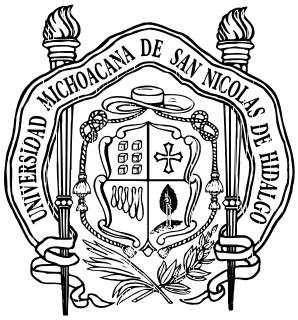


Universidad Michoacana de San Nicolás de Hidalgo

División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica

# EVALUACIÓN AUTOMÁTICA DE INTERPRETACIONES MUSICALES BASADA EN INFORMACIÓN MIDI

TESIS

Que para obtener el grado de

**Maestro en Ciencias en Ingeniería Eléctrica**

Presenta

**Ing. Jessica Díaz Estrada**

Director de Tesis

**Dr. José Antonio Camarena Ibarrola**

Morelia Michoacán, Agosto 2016.

*A mis padres José y Gloria por su cariño y enseñanzas, por haber fomentado en mí el deseo de superación y el anhelo de triunfo, siempre apoyando e impulsándome en todo momento de mi vida. Va por ustedes, por lo que valen.*
*Con todo mi amor y admiración eterna.*

*A mis hermanas Yuli y Viri por su apoyo, amistad, por compartir momentos valiosos, pero en especial porque son una razón más en mi vida.*

**Agradecimientos:**

- A mis padres, que sin escatimar esfuerzo alguno, han sacrificado gran parte de su vida para forjarme como la persona que hoy soy. Buscaron hacer todo para que mi vida sea perfecta.
- A mis abuelitos por sus sabios consejos y cariño, en general a toda mi familia que es lo mejor y más valioso que tengo.
- Al Dr. José Antonio Camarena Ibarrola, mi asesor, por brindarme la oportunidad de recurrir a su capacidad y conocimiento científico, así como la paciencia para guiarme durante el desarrollo de esta tesis.
- A la Universidad Michoacana de San Nicolás de Hidalgo especialmente a la División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica por haberme aceptado y abierto las puertas de su seno científico, así como a los diferentes docentes que proporcionaron sus conocimientos durante esta etapa.
- Al Consejo Nacional de Ciencia y Tecnología, por la beca otorgada para la realización de esta tesis.
- A mis amigos que durante la estancia en el posgrado compartimos conocimientos, desvelos, vivencias y alegrías.

# Resumen

La masificación de la enseñanza de la música está limitada por el número de profesores de música, sería altamente deseable un profesor de música que pueda multiplicarse, esto no es posible con los profesores humanos, pero si con los maestros virtuales ya que un maestro virtual es realmente un software que puede ser instalado en muchas computadoras. Un maestro virtual; al igual que los profesores humanos, necesitan para evaluar a sus estudiantes, decidir si el estudiante ya puede o no pasar a la siguiente lección, que es el tema central de esta tesis. En esta investigación el estudiante es evaluado con datos simbólicos, específicamente la información grabada en los archivos MIDI mediante un teclado musical. El enfoque en la evaluación de los estudiantes en este trabajo consiste en convertir el archivo MIDI producido por el estudiante en cadenas de caracteres largas donde la secuencia de notas, su duración e incluso la fuerza han sido concatenados. Tales cadenas largas son comparadas con otra cadena larga que corresponde a una versión "correcta" de la misma pieza musical la cual ha sido interpretada por algunos músicos de formación. Entre más se parezca la cadena generada por el estudiante a la cadena generada por el músico de formación la calificación sera más alta para el estudiante. Hay varias maneras para comparar cadenas como sabemos por la teoría de procesamiento de texto, por lo que en este trabajo se experimentó con la distancia de Levenshtein, la distancia InDel y algunas métricas diseñadas específicamente para trabajar con la información de los archivos MIDI como time-warped Longest Common Subsequence y la distancia invariante a transposición InDel. En los experimentos fueron evaluados 13 estudiantes de música y sus calificaciones comparadas con las calificaciones asignadas por un profesor de música. En este experimento la distancia invariante a transposición InDel demostró ser la más adecuada para este propósito, aunque es una distancia computacionalmente costosa, el algoritmo conocido como Branch and Bound fue utilizado para reducir el tiempo de ejecución computacional de la distancia. Las calificaciones obtenidas son similares a las asignados por el profesor, por lo tanto el sistema sería confiable e imparcial al usarse para evaluar a estudiantes de música.

**Palabras clave:** Evaluación, Comparar, Cadenas, MIDI, Distancia.

# Abstract

the massification of music teaching is limited by the number of music teachers, it would be highly desirable for a music teacher to multiply, that is not possible with human teachers but it is with virtual teachers since a virtual teacher is really a software that can be installed in many computers. A virtual teacher just as human teachers needs to evaluate their students to decide if they may or may not go to the next lesson, that is the central issue of this thesis. In this work the student is evaluated from symbolic data, specifically the information recorded in MIDI by a music keyboards. The approach in evaluating the student in this work consists of converting the MIDI file produced by the student to long strings where the sequence of notes, their duration and even strength have been concatenated. Such long strings are compared to another long string that corresponds to a "correct" version of the same music which has been played by some trained musician. The more similar the string generated by the student is to the string generated by the trained musician is the higher the grade to the student would be. There are several ways to compare strings as we know from Text processing theory, so in this work we experimented with the Levenshtein distance, the InDel distance and some metrics specifically designed for working with simbolic music such as the time-warped Longest Common Subsequence and the Transposition invariant InDel distance. In the experiments 13 music students were evaluated and their grades compared with those assigned by a music teacher. In this experiment the Transposition invariant InDel distance proved to be the most adequate for this purpose, even though it is a expensive distance to compute, an algorithm known as Branch and Bound was used to reduce the time spent in computing this distance. The grades of the students are similar enough to those assigned by the teacher so the system would reliably and impartially be used to evaluated music students.

# Contenido

# Lista de Figuras

# Lista de Tablas

# Lista de Símbolos

| | |
|---|---|
| $Hz$ | Hertz. |
| $\Sigma$ | Alfabeto. |
| $\beta$ | Conjunto de valores (transposiciones). |
| $\mathbb{Z}$ | Conjunto finito de números enteros. |
| $\sigma$ | Tamaño del alfabeto. |
| $8^{va}$ | Una octava arriba. |
| $8^{vb}$ | Una octava abajo. |
| $\#$ | Nota alterada o sostenidos. |
| $pitch$ | Valor del tono musical. |
| $a$ | Cadena de caracteres o serie de tiempo. |
| $b$ | Cadena de caracteres o serie de tiempo. |
| $S$ | Conjunto de cadenas. |
| $\mathbb{R}$ | Números reales. |
| $S \times S$ | Producto cartesiano. |
| $\in$ | Que pertenece a un conjunto de elementos. |
| $\forall$ | Para todos en un conjunto de elementos. |
| $\leq$ | Menor o igual a. |
| $=$ | Igual a. |
| $P$ | Indica patrón. |
| $T$ | Indica texto. |
| $K$ | Número limitado de diferencias. |
| $A$ | Cadena de caracteres de tamaño N. |
| $B$ | Cadena de caracteres de tamaño M. |
| $D$ | Matriz de distancias. |
| $min$ | Valor mínimo. |
| $max$ | Valor máximo. |
| $i$ | Índice. |
| $j$ | Índice. |
| $N$ | Tamaño de cadena de caracteres. |
| $M$ | Tamaño de cadena de caracteres. |
| $\neq$ | Diferencia. |
| $L$ | Matriz de longitudes. |
| $\|A\|$ | Tamaño de la cadena A. |
| $\|B\|$ | Tamaño de la cadena B. |

| | |
|---|---|
| $D_{ID}$ | Valor de distancia. |
| $DT$ | Matriz de distancias. |
| $t$ | Valor de desplazamiento. |
| $\tau$ | Valor de rango de transposición. |
| $\tau'$ | Valor de rango de transposición. |
| $\lfloor \rfloor$ | Indica el piso de un valor decimal. |
| $\lceil \rceil$ | Indica el techo de un valor decimal. |
| $\theta$ | Valor para calcular nodo hijo en árbol binario. |
| $Q$ | Cola de prioridad en árbol binario. |
| $val$ | Valor de LCTS almacenado en la cola de prioridad. |
| $ticks$ | Pulsos por segundo. |
| $dR$ | Duración real en *seg.* que una nota musical fue presionada. |
| $I$ | Inicio de nota en Ticks. |
| $F$ | Final de nota en Ticks. |
| $r$ | Valor de resolución en ticks por segundo. |
| $dM$ | Valor en cuartos de nota (se define como duración musical) . |
| $\mu s$ | Microsegundos. |
| $te$ | Valor de tempo en $\mu s$. sobre cuartos de nota. |
| $p$ | Identificar la posición en una lista dinámica. |
| $x$ | Valor de parámetro para ponderar calificación de 0-10. |
| $E$ | Número de estudiantes. |
| $Y$ | Dato correspondiente a la calificación obtenida por la técnica de comparación. |
| $Z$ | Dato correspondiente a la calificación asignada por el profesor de música. |
| $\|\|$ | Valor absoluto. |
| $ms$ | Milisegundos. |
|  | Nota musical redonda. |
| | Nota musical negra. |
| | Nota musical blanca. |
| | Nota musical blanca con punto. |

Capítulo 1

# Introducción

En la actualidad, la investigación científica se abre camino hacia el uso de herramientas computacionales, para automatizar diversas tareas, reduciendo tiempo y costos innecesarios. De esta manera satisfacer necesidades a la sociedad, enriquecer la enseñanza, proveer distracción, intervenir en procesos de aprendizaje en entornos virtuales, etc. Todo esto para brindar servicios en varios ámbitos, siendo uno de ellos en el área de la música. Uno de los aspectos que se desea automatizar, es el evaluar a estudiantes de música mediante la comparación de interpretaciones musicales.

La comparación de interpretaciones musicales es posible realizarse en archivos de música, estos archivos son piezas musicales que pueden ser representadas computacionalmente de diferente manera.

## 1.1. Representación computacional de los datos musicales

Existen varios problemas en el área de identificación musical MIR (del inglés Music Information Retrieval). MIR es el área interdisciplinaria de recuperación de información musical, siendo un pequeño pero creciente campo de la investigación con muchas aplicaciones del mundo real. MIR cuenta con una clasificación para la representación computacional de los datos de una pieza musical, siendo estos datos musicales útiles para un sistema de recuperación musical, sistemas de evaluación de melodías musicales, etc., consistiendo en dos formas distintas, propuesta por Cheng Yang en [Yang02]:

Una manera de representación de los datos es ***Simbólica*** basada en partituras musicales, es decir, las notas de la partitura, son guardadas en una pista, almacenando el valor del tono o pitch[1], la duración (tiempo de inicio y final de la nota), la velocidad, así como información adicional. Estos datos son almacenados como secuencia de símbolos o cadena de caracteres.

Como ejemplo de la representación simbólica se tienen los archivos en formato MIDI (del inglés Musical Instrument Digital Interface) en español es la Interconexión Digital de Instrumentos Musicales, José Valenzuela en [Valenzuela96]. Adicional a esto, se puede decir que, las partituras [2] de una pieza musical también podrían ser consideras como representación simbólica.

Los datos musicales también se pueden representar de manera ***Acústica***, lo que se tiene es una señal de audio, es una serie de tiempo muestreada a una frecuencia determinada, este tipo de señales suelen ser comprimidas para ahorrar espacio en memoria. Ejemplo de esta representación se incluyen los formatos .wav[3], .au[4] y MP3[5]. Estos datos son almacenados o representados como series de tiempo[6]. La Tabla 1.1 contiene la clasificación de las dos maneras de representar computacionalmente piezas musicales.

Los archivos de música simbólica y acústica pueden dividirse de acuerdo al tipo de melodía, es decir, monofónica que en la música es la textura más sencilla consistiendo en una sola línea melódica sin acompañamiento alguno, solo participa una voz o polifónica es un tipo de textura musical en la que suenan simultáneamente múltiples voces melódicas, es decir participan más de una voz.

[1]Pitch: Indica la frecuencia percibida por el oyente.

[2]Partitura Musical: Texto escrito de una obra musical en el que se anotan los sonidos que han de ejecutar los distintos instrumentos o voces y el modo en que han de hacerlo.

[3]WAV: Wave form audio file format, es un formato de audio digital normalmente sin compresión de datos, desarrollado y propiedad de Microsoft y de IBM que se utiliza para almacenar sonidos en el PC, admite archivos mono y estéreo a diversas resoluciones y velocidades de muestreo [Gómez07].

[4]AU: Es un formato de archivo de audio introducido por Sun Microsystems. El formato fue común en sistemas NeXT y en páginas antiguas de internet.

[5]MP3: (MPEG 1 Layer 3) fue creado por el Instituto Fraunhofer y por su extraordinario grado de compresión y alta calidad está prácticamente monopolizando el mundo del audio digital. Es un formato de compresión de audio digital patentado que usa un algoritmo con pérdida para conseguir un menor tamaño de archivo [Cvejic07].

[6]Serie de Tiempo: Consta de una secuencia de valores o eventos obtenidos sobre mediciones repetidas de tiempo.

Tabla 1.1: Clasificación de la Representación Computacional de los Datos Musicales.

| | | | **Pieza Musical Interpretada por el Estudiante a Evaluar** | | | |
|---|---|---|---|---|---|---|
| | | | *Simbólica* | | *Acústica* | |
| | | | Monofónica | Polifónica | Monofónica | Polifónica |
| **Pieza Musical Referencia Base (considerada como correcta)** | *Simbólica* | *Monofónica* | **A** | | **D** | |
| | | *Polifónica* | | | | |
| | *Acústica* | *Monofónica* | **B** | | **C** | |
| | | *Polifónica* | | | | |

Cabe mencionar que, una pieza musical que es interpretada en un instrumento musical (teclado), si se toca una sola nota a la vez, esa melodía es considerada como monofónica, ahora bien, en presencia de acordes la melodía es considerada polifónica. En la música, un acorde es dos o más notas musicales tocadas al mismo tiempo, de acuerdo a Baily y Collyer en [Baily06].

De acuerdo a la Tabla 1.1, en la categoría marcada con *B* se trabajaría con una interpretación ejecutada correctamente llamada referencia base *acústica* y la interpretación del estudiante a ser evaluada es *simbólica*. Un ejemplo sería, el estudiante puede estar tocando una melodía directamente de un teclado musical MIDI, y a la hora de comparación con el sistema de evaluación hacerlo con la pieza musical referencia base que fue capturada por un micrófono quedando en un formato (.wav).

La categoría con *C* indica que tanto en la referencia base como en la interpretación del estudiante se tienen datos *acústicos*. Como ejemplo de este apartado, se puede citar al artículo realizado por Camarena y Morales [Ibarrola15], el cual desarrollan un método para la evaluación automática de los estudiantes de música. Haciendo el alineamiento entre las series de tiempo en audio digital contra audio digital, es decir, el estudiante selecciona la pieza que va a interpretar entre una lista de piezas musicales conocidas por el sistema de evaluación llamadas referencias base. Posteriormente, el estudiante interpreta la pieza de música ejecutándola con un instrumento musical frente de un micrófono. El micrófono esta conectado a un equipo que ejecuta el programa de evaluación, por lo tanto, el micrófono captura la interpretación del estudiante para que sea guardada en formato (.wav), finalmente se comparan los dos audios digitales y se brinda una calificación a la interpretación del alumno.

En la categoría nombrada con *D* se tiene como referencia base datos *simbólicos* y en la interpretación del estudiante datos *acústicos*. Ejemplo: un estudiante puede estar tocando una pieza musical y ser capturada por el micrófono, al momento de comparar con el sistema de evaluación hacerlo directamente con las partituras de la melodía referencia base. Este sería un problema bastante grande para resolver, ya que en base a las partituras musicales se quiere saber que tan bien interpretó la pieza musical el estudiante.

Ahora bien, en la categoría mencionada como *A*, la presente investigación se centra

en este problema, consistiendo en el desarrollo de un sistema de evaluación automática de interpretaciones musicales basada en información MIDI, es decir datos simbólicos, el cual permite la evaluación automática del estudiante de música. Dicha tarea se lleva a cabo mediante la comparación de una pieza musical llamada *referencia base* contra la *interpretacion musical del estudiante*. La investigación pretende explotar el gusto por la música al usuario final, sean estos, niños, jóvenes o adultos, con el fin de poder automatizar tareas como, comparación y evaluación de interpretaciones musicales realizadas por el estudiante. Con el propósito de no verse en la necesidad de contar con un maestro presencial que realice personalmente la labor como tal. El usuario final podrá interpretar melodías musicales y recibir una evaluación de acuerdo a la interpretación realizada.

Aprovechando que los teclados MIDI brindan una buena tecnología hoy en día, cabe decir, que tanto la referencia base e interpretación del estudiante, ambas interpretaciones son tocadas desde un teclado musical, por lo tanto solo se propone evaluar melodías interpretadas del teclado, de esta manera trabajando con ***datos simbólicos***. Es decir, el uso de los archivos con formato MIDI, posteriormente la información contenida en ellos es procesada por el sistema de evaluación, para poder arrojar una calificación pondera de 0 a 10, de acuerdo al desempeño de la interpretación musical del estudiante.

Cabe mencionar con relación a la investigación, las melodías monofónicas como recomendación son interpretadas por estudiantes principiantes, en donde una melodía simple puede ser interpretada con tan solo presionar las teclas musicales del teclado MIDI con un solo dedo. Las melodías polifónicas son interpretadas por estudiantes de nivel avanzado en donde la interpretación musical requiere presionar varias teclas a la vez.

## 1.2. Planteamiento del problema

Debido al gran auge que ha tomado la música en la sociedad, surge la necesidad de desarrollar una herramienta apta para llegar a la automatización de tareas como la evaluación de estudiantes de música, de esta manera beneficiando a una gran cantidad de escuelas dedicadas a la enseñanza de la música. Permitiendo reducir costos y tiempo que dedica un profesor para evaluar a un estudiante. Pero aún mejor obteniendo resultados

confiables e imparciales hacia la calificación de los estudiantes, así evitar contar de forma presencial con una gran cantidad de profesores de música.

De acuerdo a lo anterior, se quisiera tener masificación de la enseñanza de la música pero eso implicaría contar con un maestro presencial dedicado a atender a un estudiante a la vez, de esta manera el profesor lograría enseña al estudiante a tocar un instrumento como es el teclado musical. Sin embargo eso incluye evaluar al estudiante para poder verificar si ya comprendió como interpretar una melodía en dicho instrumento musical, identificar si el estudiante ya lo sabe hacer correctamente, entonces pueda pasar a la siguiente lección, etc.

## 1.3. Propuesta de solución

Consiste en desarrollar un sistema de evaluación automática de interpretaciones musicales basada en información MIDI, es decir, mediante datos simbólicos.

El sistema compara dos interpretaciones musicales, siendo ambas la misma melodía pero una interpretada correctamente nombrada como *referencia base*, contra la *interpretación realizada por el estudiante de música.*

El sistema desarrollado proporciona una evaluación a la interpretación realizada por el estudiante, dando una calificación ponderada de 0 a 10. Cabe mencionar que, solo se propone evaluar melodías interpretadas desde un teclado musical MIDI. Por lo tanto, la referencia base e interpretación del estudiante son procesadas en una codificación en archivos con formato MIDI, trabajando solo con datos simbólicos. De esta manera, los archivos teniendo las mismas características estructurales para poder realizar el procesamiento de los datos simbólicos extraídos de una forma adecuada, evitando involucrar información innecesaria.

El enfoque en la evaluación de los estudiantes en este trabajo consiste en convertir el archivo MIDI producido por la interpretación del estudiante en cadenas de caracteres largas donde la secuencia de notas, su duración e incluso la fuerza con la que se presiona la nota han sido concatenados. Tales cadenas largas son comparadas con otra cadena larga que corresponde a la *referencia base* de la misma pieza musical. Entre más se parezca la cadena generada por el estudiante a la cadena de referencia base, la calificación será más

alta para el estudiante.

Para la realización de la evaluación automática de interpretaciones musicales se hace uso de la teoría de procesamiento de texto, es decir son técnicas que permiten comparar cadenas de texto. Por lo que en esta investigación se experimentó implementando varias *técnicas de comparación*, como son: la distancia de Levenshtein de acuerdo a Navarro y Raffinot en [Navarro02], distancia InDel por Lemström *et al.* [Lemström04], y algunas métricas diseñadas específicamente para trabajar con datos simbólico como TWLCS (del inglés,Time Warped Longest Common Subsequence) por Guo y Siegelmann en [Guo04]. También fue necesario considera, que una característica particular en el análisis de la música es, la transposición de notas musicales: La misma melodía es percibida bien auditivamente incluso si la secuencia del tono es desplazado de una tecla a otra. Para ello, se implementa la distancia invariante a transposición InDel, calculando la misma distancia pero de dos diferentes maneras, una utilizando la metodología del algoritmo *Branch and Bound* por Lemström *et al.* [Lemström04] y la segunda como forma de validar resultados es calcular la distancia de *manera directa*. Cada técnica antes mencionada proporciona un valor de distancias entre las cadenas de referencia base e interpretación del estudiante, ese valor de distancia determina que tanto se parecen las dos cadenas a comparar, además de ser utilizado para la etapa de evaluación la cual es ponderada a una calificación. Al final se realiza una comparativa entre las diferentes métricas implementadas para identificar que técnica brinda resultados favorables o apegados a las calificaciones asignadas por el profesor de música y poder concluir que métrica funge como mejor calificador.

