadencia imperfecta es una cadencia en la que alguno de sus acordes se encuentra invertido siendo la más frecuente la sucesión V-I. Tiene un carácter afirmativo y se utiliza como un elemento ordinario del discurso sin tener un rol específico de puntuación.

[10] Es utilizada habitualmente en la representación musical de un Amén.

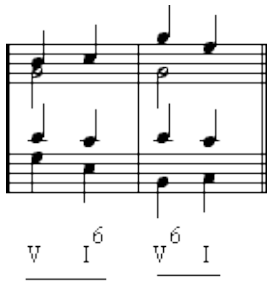


Figura 78: Cadencias imperfectas

Cadencia evitada

Se trata de una cadencia perfecta en la que el primer grado deviene a su vez de la 7ª de dominante para modular a la subdominante. No se va a tener en cuenta en este trabajo al no tratar los acordes de séptima.

AI.3.3 **Reglas**

Existen una serie de reglas que toda composición armónica clásica debe cumplir. Estas reglas se encargan de mantener la coherencia del estilo evitando vulgaridades armónicas, tratando de velar por la correcta construcción de un esqueleto armónico. Se muestran a continuación brevemente las más importantes.

AI.3.3.1 **Reglas melódicas**

Las reglas melódicas se encargan de regular el funcionamiento de cada melodía por separado, afectando a la parte horizontal de la composición armónica.

- Intervalo entre notas consecutivas: por norma general dos notas consecutivas no podrán superar la distancia interválica de octava, así como un enlace de séptima.
- Tesitura de las voces: Una voz no podrá sobrepasar los límites marcados por su tesitura.

AI.3.3.2 **Reglas armónicas**

Las reglas armónicas son aquellas que afectan a la obra de manera vertical, importando únicamente la relación de las voces entre sí.

- Cruzamiento de voces: las voces no pueden cruzarse entre sí, la voz superior no puede quedar por debajo de una voz más grave.
- Distancia entre voces: la distancia existente entre voces no debe superar la octava salvo la distancia entre el bajo y el tenor que puede alcanzar las dos octavas.

AI.3.3.3 **Reglas armónico-melódicas**

Existe un tercer grupo de reglas que regula el comportamiento armónico de forma horizontal, siendo el encargado de marcar las pautas a seguir en los enlaces de acordes.

- Quintas y Octavas seguidas: por norma general no debe existir una distancia de quinta o de octava entre las mismas voces. Como excepción se admiten las quintas seguidas siempre que la segunda sea disminuida.
- Quintas y Octavas directas: quedan prohibidas los intervalos de quinta y octava entre voces que se produzcan por movimiento directo salvo que alguna de las voces, salvo el bajo, se mueva por grados conjuntos[11].

[11] Se entiende como movimiento por grados conjuntos a aquellos que forman un intervalo de segunda.

Anexo II

Manual de Usuario

AII.1 Requisitos del Sistema

Para el correcto funcionamiento del sistema será necesario disponer de una máquina virtual de Java (JVM) en su versión 1.7 instalada en el equipo. En caso de no disponer de ella puede ser descargada desde el siguiente enlace:

- http://www.oracle.com/technetwork/java/javase/downloads/index.html

El sistema está preparado para ser ejecutado en equipos con Sistema Operativo Windows XP o superior tanto para arquitecturas de 32 como de 64 bits. De igual modo la aplicación puede ser ejecutada en sistemas Mac y Linux previa configuración de las librerías CLIPSJNI[12].

[12] Para más información sobre la ejecución en entornos Mac y Linux consultar la siguiente dirección web: http://clipsrules.sourceforge.net/CLIPSJNIBeta.html

AII.2 Instalación

La aplicación cuenta con dos versiones diferentes para ser ejecutadas en sistemas Windows, una versión específica para sistemas de 32 bits y otra versión para sistemas de 64 bits. El proceso de instalación únicamente necesita la descompresión del archivo en el directorio deseado.

```
 __data
|     |__analysis.csv
|     |__1TFandAggregate.csv
|
|__input
|     |__ejemplo1.xml
|     |__ejemplo2.xml
|     |__ejemplo3.xml
|
|__musa
|     |__acordes-triada.clp
|     |__estructura-armonica.clp
|     |__hechos-iniciales.clp
|     |__salida-mousike.clp
|     |__templates.clp
|
|__output
|__musa.jar
|__CLIPSJNI.dll
```

Figura 79: Estructura del sistema de archivos de la aplicación

AII.3 Ejecución

El sistema está pensado para generar una armonización con sustitución de los patrones rítmicos utilizando para ello la configuración que mejores resultados proporciona para garantizar soluciones de calidad.

Para realizar una ejecución es necesario disponer de la partitura a armonizar en formato MusicXML en el directorio input y únicamente ejecutar este comando una vez situados en el directorio de la aplicación:

```
java –jar musa.jar nombre_fichero_input nombre_fichero_output
```

Siendo *nombre_fichero_input* el nombre de la partitura sin la extensión del directorio *input*, y *nombre_fichero_output* el nombre que deseamos que tenga la salida, siendo este último un campo opcional cuya inexistencia generará una salida con el mismo nombre que la entrada.

El resultado de la ejecución se generará en la carpeta *output*, siendo un fichero en formato MusicXML que podremos abrir y reproducir con cualquier editor compatible como Sibelius o Finale.