## 1.4. Objetivos de la tesis

### 1.4.1. Objetivo general

El objetivo de la tesis radica en comparar una interpretación musical con otra ejecución considerada como correcta aplicando técnicas de comparación, para desarrollar un sistema de evaluación automática de interpretaciones musicales basada en información MIDI. De esta manera, evaluar con exactitud la interpretación musical ejecutada por el estudiante de música en comparación con la melodía tomada como referencia base.

### 1.4.2. Objetivos particulares

- Identificar la estructura de un archivo de audio en formato MIDI.
- Obtener las interpretaciones musicales realizadas por el estudiante y profesor de música en archivos MIDI. Los archivos MIDI son interpretaciones musicales de melodías monofónicas y polifónicas solo interpretadas en un instrumento como el teclado musical.
- Implementar un proceso de extracción de datos simbólicos para el desarrollo del sistema. Tomando la información de los archivos MIDI como son: la secuencia de notas, duraciones e incluso las fuerzas con la que se presiona las notas.
- De acuerdo a la teoría de procesamiento de texto, implementar técnicas de comparación, tales como: distancia de Levenshtein, distancia InDel, distancia Time Warped Longest Common Subsequence y la distancia invariante a transposición InDel.
- Implementar la distancia invariante a transposición InDel de dos formas: de manera directa y utilizando la metodología del algoritmo Branch and Bound, así mismo comparar tiempos de ejecución entre las métricas de comparación antes mencionadas.
- De las pruebas realizadas ponderar los resultados arrojados por cada técnica de comparación a una calificación entre 0 a 10 para cada pieza interpretada.
- Evaluar los resultados del sistema utilizando los diferentes esquemas implementados, mediante el cálculo de suma del valor absoluto de diferencias, para identificar que métrica de comparación obtuvo resultados iguales o aproximados a los asignados por el profesor de música.

## 1.5. Descripción de capítulos

En el capítulo 2 se mencionan trabajos e investigaciones previas al presente proyecto, sirviendo de conocimiento para decidir la manera en que se abordaría el problema que compete a esta investigación. Además de exponer una revisión del estado del arte.

En el capítulo 3 se exponen las diferentes técnicas de comparación utilizadas, como son la distancia de Levenshtein, distancia InDel, la distancia invariante a transposición InDel y distancia Time Warped Longest Common Subsequence.

En el capítulo 4 se describe el sistema desarrollado para la evaluación automática de interpretaciones musicales basada en información MIDI, exponiéndose el proceso llevado a cabo. Describiendo la etapa de extracción de los datos simbólicos contenidos en los archivos MIDI, así como la etapa de comparación y finalmente como se obtiene la evaluación de la interpretación del estudiante.

En el capítulo 5 se muestran las pruebas y resultados obtenidos de las diferentes técnicas de comparación. Así como la comparativa y rendimiento entre los distintos esquemas implementados.

Finalmente, en el capítulo 6 se presentan las conclusiones y se enlistan los trabajos a futuro con relación a la investigación, con el fin de surgir posibles mejoras al sistema de evaluación.

## 1.6. Conclusiones del capítulo

En este capítulo se ha planteado que para el desarrollo del sistema de evaluación automática se requiere de contar con una interpretación de referencia base y la interpretación a ser evaluada. Ambas interpretaciones en una codificación MIDI, por lo que el enfoque de la evaluación de los estudiantes consiste en la extracción de la información, es decir convertir el archivo MIDI producido por el estudiante en cadenas de caracteres largas donde la secuencia de notas, su duración e incluso la fuerza han sido concatenados. Tales cadenas largas son comparadas con otra cadena larga que corresponde a una versión correcta, es decir referencia base de la misma pieza musical, la cual ha sido interpretada por algún músico de formación.

De acuerdo a la revisión y estado del arte se encuentra que existen dos formas de representar computacionalmente los archivos de música, es decir las piezas musicales. Una manera es la representación en datos *Simbólicos*, es decir, pueden ser partituras musicales y archivos en formato MIDI. Los archivos MIDI tienen una bitácora de los eventos ocurridos cuando un músico toca en un instrumento MIDI, es decir, almacena valores de tono,

duración, velocidad, etc. Otra forma son los datos *Acústicos*, se tiene es una señal de audio, es una serie de tiempo muestreada a una frecuencia determinada, entre estos se encuentra los formatos .wav, .au y MP3. Ambas representaciones de datos son divididas de acuerdo al tipo de melodía, monofónica o polifónica. Dicho lo anterior la presente investigación se centra en el procesamiento de datos simbólicos haciendo uso de archivos MIDI.

En la investigación se plantea implementar diferentes métricas de comparación de procesamiento de cadenas, las cuales calculen una medida de distancia de acuerdo al desempeño de la interpretación a ser evaluada con relación a la referencia base. Estas métricas, tienen distintas formar de abordar y resolver el problema. Siendo las siguientes: distancia de Levenshtein, distancia InDel, TWLCS y la distancia invariante a transposición InDel, esta última calculando la misma distancia pero de dos diferentes maneras, una utilizando la metodología del algoritmo *Branch and Bound* y la segunda como forma de validar resultados es de *manera directa*. Finalmente, se contemplan los valores de distancias para ser ponderados a una calificación hacia el estudiante.

Capítulo 2

# Estado del arte

## 2.1. La notación musical y el audio

La notación musical se empezó a usar siglos antes de la edad media para poder guardar y compartir ideas y piezas musicales, de manera que estas podían ser interpretadas más veces.

Antes de la forma de notación musical que se utiliza hoy en día y que la notación en pentagrama fuera creado, se originaron otros tipos de notación como, la notación neumática y la notación dasiana, entre otras. Debido al pentagrama, cualquier persona dedicada a la música, es capaz de interpretar una melodía musical compuesta por otra persona, Francisco Morales en [Morales09].

En la música occidental se dividen las posibles frecuencias en porciones que son llamadas *octavas* y cada octava es dividida en 12 porciones llamadas *notas*, como se muestra en la Figura 2.1 el teclado musical. Los teclados estándar por lo general tienen 11 octavas, numeradas de 0 a 10, aunque de igual manera suelen haber teclados más chicos conteniendo menos octavas.

La octava corresponde a la distancia entre la parte más grave y más aguda de la escala diatónica[1]. De manera formal, la octava tiene una proporción o radio de 2 a 1. Por ejemplo la nota *La* en la octava *3* está una octava arriba de la nota *La* octava 2, puesto que *La3* tiene una frecuencia de 220 Hz., mientras que *La2* tiene una frecuencia de 110 Hz., es

[1]Escala diatónica: conjunto de sonidos ordenados, está formada por 12 notas musicales.

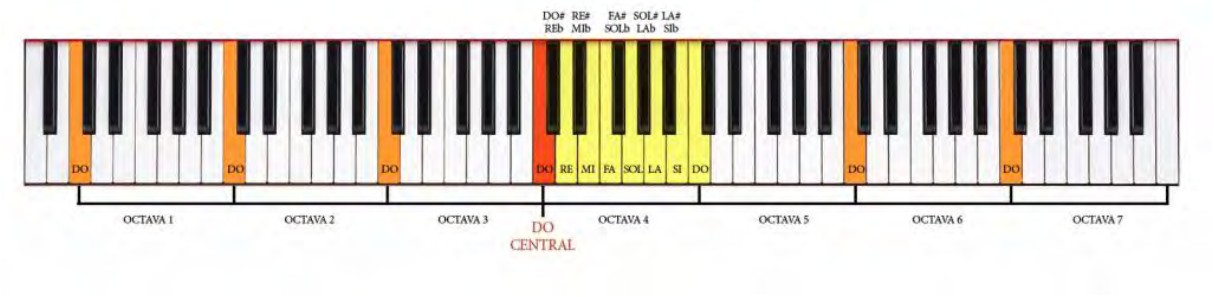


Figura 2.1: División en Octavas en un Teclado Musical

decir, la frecuencia *La3* es el doble de la frecuencia de *La2*. Cada nota de una octava tiene exactamente la mitad de frecuencia que la misma nota en la octava superior.

Las notas de la escala diatónica, como se ve en la Figura 2.2 dictan los siete timbres de una tonalidad a ser usados en una pieza de música. Entonces, las notas musicales (semitonos) son 12, de las cuales 7 son naturales y 5 son alteradas.

Las notas naturales, son denominadas *Do, Re, Mi, Fa, Sol, La, Si* . Estás son nombradas así, ya que son las habituales en la notación musical latina, establecidas por el benedictino Guido de Arezzo (990-1058), a quien se atribuye la invención del actual sistema de solfeo, Francisco Morales en [Morales09].

En la práctica no son suficientes los siete sonidos naturales, pues con frecuencia es necesario modificar el tono de una melodía, para ello, es necesario intercalar nuevas notas o notas alteradas llamadas sostenidos siendo las siguientes: *Do #, Re #, Fa #, Sol #, La #.*

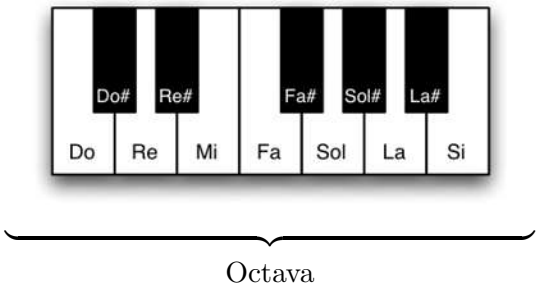


Figura 2.2: Notas de la Escala Diatónica

Si se desean notas más graves o agudas que las 7 anteriores o sus intermedias, se usa el nombre de las mismas notas para los tonos más altos o más graves. Es decir en la Figura 2.1 el sonido del Do de la octava 1 es grave en comparación del Do de la octava 2. Por supuesto que no será la misma nota en la práctica, al escuchar el tono; sin embargo se emplean los mismos nombres mencionando que la nota está en otra octava. De esta manera contar con notas de Do a Si, en la siguiente octava nuevamente de Do a Si, luego la octava posterior de Do a Si, y así sucesivamente todo dentro del rango que pueda ser perceptible por el oído humano. Estas pueden darse en las siguientes formas:

*Octava alta:* Es representada como $8^{va}$, en ocasiones es mostrada en las partituras para marcar que la pieza musical sea tocada una octava más alto de lo que está escrito.

*Octava baja:* Es representada como $8^{vb}$, el cual indica al músico que interprete la pieza musical una octava más baja.

Ahora bien, hoy en día, muchas personas que trabajan en la industria musical o en el ámbito educativo necesitan de una aplicación que realice tareas de manera automática, como es la comparación de interpretaciones musicales, área del que ya se lleva tiempo trabajando. Surge por la necesidad de tratar aspectos como es el medir la similitud entre dos interpretaciones, emparejamiento de notaciones musicales entre melodía, clasificar interpretaciones musicales, etc.

En los años ochenta, y a principios de los noventa, surgen algoritmos con el propósito de realizar la labor de alinear un audio a notación musical. El algoritmos de *string matching*, también conocido como *string searching*, en español es conocido como búsqueda de cadenas o procesamiento de cadenas de texto. Este tipo de algoritmos se encargan de buscar uno o más patrones, o subcadenas, dentro de una cadena larga de acuerdo a Gonzalo Navarro en [Navarro02]. Dichos algoritmos obtienen la extracción de características de las melodías, es decir, valor de pitch, velocidad, etc.

En el año de 1999, Baeza y Navarro en [Baeza99] presentan un algoritmo para el procesamiento de texto. Se basa en paralelización en bits, donde las operaciones de bits son usadas para simular el comportamiento de un autómata finito no determinista NFA (del inglés Non-deterministic Finite Automaton) construido a partir del patrón y utilizando el texto como entrada.

De igual manera a finales de los 90's, se presentó una aproximación para emparejar notación empleando Modelos Ocultos de Markov HMM (del inglés, Hidden Markov Model). Metodología que se comenzó a utilizar en el campo de emparejamiento de notaciones musicales por Cano *et al.* [Cano99], para generar un algoritmo emparejamiento de interpretaciones de partituras utilizando HMM. Incluso Christopher Raphael en [Raphael99] desarrolló un sistema encargado de segmentar automáticamente señales musicales acústicas utilizando este mismo método.

En la investigación del autor Orion y Schwarz [Orio01b], en el 2001 se introdujo una nueva metodología llamada Doblado Dinámico de Tiempo DTW (del inglés, Dynamic Time Warping). La primera técnica en el entorno del alineamiento de audio, para crear un sistema que alineará tanto piezas de audio monofónicas como piezas de audio polifónicas con su notación.

En el 2003, Turetsky y Ellis [Turetsky03], obtienen un sistema de alineación, pero no en tiempo real, mediante el cálculo de una matriz de coste como la correlación entre el espectrograma de la señal original y el de una señal sintética obtenida a partir de un archivo MIDI.

Dos años más tarde, en el 2005 Simón Dixon en [Dixon05] desarrollar un sistema en tiempo real, mediante su sistema de seguimiento de interpretaciones musicales en vivo. Haciendo uso de la técnica DTW tomando las características espectrales de la señal de audio.

Ahora bien, el HMM y el DTW son dos metodologías con una forma de abordar un problema de manera diferente, puesto que el HMM lo hace mediante la utilización de bases probabilísticas y en cambio el DTW es un algoritmo que hace uso de la programación dinámica. Dicho lo anterior, no hay motivo por el cual no pueda haber sistemas que combinen ambos procedimientos, sino todo lo opuesto, las dos metodologías se pueden juntar sin problema alguno. Un ejemplo de ello es el de Durbin *et al.* que en 1998 hace uso de ambas técnicas.

En el 2006, Arshia Cont en [Cont06] desarrolló un sistema de alineamiento de audio a partitura en tiempo real para instrumentos polifónicos utilizando una matriz factorizada no negativa NMF (del inglés Non-negative Matrix Factorization) y un HMM jerárquico.

Por otra parte, en el 2010 en la investigación de Antonio Camarena en [Ibarrola10b] con nombre *Seguimiento en tiempo real de interpretaciones musicales* propone que para tener una buena alineación del *audio objetivo o referencia base* y la interpretación en turno, mediante el seguimiento en tiempo real de interpretaciones musicales, lleva a cabo búsquedas de pequeños segmentos de audio (de un segundo de duración) de la interpretación en el audio objetivo utilizando para ello índices de proximidad. Esta estrategia es más robusta puesto que los errores no se acumulan y además se puede realizar el seguimiento de una pieza musical ya iniciada.

Así mismo en el 2010 Camarena y Chávez en [Ibarrola10a] proponen en su investigación que para el seguimiento de audio en interpretaciones musicales, utilizar la búsqueda de los k-vecinos más cercanos del segmento de audio actual entre todos los segmentos de audio de la referencia base, utiliza algunas heurísticas para decidir la posición del segmento de audio actual.

En el 2011, Manzo y Camarena en [Manzo11] utilizan la Técnica de Alineamiento por Distancia Coseno (TADC). Algunas características importantes de la técnica son que no requiere conocimiento previo de las series de tiempo, ni tampoco fase de entrenamiento. La TADC se basa en la distancia coseno y mínimos cuadrados para realizar el alineamiento. La TADC requiere como parámetro la dimensión de dos vectores. Cuando se consideran dimensiones grandes sobre estos vectores se logra el mejor desempeño proporcionando la medida de similitud más pequeña posible.

Percival *et al.* [Percival07], en su artículo presentó un estudio de trabajos recientes de enseñanza asistida por computadora de instrumentos musicales y afirma que la música es el área que más necesita la ayuda de la enseñanza asistida por computadora pues se requiere de mucha práctica diaria y de motivar a los alumnos a la práctica a través de juegos y multimedia, afirma también que se debe de proporcionar un análisis objetivo del rendimiento de los estudiantes. Muchos proyectos existentes intentan reemplazar a los maestros humanos, proporcionando lecciones durante la práctica diaria.

Ahora un proyecto muy ligado a la presente investigación es el realizado por Camarena y Morales en el artículo [Ibarrola15], presentado el en 2015 en el MICAI (del inglés Mexican International Conference on Artificial Intelligence). En donde se desarrolla un

método para la evaluación automática de los estudiantes de música. La diferencia entre la investigación de Camarena y Morales con relación a la expuesta en el presente documento, radica en que, el alineamiento entre las series de tiempo se realiza de audio digital contra audio digital y no con música simbólica como es el caso de esta investigación.

Es decir, el estudiante interpreta una pieza de música específica, tocando su instrumento musical enfrente de un micrófono conectado a un equipo que ejecuta el programa de evaluación. El estudiante selecciona la pieza que va a interpretar entre una lista de piezas musicales conocidas por el sistema llamadas referencia base. Después, transforman la señal de audio en una secuencia de vectores de características, posteriormente se alinean tanto la interpretación del estudiante como la referencia base para determinar que tan parecidas son. En la investigación utilizan las técnicas de alineamiento de series de tiempo como son, DTW (del inglés Dynamic Time Warping), distancia Levenshtein y la distancia LCS, de esta manera dan una calificación a la interpretación del estudiante. El sistema desarrollado por los autores, consiste en dos módulos, el módulo de extracción de características de la señal de audio y el módulo de evaluación.

## 2.2. Comparación de interpretaciones musicales

La comparación de interpretaciones musicales se puede definir como la tarea de comparar una pieza musical contra otra pieza musical utilizando técnicas de comparación.

Las técnicas de comparación, proporcionan un análisis y evaluación de la interpretación musical. Hoy en día, varias aplicaciones o sistemas utilizan numerosas técnicas que han sido propuestas e implementadas para realizar comparación de audios. Dichas técnicas teniendo cada una distintas formar de abordar y resolver el problema encomendado.

En un sistema de comparación de audios se puede realizar la comparación de dos interpretaciones musicales de dos formas. La primera desarrollando un sistema de comparación en tiempo real ó un sistema no en tiempo real.

### 2.2.1. Tiempo real

Hay algoritmos que trabajan mientras se esta interpretando o reproduciendo la pieza musical en el instante, procesan el audio en tiempo real.

Un sistema en tiempo real obliga que sea computacionalmente rápido, puesto que tiene que procesar la información introducida en un instante preciso y de inmediato retornar un resultado final.

### 2.2.2. No en tiempo real

Son aquellos sistemas que ejecutan la tarea tomándose el tiempo necesario para procesar la información y poder retornar un resultado final. Es decir, para comparar 2 interpretaciones musicales, una vez habiendo finalizado la interpretación musical a comparar se prosigue a la ejecución del algoritmo desarrollado y posteriormente dar un resultado.

Por lo tanto, en ciertos sistemas no es fundamental el tiempo de proceso al igual que, el contar con información futura para poseer más conocimiento de la interpretación. Para esta investigación, se lleva la implementación a un sistema en tiempo no real.

## 2.3. Técnicas de comparación

Actualmente existen una variedad de técnicas de comparación que calculan una medida de distancia, proveen un valor de similitud, etc., entre dos secuencias de símbolos o cadena de caracteres cuando ambas son comparadas.

Hay diferentes técnicas, como lo son las que utilizan bases probabilísticas como los Modelos Ocultos de Markov (HMM del inglés Hidden Markov Model) por Orio y Déchelle en [Orio01a], Cano *et al.* [Cano99], o puede ser basadas en programación dinámica como Doblado Dinámico de Tiempo DTW por Juan García en [García07]. También las utilizadas en procesamiento de cadenas de caracteres o series de tiempo por Gonzalo Navarro [Navarro01] como la distancia de Levenshtein, distancia Hamming, distancia de la subsecuencia común más larga LCS, entre otras.

A continuación se explica la definición de lo que es una medida de distancia y sus propiedades.

### 2.3.1. Medida de distancia

Desde un punto de vista formal, una distancia entre dos cadenas $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{S}$ es una función $d(\boldsymbol{a}, \boldsymbol{b}) : \boldsymbol{X} \times \boldsymbol{X} \rightarrow \mathbb{R}^+ \cup \{\boldsymbol{0}\}$ y que cumple las siguientes propiedades, de acuerdo a Escobedo y Mendoza en [Portillo08].

#### No negatividad

La distancia entre dos cadenas es igual o mayor de 0 (2.1).

$$d(\boldsymbol{a}, \boldsymbol{b}) \geq 0, \forall \boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{X} \tag{2.1}$$

#### Reflexividad

La distancia entre dos cadenas es nula si y solo si las dos cadenas son coincidentes. O bien, la distancia de una cadena comparado consigo mismo es 0 (2.2).

$$d(\boldsymbol{a}, \boldsymbol{b}) = 0 \text{ si y solo si } \boldsymbol{a} = \boldsymbol{b} \tag{2.2}$$

#### Simetría

La distancia de la cadena $\boldsymbol{a}$ a la cadena $\boldsymbol{b}$ es igual a la distancia de la cadena $\boldsymbol{b}$ a la cadena $\boldsymbol{a}$ (2.3).

$$d(\boldsymbol{a}, \boldsymbol{b}) = d(\boldsymbol{b}, \boldsymbol{a}), \forall \boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{X} \tag{2.3}$$

#### Desigualdad triangular

Las distancias entre tres cadenas cumplen con (2.4). Es decir, la distancia entre $\boldsymbol{a}$ y $\boldsymbol{c}$ mas la distancia entre $\boldsymbol{b}$ y $\boldsymbol{c}$, deben ser mayor o igual a la distancia entre $\boldsymbol{a}$ y $\boldsymbol{b}$.

$$d(\boldsymbol{a}, \boldsymbol{b}) \leq d(\boldsymbol{a}, \boldsymbol{c}) + d(\boldsymbol{b}, \boldsymbol{c}), \forall \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \boldsymbol{X} \tag{2.4}$$

Cabe señalar que, si estas tres características las cumple una función de comparación, es correcto llamarle a esa medida *distancia*. Existen técnicas de comparación las cuales

son consideradas como medida de *similitud*, tal es el caso de la métrica TWLCS. TWLCS es una técnica de similitud definida de esta manera por el autor, pero en la presente investigación se hacen las modificaciones necesarias para obtener una medida de distancia TWLCS. A continuación se menciona la diferencia entre distancia y similitud.

#### Distancia

Una medida de distancia calcula un valor mayor mientras menos se parecen dos cadenas de caracteres. Es decir, incrementa el valor de distancia cuando hay más diferencia entre dos cadenas de caracteres.

Entonces, para esta investigación se puede decir que, a mayor distancia se obtendrá menor calificación, debido a que son menos parecidas las dos interpretaciones musicales a comparar.

#### Similitud

Similitud es opuesto a distancia, ya que es una cantidad que es mayor mientras más se parecen dos cadenas de caracteres o series de tiempo. Por lo tanto, es mayor el valor de similitud conforme más parecidos o más ligados están dos cadenas o series.

Para fin del proyecto, a mayor similitud se tendrá mayor calificación, ya que más se parecen las dos piezas musicales comparadas.

Cabe mencionar que, distancia y similitud son métricas que permiten llegar a cuantificar las operaciones de edición necesarias para convertir una cadena de caracteres o serie de tiempo en otra cadena de caracteres o serie de tiempo.

## 2.4. Archivos MIDI

Como pequeña introducción, el protocolo estándar de comunicaciones MIDI se concibió en 1983 para comunicar sintetizadores musicales. MIDI en español se conoce como la Interconexión Digital de Instrumentos Musicales, José Valenzuela [Valenzuela96].

El formato MIDI ha tomado importancia en la industria musical permitiendo que se use con más frecuencia en la producción musical, post-producción y multimedia. Además

del uso en aplicaciones o herramientas de investigación que permiten realizar tareas con la información grabada en los archivos MIDI's.

MIDI no se considera de audio digital, ya que no almacena muestras de un determinado sonido sino una descripción musical. Es decir, los archivos MIDI tienen una bitácora de los eventos ocurridos en una pieza musical. Cuando un músico toca en un instrumento MIDI (ejemplo: teclado), se almacena en el archivo MIDI valores de tonos, duraciones, velocidades, etc. Sólo puede representar sonidos musicales, es decir contienen una secuencia de música grabada como un conjunto de números que indican como debe ser reproducida esta música, son datos simbólicos.

La principal ventaja de este tipo de archivo es el tamaño tan reducido de espacio en disco que ocupa, de esta manera agiliza y procesa rápidamente la información contenida en él para la aplicación desarrollada.

A continuación se expone la estructura de un archivo MIDI, ya que es necesario analizar y estudiar cómo está conformado este tipo de formato para poder llevar a cabo la investigación.

### 2.4.1. Estructura del archivo MIDI

Al investigar sobre el formato, se encontró que existe gran cantidad de MIDI's que contienen varias pistas, los cuales involucran diferentes instrumentos musicales a la vez (como bajo eléctrico, piano, guitarra, saxofón, etc.). Pero para el fin de la investigación, esto no es funcional, puesto que solo se propone evaluar melodías interpretadas del teclado musical.

El desarrollo de la investigación se da inicio con el análisis y extracción de la información que contiene el archivo MIDI y en base a la información extraída realizar la comparación de las interpretaciones a evaluar con las técnicas de comparación implementadas.

MIDI efectúa la transmisión de los datos a partir de mensajes. En términos simples, un archivo MIDI también se puede entender como una partitura, puesto que en él se guarda la información de, cuándo tiene que sonar una nota, así como la característica de dicha nota. Los parámetros más importantes que definen una nota según el estándar son los siguientes:

1. *Valor de la nota:* Cada nota tiene asignada un valor de frecuencia fundamental, que las distingue entre ellas, nombrado como *Pitch*.

2. *Velocidad:* Define la fuerza con la que se tocó la nota MIDI. Al escuchar una pieza musical, este parámetro de velocidad se ve reflejado ante el oído humando como volumen.

3. *Duración:* Tiempo que permanece sonando la nota.

La estructura de un archivo MIDI se compone de una sucesión de segmentos, siendo enlistados de la siguiente manera:

1. Contiene un segmento de cabecera.

2. Seguido de un número variable de pistas. Una pista se asemeja conceptualmente a un pentagrama, puede describir el sonido de un determinado instrumento.

3. A su vez, cada pista contiene un número de eventos, los cuales constituyen una secuencia de mensajes.

**Mensajes MIDI**

MIDI es un protocolo el cual rige un sistema de comunicaciones serie cuya unidad es el mensaje, este protocolo contiene una variedad de mensajes como son: *Mensaje de Nota* (Note off y Note on), *Mensajes de Control* (Polyphonic aftertouch, Control change, Program change, Channel aftertouch, Pitch Wheel), *Mensajes de sistema exclusivo* (System exclusive, End of SysEx), *Mensajes comunes* (Quarter trame, Quarter trame, Song pointer, Song select, Tune request). Para el objetivo de la investigación los *Mensajes de Nota* son los necesarios, estos contienen la información requerida e importante para la comparación de la ejecución de dos interpretaciones musicales. Ahora bien, cada mensaje de nota está producido por una acción realizada en el teclado musical, al enviar un mensaje de nota MIDI, sucede lo siguiente de acuerdo a Reverte y Hidalgo [Reverte06]:

- Se presiona una nota musical, genera mensaje MIDI de *Note on*.
  - Mensaje que indica que se ha tocado una nota.

  - Información del tono (pitch) de la nota.
  - Información de su velocidad

- Se suelta la nota, se genera mensaje *Note off*:
  - Mensaje que indica que una nota ha finalizado.
  - Información del tono (pitch) que finaliza.
  - Información de velocidad, que fue liberada la tecla.

Para esta investigación nos centramos solo en los mensajes de nota, son los que sirven para identificar u comparar la forma en que fue interpretada una melodía musical. Estos mensajes, permiten codifican la tecla musical para representar las notas del teclado. Los mensajes contenidos en el MIDI almacenan información como son los parámetros: nota, velocidad, número de tick, este último permite obtener a duración de una nota musical, en música se toma como un marcador o un tictac, el cual se puede denominar como pulsos por segundo. Para más información del formato MIDI, ir al Apéndice A: Formato MIDI.

## 2.5. Conclusiones del capítulo

En este capítulo se hace una revisión del estado del arte, además de identificar trabajos relacionados a la comparación y evaluación de interpretaciones musicales. La comparación de interpretaciones musicales es una área que surge por la necesidad de tratar aspectos de clasificar interpretaciones musicales, medir similitud entre dos interpretaciones, alineamiento de notaciones musicales entre melodía, etc.

La comparación de interpretaciones musicales se puede definir como la tarea de comparar una pieza musical contra otra pieza musical, siendo representados en cadenas de caracteres utilizando técnicas de comparación. Cabe mencionar que, el desarrollar un sistema de comparación y evaluación de interpretaciones musicales se puede abordar de dos maneras, sistema de evaluación en tiempo real y sistema de evaluación no en tiempo real.

De acuerdo a la revisión del estado del arte, se considera que hay varias técnicas de comparación a utilizar: como lo son las que utilizan bases probabilísticas como los Modelos

Ocultos de Markov HMM o basadas en programación dinámica como DTW, las utilizadas en procesamiento de texto como la distancia de Levenshtein, distancia LCS, entre otras.

Por otra parte, se analiza la estructura de un archivo MIDI, para identificar la información que contiene y la forma en que se puede extraer. Concluyendo que los datos necesarios para realizar una comparación de interpretaciones requiere de obtener valores de tonos, duraciones y velocidades por cada nota musical. MIDI efectúa la transmisión de los datos a partir de mensajes y no de señales digitales para la generación de sonidos.

Capítulo 3

# Técnicas de comparación

En este capítulo se describen las técnicas de comparación a utilizar, además de mencionan ejemplos para una mejor comprensión de la forma en que trabaja cada técnica.

Los métodos de comparación empleados están basados en programación dinámica. Juan García en [García07] propone la técnica denominada programación dinámica que consiste en reducir el espacio o el tiempo de ejecución de cierto algoritmo mediante la utilización de sub-problemas, es decir, la descomposición de un problema general en problemas más pequeños que al ser resueltos de forma independiente, organizan la ejecución, la programación y reducen costos innecesarios de ejecución.

Para dar inicio, se describen los conceptos básicos, mostrando una nomenclatura que permita su correcta interpretación, siendo los siguientes:

1. *Alfabeto*: Es un conjunto finito de símbolos o caracteres, donde cada símbolo representa una unidad. Estos símbolos, también llamados caracteres, no se repiten en el mismo alfabeto. Estos caracteres pueden ser letras, números enteros, números decimales. Al referirse a un alfabeto se utiliza la letra griega mayúscula $\Sigma$. En el caso de esta investigación nuestro alfabeto se define como $\Sigma = \{\mathbb{Z} | 0 \leq z \leq 127\}$.

2. *Cadena de caracteres*: Es una secuencia finita de símbolos que pertenecen a determinado alfabeto, se representa por sus símbolos seguidos. Es posible identificar la cadena de caracteres con un nombre, seguido de un signo ":" o "=". Ejemplo: Cadena 1: 22044310 o Cadena 2= AABBCCDDEE.

3. *Patrón y errores*: Patrón es la cadena de caracteres que es buscada en un texto, es un patrón porque también se busca sus similitudes, es decir las cadenas o sub-cadenas contenidas en el texto, que no siendo idénticas al patrón se puede convertir en él, haciendo una serie de operaciones; si se le asigna determinado peso a cada operación realizada, la suma de estos pesos se definen como errores.

## 3.1. Técnicas de procesamiento de cadenas o secuencias biológicas (ADN)

El procesamiento de texto es un problema común para varias áreas de las ciencias en computacionales. El procesamiento de cadenas también se usa en bioinformática para encontrar subcadenas de genes que son comunes en varios organismos, es decir localizar si una secuencia de ADN es similar a otra secuencia de ADN, Navarro y Raffinot en [Navarro02].

Es el procesamiento de texto o como lo mencionan los autores Navarro y Raffinot en el libro [Navarro02], la búsqueda aproximada de texto, más conocido por su nombre en inglés *Approximate String Matching*, además llamado *String Matching Allowing Errors*, es el problema de encontrar un patrón $P$ en un texto $T$ cuando un número limitado de diferencias $K$, es permitido entre el patrón y sus ocurrencias en el texto. En términos más sencillos, se trata del valor de distancia que hay entre dos cadenas de caracteres, es decir valor que se obtiene dependiendo de qué tan parecidas son ambas cadenas.

Aludido a lo anterior, para abordar el problema de procesamiento de texto, se emplean las técnicas de comparación:

1.- Distancia de Levenshtein.

2.- Distancia InDel.

3.- Subsecuencia Común Más Larga LCS.

4.- Time Warped Longest Common Subsequence TWLCS.

### 3.1.1. Distancia de Levenshtein

La distancia de Levenshtein de acuerdo a Bar-Yossef en [Bar-Yossef04], recibe ese nombre en honor al científico ruso Vladimir Iosifovich Levenshtein[1], quien ideó el algoritmo en 1965. Se usa ampliamente en teoría de la información y ciencias de la computación, específicamente en biología computacional y genómica, procesamiento de texto o del lenguaje natural y búsqueda en la Web. Es útil en programas que determinan que tan parecidos son dos cadenas de caracteres, como es el caso de los correctores de ortografía.

La distancia de Levenshtein, Navarro y Raffinot en [Navarro02] llamada también distancia de edición calcula la medida de distancia entre dos cadenas de caracteres. Esta distancia se basa en obtener el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra, es decir toma un peso (costo) mínimo de 1 en las operaciones de edición. Se entiende por operación, bien una *inserción*, *eliminación* o la *sustitución* de un carácter por otro.

Como un ejemplo aludido a esta distancia se tiene lo siguiente: se desea obtener la distancia que hay entre las cadenas de caracteres *casa* y *calle*, de acuerdo a la distancia de Levenshtein el resultado es de **3** porque se necesitan al menos tres ediciones elementales para cambiar una cadena en la otra.

1. **casa** $\rightarrow$ **cala** (sustitución de $s$ por $l$)
2. **cala** $\rightarrow$ **calla** (inserción de $l$ entre $l$ y $a$)
3. **calla** $\rightarrow$ **calle** (sustitución de $a$ por $e$)

#### Cálculo de la distancia de Levenshtein en base a una recurrencia

Para obtener la distancia de Levenshtein se tiene la recurrencia (3.1) de esta manera se puede saber cuántas operaciones se requieren para convertir una cadena de caracteres en otra.

[1]Vladimir I. Levenshtein: Científico ruso nacido en 1935 reconocido por sus trabajos en teoría de la información y por el algoritmo para el cálculo de la distancia que lleva su nombre.

$$D_{i,0} = i \quad , \forall\ 0 \leq i \leq N$$
$$D_{0,j} = j \quad , \forall\ 0 \leq j \leq M$$

$$D_{i,j} = \begin{cases} D_{i-1,j-1} & \text{Si } A_i = B_j, \\ 1 + min(D_{i-1,j-1}, D_{i-1,j}, D_{i,j-1}) & \text{Si } A_i \neq B_j \end{cases} \tag{3.1}$$

donde

$A$ Cadena de caracteres 1 de tamaño N.
$A =$"$A_1, A_2, A_3, ..., A_N$"

$B$ Cadena de caracteres 2 de tamaño M.
$B =$"$B_1, B_2, B_3, ..., B_M$"

$(i-1, j-1)$ $(i-1, j)$ $(i, j-1)$ Restricciones locales, que marcan la trayectoria óptima o el camino de alineamiento.

$D$ Contiene las distancias para cada prefijo de la cadena A y cada prefijo de una cadena B.

Para comparar las dos cadenas se va mapeando cada índice $i$ de la cadena $A$ a un índice $j$ de la cadena $B$. En un rango de $0 \leq i \leq N$ y $0 \leq j \leq M$. Se comparan los elementos $i$ de $A$ contra los $j$ de $B$, si son iguales no se tiene ningún costo, es decir el costo es de 0. Asigna el valor de la restricción que hay en $D$ en el punto $(i-1, j-1)$ al punto $(i, j)$.

Pero si los elementos son diferentes se toma el valor mínimo de las siguientes 3 posiciones $(i-1, j-1), (i-1, j), (i, j-1)$ en la matriz $D$ para la trayectoria óptima, al referir el valor mínimo es para tomar la distancia mínima. Sumándole un costo o peso de 1. Al terminar la recurrencia, se obtiene la distancia entre las dos cadenas a comparar.

Tomando el ejemplo anterior, la distancia de Levenshtein entre la cadena *casa* y *calle* es de 3 como se ve en la posición (4,5) de la matriz en la Figura 3.1 corresponde a tres operaciones (sustitución de $s$ por $l$, insertar $l$ entre $l$ y $a$ finalmente sustituir $a$ por $e$).

| | | **c** | **a** | **l** | **l** | **e** |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| **c** | 1 | 0 | 1 | 2 | 3 | 4 |
| **a** | 2 | 1 | 0 | 1 | 2 | 3 |
| **s** | 3 | 2 | 1 | 1 | 2 | 3 |
| **a** | 4 | 3 | 2 | 2 | 2 | **3** |

Figura 3.1: Ejemplo de uso de recurrencia de la distancia de Levenshtein

### 3.1.2. Distancia InDel

La distancia InDel como la menciona Lemström *et al.* en [Lemström04], también es conocida como *distancia LCS*. Además es conocida por su relación a la distancia de Levenshtein, debido a que si se cambian los pesos en la distancia Levenshtein se tiene la distancia Indel. Solo es cuestión de usar un peso de dos en la sustitución en Levenshtein, esto es porque en distancia Indel se ve la sustitución como dos operaciones, es decir, solo permite operaciones de *inserción* y *eliminación* dando un costo de 1 a cada una.

Tomando el ejemplo de querer convertir la cadena *casa* en *calle* mediante esta distancia, sería de la siguiente manera:

1. **casa** $\rightarrow$ **cala** (eliminación de $s$ y posteriormente inserción de $l$)
2. **cala** $\rightarrow$ **call** (eliminación de $a$ y posteriormente inserción de $l$)
3. **call** $\rightarrow$ **calle** (inserción de $e$)

El resultado es de **5** operaciones a realizar, para cambiar una cadena en la otra. Teniendo la distancia de entre ellas de 5, mientras que con la distancia de Levenshtein es de 3.

#### Cálculo de la distancia InDel en base a una recurrencia

La distancia Indel fue diseñada para calcular un bajo valor donde la misma secuencia de símbolos están presentes en dos cadenas de caracteres, es común en ocasiones cuando se tiene señales que fueron habladas o cantadas, es decir en representaciones acústicas.

Ejemplo: la distancia de Levenshtein no puede decir que la cadena *computadora* y *campamento* son muy diferentes en comparación con las cadenas *computadora* y *cooompuuutaaadooraa*. Por lo contrario la distancia Indel puede reportar que la primer pareja de cadenas son más diferentes que la segunda pareja de cadenas. Véase mas adelante en la Tabla 3.1 para la comparación del cálculo ente las dos parejas de cadenas antes mencionadas.

Para el cálculo de la distancia Indel la recurrencia es obtenida de Lemström *et al.* en [Lemström04], de la siguiente (3.2):

$$D_{i,0} = i \quad , \forall\ 0 \leq i \leq N$$
$$D_{0,j} = j \quad , \forall\ 0 \leq j \leq M$$

$$D_{i,j} = \begin{cases} D_{i-1,j-1} & \text{Si } A_i = B_j, \\ 1 + min(D_{i-1,j}, D_{i,j-1}) & \text{Si } A_i \neq B_j \end{cases} \tag{3.2}$$

donde

- $A$ Cadena de caracteres 1 de tamaño N.
- $B$ Cadena de caracteres 2 de tamaño M.
- $(i-1, j-1)$, $(i-1, j)$, $(i, j-1)$ Restricciones locales, que marcan la trayectoria óptima o el camino de alineamiento.
- $D$ Contiene las distancias para cada prefijo de la cadena A y cada prefijo de una cadena B.

Para comparar las dos cadenas se va mapeando cada índice $i$ de la cadena $A$ a un índice $j$ de la cadena $B$. En un rango de $0 \leq i \leq N$ y $0 \leq j \leq M$.

Se comparan los elementos $i$ de $A$ contra los $j$ de $B$, si son iguales no se tiene ningún costo, es decir el costo es de 0. Asigna el valor de la restricción que hay en $D$ en el punto $(i-1, j-1)$ al punto $(i, j)$.

Pero si los elementos son diferentes se toma el valor mínimo de los siguientes 2 puntos $(i-1, j), (i, j-1)$ en $D$, para la trayectoria óptima, al referir el valor mínimo es para tomar la distancia mínima. Sumándole un costo o peso de 1. Al terminar la recurrencia se tiene la distancia entre las dos cadenas.

Ejemplo del uso de la recurrencia InDel, se quiere calcular la distancia entre las cadenas *casa* y *calle*, como se ve en la posición (4,5) de la matriz en la Figura 3.2 la distancia entre ambas es de 5. Es decir se requieren 5 operaciones necesarias para convertir la cadena casa a la cadena calle, siendo las siguientes: eliminar $s$, insertar $l$ luego eliminación de $a$ e inserción de $l$ finalmente insertar $e$.

| | | **c** | **a** | **l** | **l** | **e** |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| **c** | 1 | 0 | 1 | 2 | 3 | 4 |
| **a** | 2 | 1 | 0 | 1 | 2 | 3 |
| **s** | 3 | 2 | 1 | 1 | 2 | 3 |
| **a** | 4 | 3 | 2 | 3 | 4 | **5** |

Figura 3.2: Ejemplo de uso de recurrencia de la distancia InDel

### 3.1.3. Subsecuencia Común Más Larga LCS

Para abordar el procesamiento de texto también se utiliza la subsecuencia común más larga. LCS consiste en encontrar la subsecuencia común más larga correspondiente a dos secuencias o cadenas de caracteres. No importando si ambas cadenas son de diferente tamaño.

LCS de acuerdo a Gonzalo Navarro en [Navarro01], el nombre de esta técnica se refiere al hecho de que se mide la ***longitud*** de la *subsecuencia común más larga* de caracteres que pueden estar entre dos cadenas. LCS respeta el orden de las letras, es decir la subsecuencia común más larga no necesita ser consecutiva, pero debe estar en orden.

Para mejor comprensión se expone un ejemplo, se encuentra manualmente la subsecuencia común más larga comparando la cadena *"ncaatournament"* con *"northcarolina"*. Veamos que letras se repiten en el mismo orden tanto en la primer cadena como en la segunda.

Podría decirse que la subsecuencia es *"ncaa"*, de longitud *4* de acuerdo como se

muestra en seguida. Las flechas indican de manera ordenada el carácter común en la cadena 1 con la cadena 2.

**Cadena 1:** *n ca a tournament*
↓ ↓↓ ↓
**Cadena 2:** *north carolina*

Pero está no es la solución, ya que hay otra *subsecuencia común más larga*. Por lo tanto la correcta solución es **ncarna**, de longitud **6**. Como se ve enseguida.

**Cadena 1:** *n caa tour nament*
↓ ↓↓ ↓ ↓↓
**Cadena 2:** *north ca rolina*

Como se menciona antes la subsecuencia no necesita ser consecutiva, pero debe estar en orden.

### Cálculo de la subsecuencia común más larga en base a una recurrencia

Formalmente el problema de LCS esta definido como sigue en (3.3) del artículo Guo y Siegelmann [Guo04]:

$$L_{i,0} = 0 \quad , \forall\ 0 \leq i \leq N$$
$$L_{0,j} = 0 \quad , \forall\ 0 \leq j \leq M$$

$$L_{i,j} = \begin{cases} L_{i-1,j-1} + 1 & \text{Si } A_i = B_j \\ max(L_{i,j-1}, L_{i-1,j}) & \text{Si } A_i \neq B_j \end{cases} \tag{3.3}$$

donde

$A$ Secuencia de caracteres 1 de tamaño N.

$B$ Secuencia de caracteres 2 de tamaño M.

$(i-1, j-1)$, $(i-1, j)$, $(i, j-1)$ Restricciones locales, que marcan la trayectoria óptima o el camino de alineamiento.

$L$ Contiene las longitudes de las secuencias.

Para obtener la longitud de las dos secuencias de caracteres, se va mapeando cada índice $i$ de la cadena $A$ a un índice $j$ de la cadena $B$. En un rango de $0 \leq i \leq N$ y $0 \leq j \leq M$. Se inicializa con ceros la primera fila y columna de la matriz $L$.

Se comparan los elementos $i$ de $A$ contra los $j$ de $B$, si son iguales se suma *1* al valor de la restricción que hay en $L$ en el punto $(i-1, j-1)$ al punto $(i, j)$. Pero si los elementos son diferentes se toma el valor máximo de los siguientes 2 puntos $(i, j-1), (i-1, j)$ en $L$, para la trayectoria óptima, al referir el valor máximo es para tomar la longitud mayor, ya que se quiere obtener la longitud de la subsecuencia común más larga entre las cadenas iniciales a comparar.

Ejemplo del uso de la recurrencia, se tienen la cadena *"abcdef"* y *"axbydezzz"*. Entonces la subsecuencia común más larga entre ambas cadenas es de *"abde"* teniendo una longitud de tamaño **4** como se ve en la posición (6,9) de la matriz en la Figura 3.3.

| | | **a** | **x** | **b** | **y** | **d** | **e** | **z** | **z** | **z** |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **a** | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **b** | 0 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| **c** | 0 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| **d** | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 3 | 3 |
| **e** | 0 | 1 | 1 | 2 | 2 | 3 | 4 | 4 | 4 | 4 |
| **f** | 0 | 1 | 1 | 2 | 2 | 3 | 4 | 4 | 4 | **4** |

Figura 3.3: Ejemplo de uso de recurrencia LCS

Como se dijo anteriormente, la subsecuencia común más larga LCS entre dos cadenas retorna un valor de *longitud* como se muestra en el ejemplo anterior. Esa longitud es posible convertirla a un valor de distancia, ya que distancia no es lo mismo a longitud, pero es posible realizar una conversión con una ecuación (longitud a medida de distancia).

Dicha conversión teniendo un fin, para obtener la distancia invariante a transposición InDel se utiliza la recurrencia LCS, pero es necesario que el valor de longitud resultante por LCS sea convertido a valor de distancia. Puesto que en la etapa de pruebas

de la investigación, se realiza una comparativa de los resultados obtenidos con la distancia de Levenshtein, distancia InDel y distancia TWLCS, por lo que es necesario comparar resultados en terminos de medida de distancia. La conversión utiliza la Ecuación (3.4) descrita en Lemström *et al.* [Lemström04].

Ahora bien, en (3.4), se ve la relación que existe entre la distancia Indel con LCS. Se puede tener el valor de distancia InDel y convertirlo a longitud LCS ó viceversa, contar con la longitud LCS y convertirlo a valor de distancia, siendo este último, el caso que interesa cumplir, para tener los resultados de las técnicas de comparación en valor de distancia.

$$LCS(A,B) = \frac{|A| + |B| - D_{ID}(A,B)}{2} \tag{3.4}$$

donde

| | |
|---|---|
| $A$ | Cadena de caracteres 1. |
| $B$ | Cadena de caracteres 2. |
| $\|A\|$ | Tamaño de la cadena A. |
| $\|B\|$ | Tamaño de la cadena B. |
| $LCS(A,B)$ | Longitud LCS entre la cadena A y B |
| $D_{ID}(A,B)$ | Distancia Indel entre la cadena A y B. |

### 3.1.4. Time Warped Longest Common Subsequence TWLCS

TWLCS de acuerdo a lo citado por Guo y Siegelmann en el artículo [Guo04], es usado para calcular una medida de similitud entre dos cadenas de caracteres. TWLCS se ocupa de los errores que implican las distorsiones rítmicas. Debido a las inexactitudes, variaciones y a los errores introducidos en la fase de interpretación de la música.

Ejemplo: se quiere saber el valor de similitud entre dos cadenas de caracteres, siendo las cadenas *"abcejki"* y *"aabbccdefghii"*. De acuerdo a como se ve enseguida, el valor de similitud es de **9** entre las cadenas.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Cadena 1:** | $a$ | | $b$ | | $c$ | | $e$ | $j$ | $k$ | | $i$ | | |
| | ↓↘ | | ↓↘ | | ↓↘ | | ↓ | | | | ↓↘ | | |
| **Cadena 2:** | $a$ | $a$ | $b$ | $b$ | $c$ | $c$ | $d$ | $e$ | $f$ | $g$ | $h$ | $i$ | $i$ |

Recordando que valor de similitud es opuesto a valor de distancia como se expuso en el Capítulo 2, ya que similitud es un valor alto (mayor) mientras más se parecen dos cadenas de caracteres.

**Cálculo de distancia TWLCS en base a una recurrencia**

La recurrencia (3.5) permite describir la forma en que se define la similitud TWLCS computacionalmente, obtenida del artículo Guo y Siegelmann [Guo04].

$$TWLCS_{i,0} = 0 \quad , \forall\ 0 \leq i \leq N$$
$$TWLCS_{0,j} = 0 \quad , \forall\ 0 \leq j \leq M$$

$$TWLCS_{i,j} = \begin{cases} max(TWLCS_{i,j-1}, TWLCS_{i-1,j}, TWLCS_{i-1,j-1}) + 1 & \text{Si } A_i = B_j \\ max(TWLCS_{i,j-1}, TWLCS_{i-1,j}) & \text{Si } A_i \neq B_j \end{cases} \tag{3.5}$$

Cabe mencionar, que en la investigación fue necesario realizar modificaciones a la recurrencia (3.5) para obtener valores de distancia, puesto que, en la etapa de pruebas se quiere comparar resultados de las técnicas implementadas en medida de distancia. Definiendo ahora la **distancia TWLCS** como se expone en (3.6).

$$DT_{i,0} = i \quad , \forall\ 0 \leq i \leq N$$
$$DT_{0,j} = j \quad , \forall\ 0 \leq j \leq M$$

$$DT_{i,j} = \begin{cases} min(DT_{i,j-1}, DT_{i-1,j}, DT_{i-1,j-1}) & \text{Si } A_i = B_j \\ 1 + min(DT_{i,j-1}, DT_{i-1,j}) & \text{Si } A_i \neq B_j \end{cases} \tag{3.6}$$

donde

| | |
|---|---|
| $A$ | Secuencia de caracteres 1 de tamaño N. |
| $B$ | Secuencia de caracteres 2 de tamaño M. |
| $(i-1, j-1)$ | Restricciones locales, que marcan la trayectoria óptima o el camino de |
| $(i-1, j)$ | alineamiento. |
| $(i, j-1)$ | |
| $DT$ | Contiene las distancias para cada prefijo de la cadena A y cada prefijo de |

Para obtener la medida de distancia de las dos secuencias de caracteres, se va mapeando cada índice $i$ de la cadena $A$ a un índice $j$ de la cadena $B$. En un rango de $0 \leq i \leq N$ y $0 \leq j \leq M$.

Se comparan los elementos $i$ de $A$ contra los $j$ de $B$, si son iguales no se tiene ningún costo, se toma el valor mínimo de la restricción que hay en $DT$ entre los siguientes puntos $(i, j-1)$, $(i-1, j)$, $(i-1, j-1)$ al punto $(i, j)$.

Pero si los elementos son diferentes se toma el valor mínimo de los siguientes 2 puntos $(i, j-1)$ y $(i-1, j)$ en $DT$, para la trayectoria óptima, sumándole un costo de 1. Al terminar la recurrencia se obtiene la distancia entre las dos cadenas.

Ejemplo del uso de la recurrencia (3.6) para el cálculo de distancia TWLCS entre las cadenas *"abcejki"* y *"aabbccdefghii"* como se ve en la posición (7,13) de la matriz en la Figura 3.4 la distancia es de **6**.

| | | **a** | **a** | **b** | **b** | **c** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **i** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| **a** | 1 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| **b** | 2 | 1 | 1 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **c** | 3 | 2 | 2 | 1 | 1 | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **e** | 4 | 3 | 3 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 3 | 4 | 5 | 6 |
| **j** | 5 | 4 | 4 | 3 | 3 | 2 | 2 | 3 | 2 | 3 | 4 | 5 | 6 | 7 |
| **k** | 6 | 5 | 5 | 4 | 4 | 3 | 3 | 4 | 3 | 4 | 5 | 6 | 7 | 8 |
| **i** | 7 | 6 | 6 | 5 | 5 | 4 | 4 | 5 | 4 | 5 | 6 | 7 | 6 | **6** |

Figura 3.4: Ejemplo de uso de recurrencia de la distancia TWLCS

Como resumen de las técnicas antes mencionadas, en la Tabla 3.1 se puede ver la comparación entre las medidas de distancia que proporciona Levenshtein, Indel y TWLCS. Para la distancia TWLCS las cadenas *computadora* y *cooompuuutaaadooraa* son iguales dando una distancia de 0. Si fuera el caso que se estuviera cantando la palabra computadora y se compara con la palabra cooompuuutaaadooraa se esta diciendo la misma palabra solo que alargada. Por lo tanto la distancia TWLCS considera las cadenas iguales. Recordando lo mencionado en el Capítulo 2, la medida de distancia es una cantidad mayor, mientras menos

se parecen dos cadenas de caracteres. Es decir, incrementa el valor de distancia cuanto hay más diferencia entre las dos cadenas. Por el contrario cuando más se parecen dos cadenas el valor de distancia es más pequeño. Es el caso del cálculo que proporciona la distancia TWLCS con el ejemplo expuesto en la Tabla 3.1.

Tabla 3.1: Ejemplo comparación de distancias

| **A** | **B** | **Levenshtein** | **LCS** | **TWLCS** |
|---|---|---|---|---|
| computadora | cooompuuutaaadooraa | 8 | 8 | 0 |
| computadora | campamento | 8 | 11 | 11 |

## 3.2. Técnicas de procesamiento de cadenas invariante a transposición

Una característica particular en el análisis de la música es, la transposición de notas musicales: La misma melodía es percibida incluso si la secuencia del tono "*pitch*" es desplazado de una tecla a otra. Por lo tanto otro punto a resolver en el procesamiento de texto es comparar dos cadenas cuando en una de ellas ha habido un desplazamiento o una transposición de notas musicales. El terminó de *cadenas invariante a transposición* es mencionado de esta manera, de acuerdo a los autores Grabowski y Navarro en [Grabowski04].

La transposición en música quiere decir que, en una pieza musical se traslada de una tonalidad a otra tonalidad. En la práctica al hablar de transponer o transportar una melodía se basa en dirigir todas la notas que la componen hacia arriba o hacia abajo en la escala musical, manteniendo en todas las notas el mismo intervalo entre nota de origen y nota de destino.

En ocasiones suele ocurrir que una melodía puede estar correctamente ejecutada, pero que este interpretada con un desplazamiento de notas (sea de 1 nota, 2 notas, 3 notas o incluso el desplazamiento sea de toda 1 octava) en el teclado musical, ya sea hacia la izquierda o derecha de modo que no sean los tonos exactos que deberían ser de acuerdo a la partitura musical que se esta siguiendo.

Ahora bien, si se considera a detalle la cuestión del cálculo de la distancia entre cadenas invariante a transposición InDel, es el problema de hacer coincidir dos cadenas cuando todos los caracteres de una de ellas puede estar desplazados por una cierta cantidad *"t"* que se requiere determinar.

Al referir a la palabra desplazamiento, significa que las cadenas son secuencias de números y se añade o resta un valor $t$ a cada carácter de una de las cadenas.

Para resolver el problema se implementa la distancia entre cadenas invariante a transposición InDel, calculando la misma distancia pero de dos maneras diferentes. Una forma es utilizando la metodología del algoritmo ***Branch and Bound*** Lemström *et al.* [Lemström04], la segunda y como forma de validar resultados en la etapa de pruebas, calcula la distancia de ***manera directa***. En la presente investigación se le dio este nombre de *manera directa*, ya que en el artículo de Grabowski y Navarro [Grabowski04] los autores realizan el procesamiento de dos cadenas invariante a transposición sumando simplemente un valor de $t$ a cada carácter de una de las cadenas.

### 3.2.1. Distancia entre cadenas invariante a transposición InDel de manera directa

El interés del problema de relacionar las cadenas invariante a transposición ha surgido recientemente en el campo de la recuperación de información musical Grabowski y Navarro [Grabowski04]. En el análisis de la música y la recuperación, a menudo se quiere comparar dos piezas de música para poner a prueba. Una forma lógica de modelar la música es considerar las duraciones y las notas musicales. En ocasiones las duraciones suelen ser ignoradas, puesto que por lo general es posible reconocer la melodía de una secuencia de valores de tonos.

Una característica particular en el análisis de la música es, la transposición de notas musicales: La misma melodía es percibida incluso si la secuencia del tono "*pitch*" es desplazado de una tecla a otra. Esto es equivalente a la adición de una constante para todos los valores de tono de una secuencia. Por lo tanto, el problema de determinar que tan parecidas son dos cadenas bajo invariancia a transposición es de interés en la recuperación de la música. También tiene aplicaciones en la comparación de series de tiempo, la comparación

de imágenes, y otras áreas.

Enseguida se muestra como es un desplazamiento en cuanto a las notas interpretadas, para poder obtener la distancia entre las siguientes dos cadenas.

Como ejemplo, se tiene la cadena 1 llamada *Referencia Base* y la cadena 2 llamada *Interpretación Estudiante*, como se ha mencionado antes, en un teclado cada nota musical también se puede representar o indicar con un valor de tono ó pitch, como se ve en la Figura 3.5

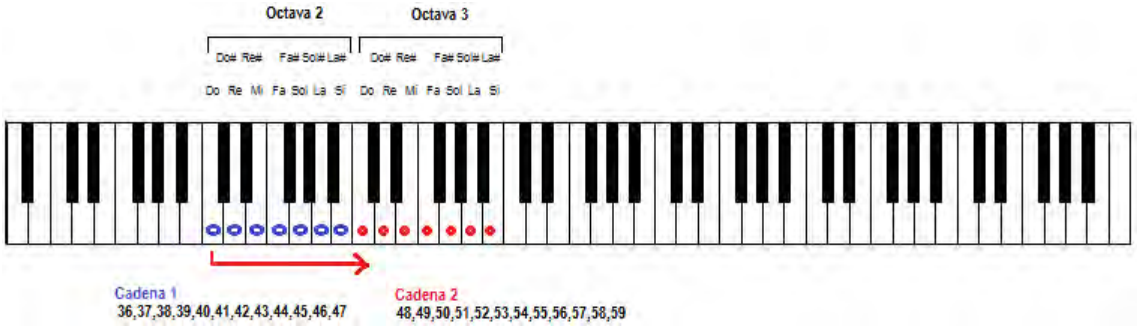


Figura 3.5: Notas desplazadas en una Interpretación

En la Figura 3.5 se ve que la cadena 1 (notas presionadas de color azul) son interpretadas en la octava 2, tocando las 7 notas naturales (teclas blanca) y las 5 notas alteradas (teclas negras). Teniendo los siguientes valores de pitch.

**Cadena 1:** 36,37,38,39,40,41,42,43,44,45,46,47

Mientras que la cadena 2 es la interpretada por el estudiante (notas presionadas de color rojo) pero en la octava 3, lo que indica que hay un desplazamiento de notas, de cantidad **t=12**.

**Cadena 2:** 48,49,50,51,52,53,54,55,56,57,58,59

Es permitido decir que la interpretación es correcta, si se presionan en la misma secuencia las notas musicales, para este ejemplo fue de la siguiente manera: Do, Do#, Re, Re#, Mi, Fa, Fa#, Sol, Sol#, La, Si, pero ejecutada una octava arriba $8^{va}$.

De ahí que, al oír la interpretación se escucha de manera correcta y podría decirse que esta bien ejecutada, simplemente cambia la frecuencia percibida por el oyente, es decir, la pieza musical se puede escuchar un poco más aguda o grave dependiendo del número de

la octava desplazada.

Entonces una vez expuesto lo anterior, la distancia entre cadenas invariante a transposición InDel de manera directa, de acuerdo a Grabowski y Navarro en [Grabowski04], se realiza el cálculo de forma tradicional. Es decir, para encontrar la distancia entre las dos cadenas invariante a transposición suma o resta el valor de desplazamiento $t$ en cada carácter de una de las cadenas.

**Cálculo de distancia invariante a transposición InDel de manera directa en base a una recurrencia**

En el documento de Grabowski y Navarro en [Grabowski04] los autores se centran en el cálculo invariante a transposición InDel, utilizando la longitud LCS, es decir, también nombrada como la Longitud de la Subsecuencia Común Más Larga Invariante a Transposición LCTS (del inglés The Length of the Longest Common Transposition Invariant Subsequence) que se puede obtener después del mejor desplazamiento posible de una secuencia.

El problema se plantea de la siguiente manera:

Sea $\Sigma \subset \mathbb{Z}$ es un alfabeto finito de símbolos. Por simplicidad se considera $\Sigma = 0, ..., \sigma$. Teniendo las cadenas $A$ y $B$: $A =$"$A_1, A_2, A_3, ..., A_N$", $B =$"$B_1, B_2, B_3, ..., B_M$", de tamaño N y M, las dos cadenas de caracteres sobre $\Sigma$, es decir caracteres $A_i$ y $B_j$ que pertenecen a $\Sigma$. Para todo $1 \leq i \leq N$ , $1 \leq j \leq M$.

La longitud de la subsecuencia común mas larga invariante a transposición esta dada por (3.7). Dado que, $t$ pertenece a los números enteros $\mathbb{Z}$, sujeto en un rango de $0 \leq$ t $\leq 127$. Por lo tanto se tiene un problema de optimización con restricciones. La manera de implementarlo es inicializando el valor de *t=0*, posteriormente $t$ va incrementando de 1 en 1. Se va ir obteniendo el valor de longitud LCS(A + t, B) para cada valor de $t$ y del conjunto que resulte de las distancias tomar el mayor.

$$LCTS(A, B) = \max_{t \in Z} LCS(A + t, B) \tag{3.7}$$

donde

$LCTS(A, B)$ La Longitud de la Subsecuencia Común Más Larga Invariante a Transposición.

$A + t$ $A + t = (A_1 + t)(A_2 + t)...(A_N + t)$.
Se va sumando valor de "t" a cada carácter.

$LCS(A + t, B)$ Longitud de la subsecuencia común más larga entre (A + t) y (B), sumando a $A$ un valor de $t$.

$t$ $0 \leq$ t $\leq 127$

Para la obtención del cálculo LCS$(A+t, B)$ para cualquier transposición $t$, se tiene la recurrencia (3.8) obtenida de Lemström *et al.* [Lemström04]:

$$\begin{aligned} LCS^t_{i,0} &= 0; \ , \forall \ 0 \leq i \leq N \\ LCS^t_{0,j} &= 0; \ , \forall \ 0 \leq j \leq M \end{aligned} \tag{3.8}$$

$$LCS^t_{i,j} \begin{cases} 1 + LCS^t_{i-1,j-1} & \text{Si } A_i + t = B_j \\ max(LCS^t_{i,j-1}, LCS^t_{i-1,j}) & \text{Si } A_i \neq B_j \end{cases}$$

donde

$t$ Valor que se suma o resta en cada carácter de una de las cadenas.
Va de $-\sigma \leq t \leq \sigma$

$(i-1, j-1)$ $(i-1, j)$ $(i, j-1)$ Restricciones locales, que marcan la trayectoria óptima o el camino de alineamiento.

$LCS$ Contiene las longitudes de las secuencias.

Cabe mencionar que (3.8), es la misma recurrencia básica de LCS (3.3), la diferencia radica en que a (3.8) se agrega el valor de desplazamiento $t$, en el condicional al momento de comparar carácter por carácter en la cadena $A$.

Al tomar el valor máximo en la recurrencia anterior, se esta obteniendo la longitud de la subsecuencia común más larga. Por lo que, como ya se ha mencionado, LCS da una medida de longitud de cadenas, así que al utilizar la recurrencia (3.8) al final es conveniente hacer la conversión de los resultados a medida de distancia con (3.4) antes mencionada.

### 3.2.2. Distancia entre cadenas invariante a transposición InDel utilizando el algoritmo Branch and Bound

Se calcula la distancia invariante a transposición InDel utilizando la metodología del algortimo Branch and Bound, el cual permite realizar menos cálculos. Este método es aplicable cuando existen muchos caminos hacia un objetivo y se requiere una trayectoria óptima.

Lo que hace este algoritmo es que genera el espacio de soluciones, organizándolo en un árbol binario[2], además poda los subárboles inútiles, es decir árboles que no vayan a generar soluciones óptimas.

En el artículo Lemström *et al.* [Lemström04] se describe el cálculo de la distancia entre cadenas invariante a transposiciones, utilizando el método de Branch and Bound en español Ramifica y Acota, bajo el cálculo de la subsecuencia común más larga LCS. Definiendo el problema de la siguiente manera:

Definimos $LCS^{\beta}(A, B)$, donde $\beta$ es un conjunto de valores (transposiciones) sobre los cuales se evalúa LCS para cada uno de los elementos. Sea $A$ y $B$ cadenas de caracteres de un alfabeto $\Sigma = 0...\sigma$, teniendo $a_i$ y $b_j$, siempre y cuando $b_j - a_i \in \beta$. El objetivo es encontrar el valor máximo $LCS^{t}(A, B)$, donde $t \in \beta$, restringiendo los subconjuntos $\beta$ donde pertenece el óptimo $t$.

Ahora bien, el algoritmo consiste en formar un árbol binario cuyos nodos tienen la forma $[\tau, \tau']$ y representa el rango de transposiciones $\beta = \tau...\tau'$. La raíz es $[-\sigma, \sigma]$. Las hojas tienen la forma $[t, t]$.

Cada nodo interno $[\tau, \tau']$ tiene dos hijos $[\tau, \lfloor\theta\rfloor]$ y $[\lfloor\theta\rfloor + 1, \tau']$. donde $\theta \leftarrow \lfloor\frac{(\tau+\tau')}{2}\rfloor$. Siendo los valores de la cota del valor $LCS^{t}(A, B)$, es decir para cada nodo $[\tau, \tau']$ del árbol, si se cálcula $LCS^{[\tau,\tau']}(A, B)$, el resultado es la cota de $LCS^{t}(A, B)$ para $\tau \leqslant t \leqslant \tau'$.

Se comienza calculando el valor de la raíz $LCS^{[-\sigma,\sigma]}(A, B)$. Ahora, la idea es calcular a sus dos hijos, y continuar con el más prometedor, es decir, el valor mayor calculado de $LCS^{\beta}$ de la cota superior. Para este valor mas alto, se cálcula a sus dos hijos y se obtiene $LCS^{[\tau,\tau']}(A, B)$ de ambos nodos y se ingresan a la cola de prioridad, este proceso se hace

[2]Árbol binario: es una estructura de datos en la cual cada nodo puede tener un hijo izquierdo y un hijo derecho. No pueden tener más de dos hijos.

hasta llegar a la solución óptima.

En cualquier momento, se tiene un conjunto de subárboles a considerar, cada uno con su propio cota superior de las hojas que contiene. El conjunto de subárboles a tener en cuenta se mantiene en una cola de prioridad máxima.

A cada paso del algoritmo, se toma el subárbol más prometedor, se calculan sus dos hijos, y se añade al conjunto de subárboles bajo consideración. Si el subárbol más prometedor resulta ser un nodo hoja $[t, t]$, entonces el valor de la cota superior es de hecho el valor exacto $LCS^t$ calculado. En este punto se puede detener el proceso, debido a que todos los valores de los subárboles restantes son menores o iguales que el valor real $LCS^t$ que se ha obtenido. Así que se esta seguro de haber obtenido el valor más alto.

Características de la metodología Branch and Bound:

*1. Prioridad:* Sirve para guiar la búsqueda y encontrar pronto una solución óptima.

*2. Cota:* Para descubrir pronto que es inútil continuar por el camino actual.

En general conseguir prioridades y cotas perfectas es imposible, pero es importante mencionar que este algoritmo es aplicable para problemas de minimización y maximización.

A continuación se describe un ejemplo de cómo se realiza el proceso del algoritmo para llegar a la solución óptima, mediante la generación del árbol binario como se ve en la Figura 3.6, tiendo las siguientes cadenas:

**Cadena 1:** 11,30,21,12,13,44,33,34,39,21,6,1,36,2,47,45,23,40,24,17

**Cadena 2:** 8,24,46,44,16,39,12,18,10,41,23,42,17,3,44,20,41,35,7,37.

Se comienza por saber que el valor del nodo raíz es calculado con LCS, se especifica un *rango de transposiciones* que para el ejemplo es de **[-50,50]**, ya que este funge como la restricción para el cálculo de la longitud LCS entre las dos cadenas. El rango de transposiciones debe ser igual o mayor al tamaño de las cadenas que se van a comparar,

Ahora sabiendo el rango de transposición (cota superior) y habiendo iniciado el árbol con el nodo raíz (en el ejemplo es de **20**). La idea ahora es calcular a sus dos hijos, y continuar con el más prometedor, es decir, el valor mayor obtenido de LCS.

De acuerdo a lo anterior, se calcula la mitad del rango de transposición o cota superior, que para el ejemplo es igual a **0**, para generar los dos hijos del nodo raíz. Quedando la rama izquierda de **[-50,0]** y derecho de **[1,50]**. Posteriormente se calcula el LCS con las

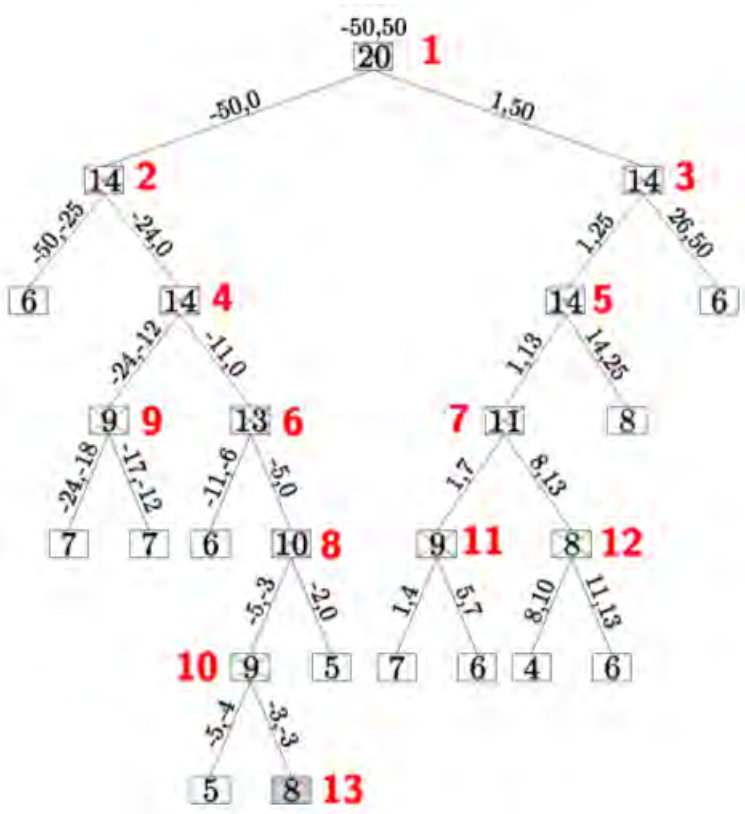


Figura 3.6: Ejemplo de Árbol Binario con Solución. Lemström *et al.* [Lemström04]

cadenas 1 y cadena 2, tomando en cuenta la cota superior sirviendo como la restricción de los nuevos rangos. Es decir, de acuerdo a la Figura 3.6 del árbol se calcula entonces la longitud LCS con limite**[-50, 0]** del nodo izquierdo, dando un valor de LCS de **14**. Y para la rama **[1,50]** el valor LCS también de **14** correspondiente al nodo derecho.

Teniendo los valores de longitud LCS calculados, estos son ingresados a una cola de prioridad. Posteriormente, para la siguiente iteración se verifica en la cola de prioridad el valor mayor de longitud LCS almacenado. El valor mayor de longitud LCS funge como subárbol más prometedor por lo tanto se calculan sus ramas y se saca de la cola de prioridad.

Entonces como resumen, en cualquier momento, se tiene un conjunto de subárboles a considerar, cada uno con su propia cota superior de las hojas que lo contienen. Ese conjunto de subárboles a tener en cuenta se mantiene en una cola de prioridad máxima. A cada paso del algoritmo, se toma el subárbol más prometedor, se calcula sus dos hijos, y se añade al

conjunto de subárboles para consideración. Si el subárbol más prometedor resulta ser un nodo hoja *[t, t]*, es decir mismo valor, entonces el valor de la cota superior es de hecho el valor **t** exacto $LCS^t$ calculado, entonces aquí se termina el proceso.

En el ejemplo de la Figura 3.6 la solución al problema de las cadenas:

**Cadena 1:** 11,30,21,12,13,44,33,34,39,21,6,1,36,2,47,45,23,40,24,17

**Cadena 2:** 8,24,46,44,16,39,12,18,10,41,23,42,17,3,44,20,41,35,7,37

Se tiene que la longitud entre ambas es de **8**, si ese valor se convierte a medida de distancia se obtiene que la distancia entre la cadena 1 y la cadena 2 es de **24**. El valor de longitud es 8 por que se llegó al nodo hoja *[t, t]*, es decir rango de transposición *[-3, -3]*, por lo tanto en la cola de prioridad ya no se encuentra un nodo que al cálcular a sus 2 nodos hijos de un valor mayor a 8. Cabe mencionar, que al inicio el nodo raíz es de valor *20*, pero no puede parar ahí ya que no es la solución óptima, puesto que el algortimo va formando el árbol binario mientras los valores del rango de transposición son diferente $[\tau \neq \tau']$.

**Cálculo de longitud LCS con un rango de transposición utilizando el algoritmo Branch and Bound en base a una recurrencia**

Para el cálculo de $LCS^{[\tau,\tau']}(A, B)$ se tiene la recurrencia (3.9):

$$\begin{aligned} LCS_{i,0}^{[\tau,\tau']} &= 0 \ , \forall \ 0 \leq i \leq N \\ LCS_{0,j}^{[\tau,\tau']} &= 0 \ , \forall \ 0 \leq j \leq M \end{aligned} \tag{3.9}$$

$$LCS_{i,j}^{[\tau,\tau']} \begin{cases} 1 + LCS_{i-1,j-1}^{[\tau,\tau']} & \text{Si } \tau \leqslant b_j - a_i \leqslant \tau' \\ max(LCS_{i-1,j}^{[\tau,\tau']}, LCS_{i,j-1}^{[\tau,\tau']}) & \text{Sino} \end{cases}$$

donde

| | |
|---|---|
| $A$ | Secuencia de caracteres 1 de tamaño N. |
| $B$ | Secuencia de caracteres 2 de tamaño M. |
| $(i-1, j-1)$ | Restricciones locales, que marcan la trayectoria óptima o el camino de |
| $(i-1, j)$ | alineamiento. |
| $(i, j-1)$ | |
| $LCS^{[\tau,\tau']}$ | Contiene los LCS con relación a la cota superior de $[\tau, \tau']$. |

## 3.3. Conclusiones del capítulo

En este capítulo se mencionan las técnicas de comparación empleadas en la investigación, estas técnicas están basadas en programación dinámica. La programación dinámica consiste en la descomposición de un problema general en problemas más pequeños que al ser resueltos de forma independiente, organizan la ejecución, la programación y reducen costos innecesarios de ejecución.

Las técnicas de procesamiento de cadenas o secuencias biológicas (ADN), son utilizadas en varias áreas de las ciencias computacionales, el manejo de cadenas, también se usa en bioinformática para encontrar subcadenas de genes que son comunes en varios organismos, es decir localizar si una secuencia de ADN es similar a otra secuencia de ADN. Cabe mencionar, que las técnicas de procesamiento de cadenas también son muy utilizadas en el ámbito musical, prueba de ello es la presente investigación, el cual se implementó la distancia Levenshtein, distancia Indel, distancia TWLCS para poder comparar dos interpretaciones musicales.

Por otra parte se considera, que una característica particular en el análisis de la música es, la transposición de notas musicales: La misma melodía es percibida incluso si la secuencia del tono es desplazado de una tecla a otra. Esto es equivalente a la adición de una constante para todos los valores de tono de una secuencia. Así, el problema de determinar qué tan parecidas son dos cadenas bajo invariancia a transposición es de interés en la recuperación de la música. Por lo tanto se concluye que para resolver este problema fue útil la implementación de la distancia invariante a transposición InDel.

Capítulo 4

# Descripción del sistema implementado

En este capítulo se describe la implementación del sistema de evaluación automática de estudiantes de música mediante datos simbólicos, componiéndose de la etapa de extracción de los datos simbólicos contenidos en los archivos MIDI y la etapa de comparación para obtener la evaluación de la interpretación del estudiante.

## 4.1. Etapa de extracción de los datos simbólicos

Para este trabajo se obtuvo una colección de archivos MIDI. Las piezas interpretadas por el profesor son las interpretaciones de referencia base consideradas como correctas y las piezas interpretadas por los estudiantes son las interpretaciones de prueba que serán evaluadas.

Para la etapa de extracción de los datos simbólicos contenidos en ambos archivos MIDI (referencia base e interpretación del estudiante) se analizó lo siguiente.

La información contenida en un archivo MIDI está organizada de acuerdo a la Tabla 4.1. Conformado por la cabecera de archivo donde se muestra información como: nombre del archivo, tamaño o número de ticks, duración total, tipo de archivo, resolución en ticks por segundo, tempo[1], etc. También indica el número de pistas que contiene el

[1]Tempo: Hace referencia a la velocidad con la que debe ejecutar una pieza musical.

archivo, cada pista tiene un tamaño el cual indica el número de eventos ocurridos en la pista, cada evento está conformado por un mensaje de nota. El número de eventos en cada pista puede variar.

Tabla 4.1: Estructura del archivo MIDI.

| |
|---|
| Información de cabecera del archivo |
| Número de pistas |
| Pista 1: Contiene número de eventos |
| Mensaje 1: nota, velocidad, número de ticks |
| Mensaje 2: nota, velocidad, número de ticks |
| Mensaje 3: nota, velocidad, número de ticks |
| . |
| Mensaje n: nota, velocidad, número de ticks |
| Pista 2: Contiene número de eventos |
| Mensaje 1: nota, velocidad, número de ticks |
| Mensaje 2: nota, velocidad, número de ticks |
| Mensaje 3: nota, velocidad, número de ticks |
| . |
| Mensaje n: nota, velocidad, número de ticks |
| . |
| Pista n: Contiene número de eventos |
| Mensaje 1: nota, velocidad, número de ticks |
| Mensaje 2: nota, velocidad, número de ticks |
| Mensaje 3: nota, velocidad, número de ticks |
| . |
| Mensaje n: nota, velocidad, número de ticks |

Los mensajes contenidos en el MIDI almacenan información como son los parámetros: nota, velocidad, número de ticks. Los parámetros *nota* y *velocidad* en un archivo MIDI son cuantificados en valores entre 0 a 127. El parámetro *número de ticks* en música se toma

un marcador o un tictac, el cual se puede denominar como *pulsos por segundo*. El valor de ticks es convertido a segundos para obtener la duración real de una nota musical pulsada, posteriormente este valor de duración real es convertido a tiempos musicales, es decir, duraciones musicales estilo partituras. Los parámetros antes mencionados son los utilizados al momento de comparar las interpretaciones musicales. Para el manejo de la información en esta etapa de extracción de los datos simbólicos, se utilizó un paquete de la plataforma Java nombrado (javax.sound.midi.*) desarrollado por Oracle [Java SE Documentation16].

A continuación se muestra en la Figura 4.1 un ejemplo de la información contenida en un archivo MIDI, mostrando información de cabecera y una pequeña parte de los eventos ocurridos en la pista 1 del archivo.

```
run:
DumpSequence: usage:
        java DumpSequence <midifile>
------------------------------------------------------------------------
Archivo: FROZEN.mid
------------------------------------------------------------------------
Tamaño: 307200 ticks
Duración: 29538464 microsegundos
------------------------------------------------------------------------
Tipo División: PPQ
Resolución: 9600 ticks por segundos
------------------------------------------------------------------------
tick 0: SMTPE Offset: 96:0:0.0.0
tick 0: Set Tempo (µs/cuarto de nota): 923077
tick 0: Termina
------------------------------------------------------------------------
Pista 1:
-----------------------
Número de eventos en la pista: 164
tick 0: Sequence/Track Name: Inst 1
tick 0: Instrument Name: Mini Grand 1
tick 0: channel 1: program change 0
tick 0: channel 1: Mensaje: Note On    Pitch: 79   Velocidad: 95
tick 2160: channel 1: Mensaje: Note Off    Pitch: 79   Velocidad: 0
tick 2400: channel 1: Mensaje: Note On    Pitch: 80   Velocidad: 95
tick 4560: channel 1: Mensaje: Note Off    Pitch: 80   Velocidad: 0
tick 4800: channel 1: Mensaje: Note On    Pitch: 72   Velocidad: 95
tick 6960: channel 1: Mensaje: Note Off    Pitch: 72   Velocidad: 0
tick 7200: channel 1: Mensaje: Note On    Pitch: 79   Velocidad: 95
tick 11520: channel 1: Mensaje: Note Off    Pitch: 79   Velocidad: 0
tick 12000: channel 1: Mensaje: Note On    Pitch: 80   Velocidad: 95
tick 18480: channel 1: Mensaje: Note Off    Pitch: 80   Velocidad: 0
tick 19200: channel 1: Mensaje: Note On    Pitch: 79   Velocidad: 95
tick 21360: channel 1: Mensaje: Note Off    Pitch: 79   Velocidad: 0
tick 21600: channel 1: Mensaje: Note On    Pitch: 80   Velocidad: 95
```

Figura 4.1: Información contenida en un archivo MIDI

### 4.1.1. Obtención de las notas musicales

Las notas musicales también pueden verse en valores de tonos (pitch) con relación a cada nota musical en su correspondiente octava de acuerdo a lo descrito por Reverte y Hidalgo en [Reverte06], véase la Tabla 4.2.

Tabla 4.2: Valores de los tonos correspondientes a cada nota en las diferentes octavas, Reverte y Hidalgo en [Reverte06].

| Octava | Nota | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Do** | **Do#** | **Re** | **Re#** | **Mi** | **Fa** | **Fa#** | **Sol** | **Sol#** | **La** | **La#** | **Si** |
| **0** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| **1** | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| **2** | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |
| **3** | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| **4** | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 |
| **5** | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 |
| **6** | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 |
| **7** | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| **8** | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 |
| **9** | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| **10** | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | | | | |

Este valor en un archivo MIDI es cuantificado mediante un número entero positivo comprendido entre 0 a 127 cubre casi 11 octavas, valor obtenido del parámetro *nota* contenido en el mensaje MIDI.

### 4.1.2. Cálculo de duraciones reales de las notas musicales

Para obtener las duraciones de las notas presionadas en la interpretación musical, se toma el parámetro *número de ticks*, este valor tomado de dos mensajes, es decir *Note on* y *Note Off* correspondiente a una misma nota musical. El mensaje Note On indicaría el número de ticks de inicio de nota y el mensaje Note Off indicaría el número de ticks de

final de nota. Se hace uso de la conversión de tick a segundos con (4.1):

$$dR = \frac{I - F}{r} \tag{4.1}$$

donde

- $dR$ Duración real en *seg.* que una nota musical fue presionada.
- $I$ Inicio de nota en Ticks.
- $F$ Final de nota en Ticks.
- $r$ Valor de resolución en ticks por segundo.

Cabe mencionar que, en (4.1) el valor de $F$ (final de nota en ticks) siempre será mayor a $I$ (inicio de nota en ticks), puesto que va aumentando a medida que va transcurriendo el tiempo en la pieza musical. El valor de los ticks siempre inicia en 0 en la interpretación musical y va incrementando.

### 4.1.3. Conversión de tiempos reales a tiempos musicales estilo partituras

Una vez que se ha obtenido la duración de las notas de acuerdo a la codificación del archivo MIDI. Es conveniente realizar una conversión de los tiempos reales, es decir, duraciones reales de cada nota, a tiempos musicales **(duraciones musicales)** estilo partituras. Recordando que en la música, la duración se establece siempre desde un punto de vista proporcional, mediante las llamadas *figuras musicales* véase Tabla 4.3. Las figuras musicales son una representación gráfica de la duración de un sonido. La figura que representa la duración mayor es la nota *redonda*, y a partir de ella se pueden definir la nota *blanca*, *negra*, *blanca con punto* etc. Hay notas que tienen duración de 1 tiempo, de 1/4 de nota, de 1/8, etc.

En el previo análisis de los archivos MIDI se detectó que estos valores de duraciones están en valor de tipo decimal dados en segundos, por lo que a la hora de realizar la comparación de las interpretaciones, sería casi imposible comparar la misma secuencia de símbolos dos veces. Es decir, un profesor de música, si interpretará la misma melodía dos veces, es poco probable que al presionar una misma tecla musical se realice exactamente con la misma duración dos veces, ya que esta duración calculada con la información del MIDI se ve reflejada en valor decimal.

Tabla 4.3: Duración de notación musical.

| *Figura Musical* | *Nombre de Nota* | *Valor* | *Tiempo* |
|---|---|---|---|
|  | Redonda | 4 | → 1 Nota con mayor duración |
| | Negra | 1 | → 1/4 de nota (con respecto de Redonda) |
| | Blanca | 2 | → 2/4 de nota (con respecto de Redonda) |
| | Blanca con punto | 3 | → 3/4 de nota (con respecto de Redonda) |

Ejemplo: suponiendo que se tiene la **cadena 1:** 26, 0.79, 127 y la **cadena 2:** 26, 0.81, 127, conteniendo el mismo valor de pitch 26 (correspondiendo a la nota *Re* en la octava 2) y mismo valor de velocidad 127, pero diferente valor de duración real calculada con la información del MIDI, véase Tabla 4.4 mostrando valores de duración en decimal. Por lo

Tabla 4.4: Ejemplo de cadenas con duraciones decimales.

| | Pitch | Duración Real | Velocidad |
|---|---|---|---|
| Cadena 1: | 26 | **0.79** | 127 |
| | ↓ | ↓ | ↓ |
| Cadena 2: | 26 | **0.81** | 127 |

que a la hora de comparar la cadena se penalizaría como error, siendo que no hay una diferencia considerable para que tome una penalización grande y esto se vea reflejado en la calificación asignada. La diferencia radica en **0.02 segundos**, que en la práctica no se percibiría auditivamente, por lo tanto si se tomaran las secuencias de símbolos tal cual se mostró arriba no habría flexibilidad al momento de comparar las cadenas.

Por lo antes mencionado, se decide convertir las duraciones reales obtenidas de cada nota, a tiempos musicales (duraciones musicales) como se leen en una partitura a la hora de interpretar una pieza musical, permitiendo brindar un nivel de flexibilidad para la comparación de dos secuencias de símbolos, utilizando (4.2).

$$dM = \lceil \frac{dR \times 1000000}{te} \rceil \tag{4.2}$$

donde

$dM$ Valor en cuartos de nota (se define como duración musical) .

$dR$ Duración real en *seg.* que una nota musical fue presionada.

$te$ Valor de tempo en $\mu s$ sobre cuartos de nota.

Con (4.2)se obtienen los valores dados en cuartos de nota como se ve en la Tabla 4.5. Siendo ahora el valor convertido como **duración en tiempo musical**, ese valor es asignado a la cadena de caracteres para la comparación de las cadenas.

Tabla 4.5: Valores en Tiempos Musicales estilo Partituras.

| Valor asignado a la cadena | Significado en tiempos musicales |
|---|---|
| 5 | Más de 1 tiempo |
| 4 | 4/4 de nota |
| 3 | 3/4 de nota |
| 2 | 2/4 de nota |
| 1 | 1/4 de nota |

### 4.1.4. Cuantización de las velocidades de las notas musicales

La velocidad correspondiente a cada nota musical, es reflejada como *volumen* ante el oído humano y conocida también como *Dinámica*, Reverte y Hidalgo en [Reverte06]. El parámetro de velocidad de la nota interpretada se cuantifica con valores de 0 a 127 ordenados de menor a mayor intensidad, este valor dependiendo de la fuerza con la que se pulse la tecla musical. Debido a lo anterior se presenta el siguiente problema:

Como se expuso antes, el valor de la nota (pitch), la velocidad y la duración, es la información necesaria para la realización de un sistemas de recuperación o comparación en la música, para evaluar dos interpretaciones es necesario comparar de cada cadena el pitch, duración y velocidad o fuerza de cada nota.

Ejemplo: suponiendo que se tiene la interpretación referencia base y la interpretación del estudiante. Cada interpretación solo tiene una pista y un solo evento almacenado, puede ser un evento *Note on* de una nota pulsada y ese es el evento que se quiere comparar. Se tienen dos cadenas a comparar véase Tabla 4.6, una cadena por cada intepretación. Se va a comparar **cadena 1:** 60, 4, 100 de la referencia base y **cadena 2:** 60, 4, 50 de la interpretación del estudiante. Por lo tanto no son iguales por el valor de velocidad.

Tabla 4.6: Ejemplo de cadenas mostrando velocidades de las notas pulsadas.

| | Pitch | Duración Musical | Velocidad |
|---|---|---|---|
| Cadena 1: | 60 | 4 | **100** |
| | ↓ | ↓ | ↓ |
| Cadena 2: | 60 | 4 | **50** |

El problema radica en el parámetro de velocidad, es muy poco probable que se presione con la misma intensidad una nota musical dos veces, por lo tanto no se tendría una cadena con la misma secuencia de símbolos en dos ocasiones. Si se interpretará una misma melodía dos veces por un profesor de música, es poco probable que realice la ejecución por segunda vez exactamente igual a la primera ejecución, con relación a la intensidad o fuerza con la que presionó las teclas musicales la primera vez.

Por lo tanto, se decidió cuantizar este parámetro para proporcionar un nivel de flexibilidad a la interpretación musical. Dividiendo los valores comprendidos del 1 al 127 en 6 rangos. Para ello se generó un archivo MIDI de prueba con la ayuda de la librería (jmusic) desarrollada en Java [Sorensen16], generando una secuencia de notas musicales con un mismo valor de pitch, mismo valor de duración y la variación del 1 al 127 en el parámetro de velocidad. De esta manera se identificaron los rangos en donde los valores de las velocidades se escuchan similares (reflejado en volumen), véase la Tabla 4.7. Cabe mencionar que al escuchar una nota que fue pulsada con una fuerza de valor 22 se refleja igual en volumen de otra nota que fue pulsada con un valor de 38, por lo tanto al escuchar ambas notas la apreciación auditiva es la misma. En la columna 3 se describe el nombre que se da en un rango específico de valores de velocidad de acuerdo a la fuerza ejercida a

una nota sobre el teclado musical, Héctor Murcia en [Murcia06].

Tabla 4.7: Cuantización del Parámetro de Velocidad.

| Rango de velocidad en el mensaje MIDI. Héctor Murcia en [Murcia06] | Valor asignado a la cadena | Nombre Héctor Murcia en [Murcia06] | Ejecución |
|---|---|---|---|
| 0 | 0 indica no hay presión de tecla musical. | - | - |
| 1 - 22 | 1 | pianíssimo | Muy suave |
| 23 - 43 | 2 | piano | Suave |
| 44 - 64 | 3 | mezzo piano | No tan suave |
| 65 - 85 | 4 | mezzo forte | Poco fuerte |
| 86 - 106 | 5 | forte | Fuerte |
| 107 - 127 | 6 | fortísimo | Muy fuerte |

La codificación final respecto al parámetro de velocidad que se utiliza para concatenar a la cadena, corresponde del valor 1 al 6.

### 4.1.5. Almacenamiento de los datos simbólicos extraídos

Cabe mencionar que no se tiene una sincronización entre las pistas del MIDI, simplemente se verifica las cadenas de la pista 1 de la referencia base con las cadenas de la pista 1 de la interpretación del estudiante, esto se hace lo mismo con las pistas restantes de ambos MIDIS's, considerándolas como cadenas independientes cada pista. Evidentemente si el estudiante se equivoca al tocar mal una nota musical, si falla en la duración o en la velocidad de la nota, se refleja e identifica el error aun así no haya una sincronización entre las pistas, es decir no se verifica en el sistema si ocurren al mismo tiempo las notas musicales respecto a la otra pista.

En el sistema de evaluación se implementa el Algoritmo 1, el cual extrae los datos simbólicos de cada evento MIDI ocurrido. El método *Datos_Simbolicos*(), obtiene el

número de eventos del MIDI y en base a ese valor define el tamaño de los vectores (pitches, velocidades, duraciones, tickInicio) mismos vectores que sirven para guardar la información simbólica correspondiente a cada evento. Se extrae el valor de *pitch* de cada evento ocurrido, el pitch corresponde a una nota musical de acuerdo a los valores de la Tabla 4.2. Se extrae el *valorTick* es útil para el cálculo de las duraciones reales de las notas haciendo uso de (4.1) y posteriormente convertir duraciones reales a duraciones musicales con (4.2). También se extrae el valor de *velocidad*, ahora bien si velocidad es diferente de 0 indica que se pulsó una tecla musical, ocurrió un evento *Note on*, por lo tanto se guardan los valores extraídos del evento los cuales son el pitch, valorTick y se cuantiza la velocidad en rangos como se ve en la Tabla 4.7, se almacenan en los vectores pitches, tickInicio y velocidades. La variable *auxIngresa* incrementa cada vez que se ingresan valores a los vectores (pitches, tickInicio, velocidades), variable que indica el número de mensajes *Note on* que han ocurrido o de igual manera indica hasta que posición de los vectores contienen información.

Por el contrario, si el valor de velocidad extraído es igual a 0 indica que se liberó una nota musical que previamente fue pulsada, entonces ocurrió un evento de *Note off*. Por lo tanto, se hace una búsqueda en el vector pitches de la nota correspondiente a la nota liberada, una vez identificada la nota se utiliza también el valor guardado previamente (valorTick que es el de inicio) en el vector tickInicio correspondiente a la misma nota musical. Ya identificada la nota y su respectivo valor de tick de inicio se calcula el tiempo que duró presionada (es decir, su duración), para ello se hace uso de 4.1 y 4.2. Ya calculada la duración de la nota ese valor es guardado en el vector duraciones. De esta manera se obtiene la información de los eventos del MIDI.

**Algoritmo 1: Datos_Simbolicos()**

```
1:  numE ← Extraer número de eventos del MIDI
2:  Definir pitches[numE/2.longitud]
3:  Definir velocidades[numE/2.longitud]
4:  Definir duraciones[numE/2.longitud]
5:  Definir tickInicio[[numE/2.longitud]]
6:  Definir auxIngresa ← 0
7:  para i ← 0 to numE hacer
8:    pitch ← Extraer pitch del evento ocurrido
9:    valorTick ← Extraer valor de tick del evento ocurrido
10:   velocidad ← Extraer velocidad del evento ocurrido
11:   Definir dR ← 0
12:   si velocidad != 0 entonces
13:     pitches[auxIngresa] ← pitch
14:     velocidades[auxIngresa] ←Cuantizar Velocidad Tabla 4.7
15:     tickInicio[auxIngresa] ← valorTick
16:     auxIngresa++
17:   si no
18:     auxBusca ← 0
19:     ingresado ← false
20:     mientras (ingresado != true) & (auxBusca < auxIngresa) hacer
21:       si (duraciones[auxBusca] = 0) & (pitches[auxBusca] = pitch) entonces
22:         dR ← Calcular dR con (4.1), dado el valor correspondiente en el vector
            tickInicio
23:         duraciones[auxBusca] ← Convertir dR a dM con (4.2)
24:         ingresado ← true
25:       fin si
26:       auxBusca++
27:     fin mientras
28:   fin si
29: fin para
```

Al extraer la información perteneciente a cada evento de cada pista del MIDI, se prosigue a hacer el almacenamiento en una estructura de datos llamada *lista dinámica*, teniendo por cada interpretación musical una lista. Cada posición de la lista dinámica almacena toda la información de los mensajes ocurridos en cada evento perteneciente a una sola pista del MIDI, es por ello que se tienen pistas independientes las cuales se van a comparar, pero no existe una sincronización entre las pistas. Se guarda el valor del pitch, duración y velocidad de cada nota musical, así que toda la información contenida en una pista del MIDI es almacenada en una posición de la lista dinámica.

Suponiendo que se tiene una interpretación con 3 pistas tanto en el archivo MIDI de referencia base como en el MIDI de la interpretación del estudiante, entonces cada lista dinámica sería de tamaño 3, como se ve en la Tabla 4.8. La información de la *pista 1* del MIDI de referencia base se almacena en la *posición 0* de la lista dinámica, la información de la *pista 2* del MIDI se almacena en *posición 1* de la lista, la información de la *pista 3* del MIDI se almacena en *posición 2* de la lista, esto mismo sucede con la lista dinámica donde se almacena la información del MIDI de la interpretación del estudiante. En la Tabla 4.8 se indica el conjunto de pistas por cada interpretación y con $p$ la posición de la lista dinámica, seguido de la información correspondiente a la pista.

Cabe mencionar que, para el cálculo de la distancia invariante a transposición InDel, fue necesario hacer una modificación del almacenamiento, ya que en las recurrencias a utilizar a cada cadena se le suma o resta un valor de desplazamiento $t$, por lo tanto se requirió separar los datos (pitch, duración y velocidad), y convertirlos a valores enteros.

Tabla 4.8: Almacenamiento de los Datos Simbólicos de las Interpretaciones Musicales.

**Referencia Base: contiene 3 pistas**

| | | | | |
|---|---|---|---|---|
| *Pistas* | p(0)→ | Pitch ⊔ Duración ⊔ Velocidad | ... | Pitch ⊔ Duración ⊔ Velocidad |
| | | Mensaje 1 | ... | Mensaje n |
| | p(1)→ | Pitch ⊔ Duración ⊔ Velocidad | ... | Pitch ⊔ Duración ⊔ Velocidad |
| | | Mensaje 1 | ... | Mensaje n |
| | p(2)→ | Pitch ⊔ Duración ⊔ Velocidad | ... | Pitch ⊔ Duración ⊔ Velocidad |
| | | Mensaje 1 | ... | Mensaje n |

**Interpretación Estudiante : contiene 3 pistas**

| | | | | |
|---|---|---|---|---|
| *Pistas* | p(0)→ | Pitch ⊔ Duración ⊔ Velocidad | ... | Pitch ⊔ Duración ⊔ Velocidad |
| | | Mensaje 1 | ... | Mensaje n |
| | p(1)→ | Pitch ⊔ Duración ⊔ Velocidad | ... | Pitch ⊔ Duración ⊔ Velocidad |
| | | Mensaje 1 | ... | Mensaje n |
| | p(2)→ | Pitch ⊔ Duración ⊔ Velocidad | ... | Pitch ⊔ Duración ⊔ Velocidad |
| | | Mensaje 1 | ... | Mensaje n |

## 4.2. Etapa de comparación de los datos simbólicos

Una vez extraídos los datos simbólicos de los archivos MIDI, se prosigue a la etapa de comparación. Para la realización de esta etapa se hizo uso de las técnicas de comparación mencionadas en el Capítulo 3. El Algoritmo 2 recibe como parámetros las dos listas dinámicas (Lista_Ref, Lista_Int) con la información correspondiente a las pistas de los MIDI's de la referencia base y la interpretación del estudiante.

**Algoritmo 2: Comparar(Lista_Ref, Lista_Int)**

1: $sumaDistancias \leftarrow 0$
2: **para** $i \leftarrow 0$ **to** $Lista\_Ref.longitud$ **hacer**
3: $valorDistancia \leftarrow tecnica\_comparacion(Lista\_Ref(i), Lista\_Int(i))$
4: $sumaDistancias \leftarrow sumaDistancias + valorDistancia$
5: **fin para**
6: **devolver** $sumaDistancias$

El Algoritmo 2, el método *tecnica_comparacion*() hace referencia a las diferentes técnicas de comparación utilizadas, es decir, distancia de Levenshtein, distancia Indel, distancia invariante a transposición InDel y distancia TWLCS. Enviando como parámetros las cadenas de la información de cada pista a cada técnica.

Es decir, se tiene la referencia base y la interpretación del estudiante, por lo tanto se tiene una lista dinámica para cada interpretación *Lista_Ref* y *Lista_Int*. Entonces se va mandando a la par la información (cadenas) contenida en cada posición de las listas.

*tecnica_comparacion*( Lista_Ref (0) , Lista_Int(0) )

Se envía a la técnica de comparación el conjunto de cadenas, es decir la *posición 0* de la Lista_Ref y la *posición 0* de Lista_Int, esa comparación retorna el *valorDistancia* de ambos conjuntos de cadenas. Posteriormente se envía, el conjunto de cadenas en la siguiente posición de las listas. Es decir, la información contenida en la *posición 1* de ambas listas.

*tecnica_comparacion*( Lista_Ref (1) , Lista_Int(1) )

De igual manera esta comparación retorna un nuevo valor de distancia (*valorDistancia*) entre ese conjunto de cadenas. Por lo que, se van sumando los *valorDistancia* calculados para obtener el valor total de distancia, es decir *sumaDistancias*. Este *sumaDistancias* es el valor final acumulado de acuerdo al Algoritmo 2, de esta manera al terminar la ejecución se proporciona la medida de distancia entre la referencia base y la interpretación del estudiante.

Una vez descrito lo anterior, a continuación se muestran los algoritmos de las técnicas de comparación implementadas. El Algoritmo 3 calcula la distancia de Levenshtein, recibe como parámetros las cadenas referentes a los datos simbólicos de la referencia base e interpretación del estudiante, define una matriz D la cual almacena los cálculos al momento de comparar los caracteres de cadena referencia base e interpretación del estudiante. Inicializa primera fila y columna de la matriz, posteriormente compara cadena referencia base con interpretación del estudiante. Finalmente retorna la medida de distancia entre los dos conjuntos de cadenas. De igual manera el Algoritmo 4 calcula la distancia Indel declarando una matriz e inicializando fila y columna de la matriz para proseguir a comparar las cadenas. El Algoritmo 5 obtiene la longitud LCS, pero retornando valor en medida de distancia (utiliza la conversión (3.4)). El Algoritmo 6 calcula la medida en distancia de TWLCS. Todos estos algoritmos reciben como parámetros las cadenas referentes a los datos simbólicos de la referencia base e interpretación del estudiante, procesan las cadenas y retornar una medida de distancia. La diferencia entre los algoritmos radica en la recurrencia que utiliza cada uno, como se expuso en el Capítulo 3.

**Algoritmo 3: distLevenshtein**($CadRef, CadInt$)

1: $m \leftarrow CadRef.longitud$, $n \leftarrow CadInt.longitud$
2: Definir $D(m+1, n+1)$
3: **para** $i \leftarrow 0$ **to** $i \leq m$ **hacer**
4: $\quad D(i, 0) \leftarrow i$
5: **fin para**
6: **para** $j \leftarrow 0$ **to** $i \leq n$ **hacer**
7: $\quad D(0, j) \leftarrow j$
8: **fin para**
9: **para** $i \leftarrow 1$ **to** $i \leq m$ **hacer**
10: $\quad$ **para** $j \leftarrow 1$ **to** $j \leq n$ **hacer**
11: $\quad\quad$ **si** $CadRef(i-1) = CadInt(j-1)$ **entonces**
12: $\quad\quad\quad D(i, j) \leftarrow D(i-1, j-1)$
13: $\quad\quad$ **si no**
14: $\quad\quad\quad D(i, j) \leftarrow 1 + min(D(i-1, j-1), D(i-1, j), D_(i, j-1))$
15: $\quad\quad$ **fin si**
16: $\quad$ **fin para**
17: **fin para**
18: **devolver** $D(m, n)$

**Algoritmo 4: distIndel($CadRef, CadInt$)**

```
1: m ← CadRef.longitud, n ← CadInt.longitud
2: Definir D(m + 1, n + 1)
3: para i ← 0 to i ≤ m hacer
4:    D(i, 0) ← i
5: fin para
6: para j ← 0 to i ≤ n hacer
7:    D(0, j) ← j
8: fin para
9: para i ← 1 to i ≤ m hacer
10:   para j ← 1 to j ≤ n hacer
11:      si CadRef(i − 1) = CadInt(j − 1) entonces
12:         D(i, j) ← D(i − 1, j − 1)
13:      si no
14:         D(i, j) ← 1 + min(D(i − 1, j), D(i, j − 1))
15:      fin si
16:   fin para
17: fin para
18: devolver  D(m, n)
```

**Algoritmo 5: distLCS**($CadRef, CadInt$)

```
1: m ← CadRef.longitud, n ← CadInt.longitud
2: Definir LCS(m + 1, n + 1)
3: para i ← 0 to i ≤ m hacer
4:    LCS(i, 0) ← 0
5: fin para
6: para j ← 0 to i ≤ n hacer
7:    LCS(0, j) ← 0
8: fin para
9: para i ← 1 to i ≤ m hacer
10:   para j ← 1 to j ≤ n hacer
11:      si CadRef(i − 1) = CadInt(j − 1) entonces
12:         LCS(i, j) ← LCS(i − 1, j − 1) + 1
13:      si no
14:         LCS(i, j) ← max(LCS(i, j − 1), LCS(i − 1, j))
15:      fin si
16:   fin para
17: fin para
18: Convertir distancia ← CadRef.longitud + CadInt.longitud − (2 ∗ LCS(m, n))
19: devolver distancia
```

**Algoritmo 6: distTWLCS($CadRef, CadInt$)**

```
1: m ← CadRef.longitud, n ← CadInt.longitud
2: Definir TW(m + 1, n + 1)
3: para i ← 0 to i ≤ m hacer
4:    TW(i, 0) ← i
5: fin para
6: para j ← 0 to i ≤ n hacer
7:    TW(0, j) ← j
8: fin para
9: para i ← 1 to i ≤ m hacer
10:   para j ← 1 to j ≤ n hacer
11:      si CadRef(i − 1) = CadInt(j − 1) entonces
12:         DT(i, j) ← min(DT(i, j − 1), DT(i − 1, j), DT(i − 1, j − 1))
13:      si no
14:         DT(i, j) ← 1 + min(DT(i, j − 1), DT(i − 1, j))
15:      fin si
16:   fin para
17: fin para
18: devolver  DT(m, n)
```

Ahora, el proceso del **cálculo de la distancia invariante a transposición InDel de manera directa** se obtiene con el Algoritmo 7 este método recibe como parámetros el conjunto de datos de ambas interpretaciones (referencia base y interpretación del estudiante) y un valor de *rango* que indica el límite de desplazamiento. Para fin del proyecto esta variable *rango* tiene un valor de 127, debido a que 127 es el número de tonos que se tienen en un teclado. Así se asegura que se calcula la distancia invariante a transposición aun así, se tenga un desplazamiento de nota hasta de 127 tonalidades.

Ejemplo: De acuerdo a los valores de la Tabla 4.2, se tiene que presionar la nota de *Do* correspondiente a la *octava 0* teniendo un valor de pitch de *0*, pero no ocurrió así y se presionó hasta la nota *Sol* de la *octava 10* con un valor de pitch de *127*, teniendo un desplazamiento de hasta 127 tonos por lo tanto inicializando el *rango=127* se asegura que se obtendrá el cálculo de la distancia invariante a transposición hasta ese valor de desplazamiento.

Retomando el análisis del Algoritmo 7, la variable $t$ representa el valor de desplazamiento incrementando de 1 en 1, dicho valor es sumado o restado a cada cadena de datos. Al iniciar el algoritmo, es necesario tener una distancia mínima (guardada en variable *mínimo*), esta variable se inicializa con el cálculo de la distancia invariante a transposición con un valor de desplazamiento (t=0) por primera vez. Cabe mencionar que en el primer cálculo de distancia invariante a transposición, si el valor contenido en *mínimo* es igual a 0, significa que la melodía a ser evaluada fue interpretada de manera correcta, aun así hubiera ocurrido un desplazamiento de notas (teniendo una distancia de 0 entre la referencia base e interpretación del estudiante). Para el cálculo de la distancia invariante a transposición con un desplazamiento $t$, se hace uso del Algoritmo 5 mencionado antes, con la diferencia de enviar un parámetro extra al método **distLCS(**$CadRef, CadInt, t$**)** siendo **t** el parámetro adicional y correspondiente al valor de desplazamiento. Además agregando un cambio en el condicional ***si*** **(**$CadRef(i-1)+t = CadInt(j-1)$**)**, es decir se le adiciona a la cadena referencia base (indicada con CadRef) el valor de desplazamiento $t$ a cada carácter de la cadena.

Posteriormente al paso de cada iteración como se dijo antes, el valor de $t$ va incrementando de 1 en 1, para calcular el nuevo valor de distancia invariante a transposición

con un nuevo valor de desplazamiento. El nuevo valor de distancia calculado se va almacenando en *minimoNuevo*. Lo que interesa siempre en este Algoritmo 7 es guardar en variable *mínimo* la distancia más pequeña hasta terminar las iteraciones.

**Algoritmo 7: distInvTranLCSDir**($CadRef, CadInt, rango$)

```
1: Definir minimoNuevo
2: min ← −rango , max ← rango
3: t ← 0
4: minimo ← LCS(CadRef, CadInt, t)
5: para t ← min to t < max hacer
6:    minimoNuevo ← LCS(CadRef, CadInt, t)
7:    si minimo ≥ minimoNuevo entonces
8:       minimo ← minimoNuevo
9:    fin si
10: fin para
11: devolver minimo
```

Para el proceso del **cálculo de la distancia invariante a transposición InDel utilizando la metodología Branch and Bound**, cabe mencionar que se obtiene la misma distancia con relación a que se hiciera de manera directa, en lo que difieren es en el tiempo de ejecución. Branch and Bound es un algoritmo que permite realizar menos cálculos.

Por lo tanto en la etapa de pruebas la distancia invariante a transposición InDel utilizando *Branch and Bound* debe mostrar una ejecución más rápida en comparación de si se realiza de *manera directa* al momento de la etapa de comparación de las interpretaciones musicales. Para ello se plasmará en los experimentos una tabla comparativa entre estas dos técnicas de los tiempos de ejecución por cada interpretación evaluada. El algoritmo Branch and Bound LCTS es descrito en el Algoritmo 8, consultado en el artículo Lemström *et al.* [Lemström04].

El Algoritmo 8 Branch and Bound LCTS calcula $LCTS(A, B)$. El método $ComputeSetLCS(A, B, \tau, \tau')$ calcula $LCS^{[\tau,\tau']}(A, B)$ descrito en la recurrencia (3.9).

Para la ejecución del método $ComputeSetLCS(A, B, \tau, \tau')$ con rango de transpo-

**Algoritmo 8: Branch and Bound LCTS($A, B, \sigma$)**

1: $Init(Q)$
2: $[\tau, \tau'] \leftarrow [-\sigma, \sigma]$
3: **mientras** $\tau \neq \tau'$ **hacer**
4: $\quad \theta \leftarrow \lfloor \frac{(\tau+\tau')}{2} \rfloor$
5: $\quad Insert(Q, ([\tau, \theta], ComputeSetLCS(A, B, \tau, \theta)))$
6: $\quad Insert(Q, ([\theta + 1, \tau'], ComputeSetLCS(A, B, \theta + 1, \tau')))$
7: $\quad ([\tau, \tau'], lcts) \leftarrow ExtractMax(Q)$
8: **fin mientras**
9: Convertir $distancia \leftarrow A.longitud + B.longitud - (2 * lcts)$
10: **devolver** *distancia*

sición $(\tau, \tau')$, se hace uso del Algortimo 5 mencionado antes, con la diferencia de enviar dos parámetros extras al método siendo $\tau$ y $\tau'$ los parámetros adicionales. Además agregando un cambio en el condicional *si* validando lo siguiente: $(\tau \leqslant CadInt(j) - CadRef(i) \leqslant \tau')$. Cabe mencionar que, al ejecutar el Algortimo 5 longitud LCS con los cambios antes mencionados, no hay necesidad de estar convirtiendo el valor de longitud a medida de distancia cada vez que se tiene ese valor de longitud, al final de que termina de iterar el Algoritmo 8 Branch and Bound LCTS se realiza esa conversión final.

El Algoritmo 8, muestra la cola de prioridad que almacena pares de la forma $([\tau, \tau'], val)$ y permite extraer los elementos en orden con $ExtractMax()$ obtiene el valor mayor $LCTS$ es decir $val$ almacenado en la cola de prioridad saca el valor más prometedor para calcular sus ramas e hijos. La cola de prioridad se inicializa vacío e inserta nuevos elementos utilizando $Insert()$.

Hay que tomar en cuenta que los valores calculados a lo largo de la ejecución del Algoritmo 8 son resultados en $LCTS$, es decir longitud de la subsecuencia común más larga invariante a transposición, por lo tanto al final del algoritmo se agrega la conversión (longitud a distancia), para obtener el resultado final a medida de distancia con (3.4) mencionada en el Capítulo 3.

## 4.3. Conclusiones del capítulo

En este capítulo se describe el desarrollo del sistema, para ello se plantea la etapa de extracción de los datos simbólicos contenidos en el archivo MIDI, haciendo uso del análisis realizado en el Capítulo 2, con relación a la estructura del archivo MIDI. Con el fin de obtener la información necesaria de los archivos de música, que permitiera realizar una adecuada comparación entre la referencia base e interpretación a evaluar. Para ello se requirió, tomar la información contenida en cada evento ocurrido en el MIDI correspondiente a los *mensajes de nota*, para concatenar el valor del tono, la duración y la velocidad por cada nota presionada del teclado musical. Para obtener los parámetros anteriores se requirió de la realización de las siguientes tareas:

1. Obtención de las notas musicales cuantificadas en un rango de 0-127.
2. Cálculo de duraciones reales en segundos de las notas musicales de acuerdo a la información almacenada en el archivo MIDI, tomando los valores en ticks por segundo. Duraciones que son reflejadas en valores decimales.
3. Conversión de tiempos reales (duraciones reales) a tiempos musicales (duraciones musicales) estilo partituras, esto con el fin de permitir una flexibilidad a la hora de comparar las interpretaciones mediante las secuencias de símbolos con relación a este parámetro.
4. Cuantización de las velocidades de las notas musicales dividiendo en rangos los valores de acuerdo a la intensidad o fuerza con la que se presionó una nota musical. Las velocidades en la práctica son reflejadas como volumen ante el oído humano. Finalmente una vez que se ha realizado el pre-procesamiento de los datos simbólicos, estos son almacenados en una estructura de datos (lista dinámica).

Posterior a la etapa de extracción de datos simbólicos, se plantea la etapa de comparación. Para la realización de esta etapa se hizo uso de las técnicas de comparación mencionadas en el Capítulo 3, para las cuales se desarrolló un método por cada técnica de comparación. Cada técnica proporciona una medida de distancia de acuerdo al desempeño de la interpretación a ser evaluada.

Capítulo 5

# Pruebas y resultados

En este capítulo se muestran los resultados a partir de las pruebas realizadas con el sistema de evaluación de estudiantes de música mediante datos simbólicos.

Las interpretaciones musicales utilizadas para las pruebas del proyecto fueron obtenidas en un *Estudio de Grabación Musical* en Morelia Michoacán, en donde se cuenta con todos los elementos necesarios para la captura de un archivo MIDI, es decir teniendo un sistema interconectado entre sí, en donde estudiante o profesor de música toca una pieza musical en el teclado MIDI. Transmitiendo la información producida por el teclado MIDI mediante una serie de conexiones, como se ve en Figura 5.1: cable MIDI, interface MIDI (capaz de generar sonidos de la altura musical correspondiente a la nota) y computadora, además de contar con un software (llamado Protools Versión 10.0) que tiene un motor de reproducción y grabación MIDI, mismo que proporciona el archivo MIDI.

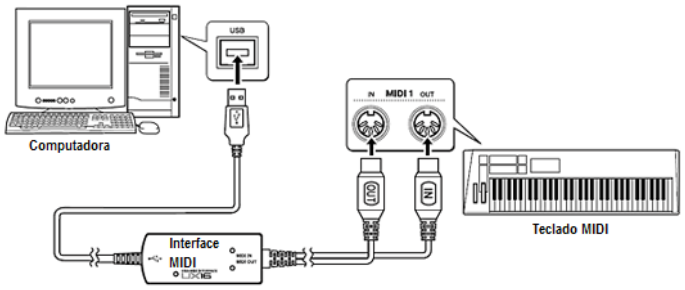


Figura 5.1: Conexión e intercambio de datos entre un teclado MIDI y computadora

Para la etapa de las pruebas fue necesario contar con una colección de interpretaciones musicales, por el que se requirió el apoyo de un músico de formación, que interpretará piezas musicales, de ahí que dichas interpretaciones son las consideradas como referencias bases, no contienen error de interpretación y son consideradas como correctas. Así mismo las interpretaciones realizadas por los estudiantes, que son las interpretaciones que se tomaron como las piezas musicales a ser evaluadas, ambas interpretaciones fueron guardadas en formato MIDI.

Una vez que se obtuvo la colección de las interpretaciones de los estudiantes para evaluar, fue necesario contar con la evaluación asignada por el profesor de música. Con el fin de comparar los resultados adquiridos por la evaluación del sistema implementado, contra los resultados correspondientes a la evaluación del profesor. De esta manera verificar si el sistema logró cumplir el objetivo, calificar de manera correcta y confiable una interpretación musical.

Cabe mencionar que las pruebas realizadas se llevaron a cabo a 13 estudiantes de música, en el cual cada alumno interpretó una melodía monofónica y una melodía polifónica, para contar con una colección de 13 interpretaciones de melodías monofónicas y otra colección de 13 melodías polifónicas, con su respectiva referencia base por cada colección de interpretaciones.

La Tabla 5.1 muestra la evaluación dada por el profesor de música de la colección de interpretaciones correspondientes a una melodía monofónica. Siendo interpretada una misma pieza musical referencia base por 13 alumnos.

La primera columna corresponde al número de estudiante, la segunda columna es la interpretación realizada por el alumno y la tercera es la evaluación dada por el profesor de acuerdo al desempeño de la interpretación del estudiante. La última columna son las observaciones en general percibidas por el profesor de música.

También, en la Tabla 5.2 se muestra la evaluación realizada por el profesor de música pero ahora con la colección de interpretaciones correspondientes a una melodía polifónica. Siendo interpretada esta misma pieza musical referencia base por los 13 estudiantes.

Tabla 5.1: Resultados de la evaluación real del profesor de música correspondiente a la colección de melodías monofónicas

| Nombre: | Frozen | | |
|---|---|---|---|
| Tipo: | Melodía Monofónica | | |
| No.Pistas: | 1 | | |
| No. Estudiante | Interpretación Estudiante | Evaluación Profesor | Observación |
| 1 | Frozen 1 | 8 | Error en notas |
| 2 | Frozen 2 | 6 | Error en notas, duraciones |
| 3 | Frozen 3 | 7 | Error duraciones, poco en notas |
| 4 | Frozen 4 | 7 | Error poco en duraciones y velocidad |
| 5 | Frozen 5 | 5 | Error en notas, duraciones, velocidad |
| 6 | Frozen 6 | 8 | Desplazamiento en notas, poco error duración |
| 7 | Frozen 7 | 9 | Bien ejecutada, desplazamiento una $8^{vb}$ |
| 8 | Frozen 8 | 7 | Ausencia de notas |
| 9 | Frozen 9 | 8 | Error notas, duraciones |
| 10 | Frozen 10 | 9 | Bien ejecutada, desplazamiento una $8^{va}$ |
| 11 | Frozen 11 | 5 | Error notas, duración, velocidad |
| 12 | Frozen 12 | 2 | Error notas, duraciones, velocidad |
| 13 | Frozen 13 | 10 | Interpretación bien |

Tabla 5.2: Resultados de la evaluación real del profesor de música correspondiente a la colección de melodías polifónicas

| Nombre: | Coldplay Clock | | |
|---|---|---|---|
| Tipo: | Melodía Polifónica | | |
| No.Pistas: | 3 | | |
| No. Estudiante | Interpretación Estudiante | Evaluación Profesor | Observación |
| 1 | Clock 1 | 8 | Error notas |
| 2 | Clock 2 | 7 | Error notas, duraciones, poco velocidad |
| 3 | Clock 3 | 8 | Error notas, poco velocidad |
| 4 | Clock 4 | 9 | Error poco duraciones |
| 5 | Clock 5 | 6 | Error poco duraciones, velocidad, poco notas |
| 6 | Clock 6 | 10 | Interpretación bien |
| 7 | Clock 7 | 3 | Error duraciones, velocidad |
| 8 | Clock 8 | 6 | Error notas, duraciones |
| 9 | Clock 9 | 7 | Ausencia de notas |
| 10 | Clock 10 | 9 | Bien ejecutada, desplazamiento una $8^{va}$ |
| 11 | Clock 11 | 10 | Bien ejecutada, poco desplazamiento de notas |
| 12 | Clock 12 | 8 | Error notas, poco velocidad |
| 13 | Clock 13 | 4 | Error duraciones, velocidad, poco notas |

## 5.1. Distancias obtenidas mediante el proceso de comparación

En la Tabla 5.3 se muestran las distancias obtenidas en la etapa de comparación de cada métrica implementada, referente a las interpretaciones de las *melodías monofónicas*. En la Tabla 5.4 mostrando las distancias obtenidas referente a las *melodías polifónicas*.

Tabla 5.3: Distancias obtenidas por las técnicas de comparación de la colección de melodías monofónicas

| **Distancias Melodías Monofónicas** | | | | | |
|---|---|---|---|---|---|
| No.Pistas: 1 | | | | | |
| No. Estudiante | Interpretación Estudiante | *Levenshtein* | *InDel* | *Invariante Trans_InDel* | *TWLCS* |
| 1 | Frozen 1 | 7 | 14 | 14 | 14 |
| 2 | Frozen 2 | 19 | 36 | 36 | 34 |
| 3 | Frozen 3 | 14 | 28 | 28 | 28 |
| 4 | Frozen 4 | 17 | 34 | 34 | 34 |
| 5 | Frozen 5 | 26 | 44 | 54 | 43 |
| 6 | Frozen 6 | 129 | 148 | 8 | 144 |
| 7 | Frozen 7 | 80 | 94 | 4 | 94 |
| 8 | Frozen 8 | 12 | 13 | 37 | 13 |
| 9 | Frozen 9 | 14 | 20 | 28 | 20 |
| 10 | Frozen 10 | 80 | 94 | 4 | 94 |
| 11 | Frozen 11 | 35 | 44 | 44 | 41 |
| 12 | Frozen 12 | 40 | 69 | 91 | 68 |
| 13 | Frozen 13 | 0 | 0 | 0 | 0 |

Tabla 5.4: Distancias obtenidas por las técnicas de comparación de la colección de melodías polifónicas

| **Distancias Melodías Polifónicas** | | | | | |
|---|---|---|---|---|---|
| No.Pistas: 3 | | | | | |
| No. Estudiante | Interpretación Estudiante | *Levenshtein* | *InDel* | *Invariante Trans_InDel* | *TWLCS* |
| 1 | Clock 1 | 16 | 23 | 31 | 23 |
| 2 | Clock 2 | 43 | 51 | 93 | 51 |
| 3 | Clock 3 | 17 | 23 | 33 | 23 |
| 4 | Clock 4 | 7 | 14 | 14 | 14 |
| 5 | Clock 5 | 50 | 76 | 108 | 76 |
| 6 | Clock 6 | 0 | 0 | 0 | 0 |
| 7 | Clock 7 | 123 | 174 | 280 | 174 |
| 8 | Clock 8 | 55 | 70 | 112 | 70 |
| 9 | Clock 9 | 30 | 35 | 87 | 35 |
| 10 | Clock 10 | 236 | 452 | 6 | 445 |
| 11 | Clock 11 | 174 | 332 | 2 | 325 |
| 12 | Clock 12 | 25 | 46 | 58 | 46 |
| 13 | Clock 13 | 90 | 141 | 247 | 174 |

## 5.2. Calificaciones obtenidas del sistema

Una vez que se tienen las medidas de distancia entre las cadenas de la referencia base e interpretación del estudiante, es necesario ponderar esos valores a una calificación de 0 a 10, con el fin de poder comparar las evaluaciones arrojadas por el sistema contra las evaluaciones dadas por el profesor.

Ahora bien, como se ha venido mencionando, los experimentos fueron hechos tanto en melodías monofónicas como polifónicas. De acuerdo a las pruebas realizadas se tiene la (5.1) empírica para convertir medida de distancias a calificación de 0 a 10.

$$calificacion = 10 - (distancia * x) \tag{5.1}$$

donde

| | |
|---|---|
| $calificacion$ | Valor ponderado entre 0 a 10. |
| $distancia$ | Valor calculado de la técnica de comparación. |
| | Distancia entre la interpretación del estudiante y la referencia base. |
| $x$ | Valor de parámetro ***0.1*** para melodías monofónicas. |
| | Valor de parámetro ***0.044*** para melodías polifónicas. |

Para poder establecer el valor de $x$, este valor fue buscado empíricamente, para ello se realizaron innumerables experimentos con los archivos de prueba MIDI's, se hizo una búsqueda exhaustiva de este parámetro $x$ (a prueba y error) para decir que el valor de parámetro $x \leftarrow 0.1$ sirvió para ponderar una calificación de 0 a 10 a partir de la medida de distancia calculada, referente a la colección de melodías monofónicas. Asimismo, utilizando un valor de parámetro $x \leftarrow 0.044$ para la colección de melodías polifónicas. La calificación obtenida se válida para que sea positiva de 0 a 10.

En seguida, a manera de comparación se exponen los resultados obtenidos por el sistema con los diferentes esquemas implementados y las calificaciones realizadas por el profesor de música. En la Tabla 5.5 correspondientes a las melodías monofónicas y en la Tabla 5.6 pertenecientes a las melodías polifónicas.

Tabla 5.5: Comparación de resultados obtenidos por el sistema con los diferentes esquemas implementados y las evaluaciones asignadas por el profesor de música referente a las melodías monofónicas

| **Comparación resultados obtenidos de las Melodías Monofónicas** | | | | | |
|---|---|---|---|---|---|
| No. / Interpretación Estudiantes | Calificación | | | | |
| | Distancia Levenshtein | Distancia InDel | Distancia Inv_Trans_InDel | Distancia TWLCS | Profesor |
| 1.- Frozen 1 | 9.3 | 8.6 | 8.6 | 8.6 | 8 |
| 2.- Frozen 2 | 8.1 | 6.4 | 6.4 | 6.6 | 6 |
| 3.- Frozen 3 | 8.6 | 7.2 | 7.2 | 7.2 | 7 |
| 4.- Frozen 4 | 8.3 | 6.6 | 6.6 | 6.6 | 7 |
| 5.- Frozen 5 | 7.4 | 5.6 | 4.6 | 5.7 | 5 |
| 6.- Frozen 6 | 0 | 0 | 9.2 | 0 | 8 |
| 7.- Frozen 7 | 2 | 0.6 | 9.6 | 0.6 | 9 |
| 8.- Frozen 8 | 8.8 | 8.7 | 6.3 | 8.7 | 7 |
| 9.- Frozen 9 | 8.6 | 8 | 7.2 | 8 | 8 |
| 10.- Frozen 10 | 2 | 0.6 | 9.6 | 0.6 | 9 |
| 11.- Frozen 11 | 6.5 | 5.6 | 5.6 | 5.9 | 5 |
| 12.- Frozen 12 | 6 | 3.1 | 0.9 | 3.2 | 2 |
| 13.- Frozen 13 | 10 | 10 | 10 | 10 | 10 |

Como forma de resumen la Tabla 5.5 y Tabla 5.6, en la columna 4 se plasman las calificaciones arrojados por la distancia invariante a transposición InDel, obteniéndose los mismos resultados calculados de manera directa o utilizando Branch and Bound. Como se ve en las tablas los resultados obtenidos por la distancia invariante a transposición InDel son casi iguales en varios casos a los asignados por el profesor. Ahora bien, con respecto a las interpretaciones de los estudiantes se analiza que, en las interpretaciones donde la ejecución fue realizada con un buen desempeño, pero ocurriendo un desplazamiento de notas sobre el teclado musical. Se puede ver que la distancia invariante a transposición InDel es la que arroja las calificaciones correctas o parecidas a las proporcionadas por el profesor.

Tabla 5.6: Comparación de todos los resultados obtenidos por el sistema con los diferentes esquemas implementados y las evaluaciones asignadas por el profesor de música referente a la melodía polifónica

| **Comparación resultados obtenidos de las Melodías Polifónicas** | | | | | |
|---|---|---|---|---|---|
| No. / Interpretación Estudiante | Calificación | | | | |
| | Distancia Levenshtein | Distancia InDel | Distancia Inv_Trans_InDel | Distancia TWLCS | Profesor |
| 1.- Clock 1 | 9.30 | 8.99 | 8.64 | 8.99 | 8 |
| 2.- Clock 2 | 8.11 | 7.76 | 5.91 | 7.76 | 7 |
| 3.- Clock 3 | 9.25 | 8.99 | 8.55 | 8.99 | 8 |
| 4.- Clock 4 | 9.69 | 9.38 | 9.38 | 9.38 | 9 |
| 5.- Clock 5 | 7.80 | 6.66 | 5.25 | 6.66 | 6 |
| 6.- Clock 6 | 10 | 10 | 10 | 10 | 10 |
| 7.- Clock 7 | 4.59 | 2.34 | 0 | 2.34 | 3 |
| 8.- Clock 8 | 7.58 | 6.92 | 5.07 | 6.92 | 6 |
| 9.- Clock 9 | 8.68 | 8.46 | 6.17 | 8.46 | 7 |
| 10.- Clock 10 | 0 | 0 | 9.74 | 0 | 9 |
| 11.- Clock 11 | 2.34 | 0 | 9.91 | 0 | 10 |
| 12.- Clock 12 | 8.90 | 7.98 | 7.45 | 7.98 | 8 |
| 13.- Clock 13 | 6.04 | 3.80 | 0 | 2.34 | 4 |

Como se ve en la Tabla 5.1 y 5.2, en la columna de observaciones realizadas por el profesor de música, las interpretaciones donde ocurrió un desplazamiento de notas en la colección de melodías monofónicas fueron: *Frozen 6, Frozen 7, Frozen 10.* En la colección de melodías polifónicas: *Clock 10, Clock 11.*

Ahora bien, excluyendo estos casos de interpretaciones donde ocurrió un desplazamiento de notas, se puede visualizar en la Tabla 5.5 y 5.6 los resultados obtenidos por la distancia InDel y TWLCS en interpretaciones donde hubo errores de duración, velocidad, pulsación de notas erróneas se lograron evaluaciones de igual manera muy parecidas sino es que iguales en varios casos, a las asignadas por el profesor. Lo cual indica que estas distancias también logran cumplir el objetivo deseado de la investigación. De hecho proporcionando calificaciones casi iguales entre ellas.

Para finalizar y poder dar una conclusión sólida y creíble hacia las distintas métricas implementados, es conveniente aplicar a las calificaciones obtenidas por el sistema como a las asignadas por el profesor una función *"Suma del Valor Absoluto de Diferencias"*. La suma del valor absoluto de diferencias, función que permite saber que tan relacionados están un conjunto de datos a considerar sobre otro conjunto de datos, teniendo (5.2).

$$\sum_{i=1}^{E} |Y_i - Z_i| \tag{5.2}$$

donde

- $E$ Número de estudiantes.
- $Y$ Dato correspondiente a la calificación obtenida por la técnica de comparación.
- $Z$ Dato correspondiente a la calificación asignada por el profesor de música.

Usando la Ecuación 5.2, se conoce si hay una relación entre el conjunto de datos $Y$ contra el conjunto de datos $Z$. Para ello se puede ver en la Tabla 5.7, los valores calculados por la suma del valor absoluto de diferencias, mostrando resultados para la colección de melodías monofónicas y colección de melodías polifónicas, referente a cada métrica.

En la Tabla 5.7 se muestra que la distancia invariante a transposición InDel obtiene el valor más pequeño de entre los diferentes esquemas implementados tanto para melodías monofónicas de **7.6** como para las polifónicas de **13.544**, lo que significa que las evaluaciones calculas por esta distancia invariante a transposición tiene mayor relación con las

Tabla 5.7: Suma del valor absoluto de diferencias entre calificaciones obtenidas por el sistema y calificaciones asignadas por el profesor de música.

| | **Suma del valor absoluto de diferencias** | | | |
|---|---|---|---|---|
| | Distancia | | | |
| | Inv_Trans_InDel | InDel | TWLCS | Levenshtein |
| Melodía Monofónica | **7.6** | 30.4 | 31.1 | 38.6 |
| Melodía Polifónica | **13.544** | 26.036 | 27.488 | 30.592 |

calificaciones asignadas por el profesor de música. Razón por la cual se puede concluir que esta métrica se apegó favorablemente a los resultados esperados. También cabe mencionar, que los resultados de la distancia InDel y distancia TWLCS son muy parecidos entre ambas distancias, por lo tanto estas dos distancias tienen mucha relación a las evaluaciones obtenidas.

Para finalizar los experimentos y de acuerdo a los resultados obtenidos por la distancia invariante a transposición InDel, se puede decir que dicha distancia funge como mejor calificador para la evaluación automática de interpretaciones musicales basada en información MIDI. A continuación se comparan los tiempos de ejecución de la distancia invariante a transposición InDel calculada de manera directa contra los tiempo obtenidos utilizando metodología Branch and Bound. Como se ve en la Tabla 5.8 se muestran los tiempos en milisegundos (ms.) de la ejecución referente a melodías monofónicas y en la Tabla 5.9 referente a melodías polifónicas, por lo tanto utilizar el algoritmo Branch and Bound reduce tiempos de ejecución al comparar cadenas invariantes a transposición, debido a que permite realizar menos cálculos, buscando una trayectoria óptima.

Tabla 5.8: Tiempos de ejecución en (ms) de distancia Invariante a Transposición InDel (Melodías Monofónicas)

| Interpretación Estudiantes | Implementación Directa | Implementación Branch and Bound |
|---|---|---|
| 1.- Frozen 1 | 112 | 30 |
| 2.- Frozen 2 | 118 | 28 |
| 3.- Frozen 3 | 114 | 45 |
| 4.- Frozen 4 | 111 | 27 |
| 5.- Frozen 5 | 113 | 25 |
| 6.- Frozen 6 | 113 | 27 |
| 7.- Frozen 7 | 118 | 29 |
| 8.- Frozen 8 | 103 | 30 |
| 9.- Frozen 9 | 109 | 34 |
| 10.- Frozen 10 | 110 | 33 |
| 11.- Frozen 11 | 119 | 30 |
| 12.- Frozen 12 | 112 | 34 |
| 13.- Frozen 13 | 107 | 31 |

Tabla 5.9: Tiempos de ejecución en (ms) de distancia Invariante a Transposición InDel (Melodías Polifónicas)

| Interpretación Estudiantes | Implementación Directa | Implementación Branch and Bound |
|---|---|---|
| 1.- Clock 1 | 381 | 55 |
| 2.- Clock 2 | 368 | 59 |
| 3.- Clock 3 | 368 | 59 |
| 4.- Clock 4 | 373 | 66 |
| 5.- Clock 5 | 364 | 55 |
| 6.- Clock 6 | 361 | 60 |
| 7.- Clock 7 | 372 | 68 |
| 8.- Clock 8 | 382 | 57 |
| 9.- Clock 9 | 335 | 57 |
| 10.- Clock 10 | 362 | 61 |
| 11.- Clock 11 | 373 | 77 |
| 12.- Clock 12 | 370 | 63 |
| 13.- Clock 13 | 369 | 67 |

## 5.3. Conclusiones del capítulo

Para la etapa de pruebas, los experimentos se llevaron a cabo con 13 estudiantes de música, en el cual cada alumno interpretó una melodía monofónica y una melodía polifónica. Contando con una colección de 13 interpretaciones de melodías monofónicas y otra colección de 13 interpretaciones de melodías polifónicas a evaluar, con su respectiva referencia base por cada colección de interpretaciones. Una vez que se calcularon las medidas de distancia con las diferentes métricas de comparación entre las cadenas de la referencia base e interpretación del estudiante, fue necesario ponderar esos valores de distancia a una calificación de 0 a 10. Finalmente las calificaciones obtenidas por el sistema se comparan con los asignados por el profesor de música.

También se hizo una comparación entre los diferente esquemas implementados, para ello, se aplicó la función de *Suma del Valor Absoluto de Diferencias* para identificar que técnica de comparación obtuvo los resultados más apegados con relación a las calificaciones asignadas por el profesor. Por lo tanto, concluyendo que la distancia invariante a transposición InDel obtiene el valor más pequeño de entre los diferentes esquemas implementados tanto para melodías monofónicas de **7.6** como para las polifónicas de **13.544**, lo que significa que las evaluaciones obtenidas por esta distancia invariante a transposición tiene mayor relación con las calificaciones asignadas por el profesor de música.

Además se demostró que si la distancia invariante a transposición InDel se realiza mediante la metodología Branch and Bound ejecuta la comparación de interpretaciones más rápido en comparación que si se hiciera de manera directa. La distancia invariante a transposición InDel procesada de manera directa obtuvo en promedio para melodías monofónicas un tiempo de ejecución de *112 ms.* y con la metodología Branch and Bound obtuvo un tiempo de ejecución de *31 ms.*. Tardando Branch and Bound 1/3 parte de lo que se tarda en realizarlo de manera directa. Ahora para melodías polifónicas si se ejecuta de manera directa tiene en promedio un tiempo de ejecución de *367 ms.* y con Branch and Bound de *61 ms.*, lo que significa que Branch and Bound realiza el proceso de comparación en 1/6 parte de lo que se tardaría en ejecutarse de manera directa. El utilizar el algoritmo de Branch and Bound reduce tiempos de ejecución, llegando a una solución óptima en menos cálculos.

Capítulo 6

# Conclusiones y trabajos futuros

## 6.1. Conclusiones generales

La comparación de interpretaciones musicales se puede definir como la tarea de comparar una pieza musical contra otra pieza musical utilizando técnicas de comparación. Estas interpretaciones musicales pueden ser representadas computacionalmente en cadena de caracteres o secuencias de símbolos. Por lo tanto una manera de abordar el problema fue comparar una interpretación de referencia base contra la interpretación del estudiante a ser evaluada. La evaluación de interpretaciones musicales requirió de la extracción de la información contenida en los archivos MIDI. La extracción de la información consistió en convertir el archivo MIDI producido por el estudiante en cadenas de caracteres largas donde la secuencia de notas, su duración e incluso la fuerza fueron concatenadas, haciendo esto mismo con la información del MIDI de la referencia base. Se utilizaron técnicas de comparación de procesamiento de texto las cuales calcularon una medida de distancia de acuerdo al desempeño de la interpretación a ser evaluada, el valor de distancia obtenido fue ponderado a una evaluación de 0 a 10.

Finalmente para poder evaluar el rendimiento del sistema se evaluaron a 13 estudiantes de música para comparar las calificaciones obtenidos con las calificaciones reales asignados por el profesor de música. En la presente investigación la distancia invariante a transposición InDel demostró ser la adecuado para la evaluación de interpretaciones. La

distancia invariante a transposición InDel si se ejecuta de *manera directa* es una distancia computacionalmente costosa, por lo que, el algoritmo de Branch and Bound fue utilizado para reducir el tiempo de ejecución computacionalmente. Las calificaciones de los estudiantes obtenidas por el sistema fueron muy similares a las asignadas por el profesor de música, por lo tanto el sistema de evaluación sería confiable e imparcial al usarse para evaluar interpretaciones musicales de estudiantes de música.

## 6.2. Conclusiones específicas

- De los resultados obtenidos y presentados en el Capítulo 5, se observa que, con el cálculo de *la suma del valor absoluto de diferencias*, los resultados proporcionados por la distancia invariante a transposición InDel utilizando el algoritmo Branch and Bound tienen mayor relación con las calificaciones asignadas por el profesor de música. Esto pudiéndose corroborar con la suma del valor absoluto de diferencias, utilizando el conjunto de calificaciones dadas por la distancia invariante a transposición y el conjunto de calificaciones asignadas por el profesor, se obtuvo que la diferencia entre estos conjuntos de calificaciones en melodías monofónicas es de **7.6** y en melodías polifónicas el valor es de **13.544**, siendo los valores más pequeño de entre los diferentes esquemas implementados. Es decir mientras más se acerque este valor a 0 significa que no hay mucha diferencia entre ambos conjuntos de calificaciones. Razón por la cual se puede concluir que esta técnica proporcionó resultados esperados favorablemente y que el sistema cumple con los objetivos presentados al inicio del documento.

- Los resultados obtenidos por la distancia invariante a transposición son aproximados o iguales a los asignados por el profesor. Además, se analizó que en las melodías interpretadas por los estudiantes, en las cuales la ejecución fue realizada con un buen desempeño, pero ocurrió un desplazamiento de notas sobre el teclado musical se observó que la distancia invariante a transposición InDel arrojó las calificaciones correctas o muy parecidas a las comparadas con las asignadas por el profesor de música. A manera de que, simplemente se identifiquen las interpretaciones donde ocurrió un desplazamiento se mencionan enseguida, en la colección de melodías monofónicas son:

*Frozen 6, Frozen 7, Frozen 10*. En la colección de melodías polifónicas fueron: *Clock 10, Clock 11*.

- Excluyendo las interpretaciones de los estudiantes donde ocurrió un desplazamiento de notas. En los resultados obtenidos por la distancia InDel y distancia TWLCS, en las interpretaciones donde hubo errores de duración, velocidad, pulsación de notas erróneas calcularon evaluaciones muy parecidas o casi iguales en varios casos con relación a las asignadas por el profesor. Lo cual indica que estas distancias también cumplen el objetivo deseado de la investigación. El rendimiento de los resultados de la distancia Indel con relación a las calificaciones asignadas por el profesor de música aplicando el cálculo de la suma del valor absoluto de diferencias para melodías monofónicas fue de *30.4* y en melodías polifónicas de *26.036*. Para la distancia TWLCS en melodías monofónicas fue de *31.1* y polifónicas *27.488*. Ahora, con relación al rendimiento de la distancia de Levenshtein, podría decir que los resultados que proporciona fueron los más alejados a los asignados por el profesor.

- Con relación a los tiempos de ejecución de la distancia invariante a transposición InDel, se vio una grande diferencia entre la utilización de manera directa contra el uso del algoritmo Branch and Bound. En la colección de melodías monofónicas se obtuvo que en promedio se ejecuta el cálculo de manera directa en un tiempo de *112 ms.* y con Branch and Bound de *31 ms.* Para melodías polifónicas de manera directa es de *367 ms.* y con el Branch and Bound de *61 ms.* Por lo tanto el utilizar la metodología de Branch and Bound reduce tiempos de ejecución, debido a que permite realizar menos cálculos, este método es aplicable cuando existen muchos caminos hacia un objetivo y se requiere una trayectoria óptima. Genera el espacio de soluciones, organizándolo en un árbol binario, poda los subárboles inútiles (árboles que no vayan a generar soluciones óptimas).

## 6.3. Trabajos futuros

En el área de comparación de interpretaciones musicales, se quiere desarrollar algoritmos que permitan realizar la comparación de interpretaciones musicales con un mejor desempeño, obteniendo buenos resultados y en especial hacer que dichos algoritmos sean ejecutados en tiempo real, etc. De la presente investigación se enumeran algunos trabajos futuros a mencionar, siendo los siguientes:

1. Una tarea trascendental para la solución del problema de la evaluación automática de interpretaciones musicales es encontrar una técnica robusta que permita saber, si hay mejora de los resultados. Un trabajo interesante puede ser el buscar una combinación adecuada de los esquemas existentes para la comparación de interpretaciones o implementar un método nuevo que pudiera tener mejor desempeño al momento de comparar una pieza musical. Se propone la utilización de los Modelos Ocultos de Markov HMM, para saber si mejoran los resultados con esta técnica.

   Para implementar los Modelos Ocultos de Markov para el problema de la evaluación de una interpretación musical, se tienen los siguientes puntos a realizar.

   - Utilizar la secuencia de observaciones producida por un archivo MIDI, correspondiente a una interpretación considerada como correcta (referencia base).
   - Crear el HMM con esa secuencias de observaciones.
   - Para entrenar ese HMM utilizar el algoritmo Baum-Welch. Entonces se tendría por cada canción un Modelo Oculto de Markov.
   - Ya que está entrenado el HMM, para la etapa de evaluación se toma la secuencia de observaciones de la interpretación del estudiante.
   - Por último, mediante el algoritmo Forward hacer la evaluación.

2. Se propone realizar la tarea de sincronización entre pistas en los archivos MIDI's, en la presente investigación solo se verifican las cadenas de las pistas de la referencia base con las cadenas de las pistas de la interpretación del estudiante, pero no se checa si ocurren al mismo tiempo las notas musicales respecto a la otra pista a comparar.

Cabe mencionar que en la investigación, al momento de hacer la comparación de cadenas se reflejaron los errores ocurridos (errores de notas, duraciones y velocidad) cuando el estudiante interpretaba mal una pieza musical. Pero se desea implementar la sincronización como un trabajo a futura y verificar si resulta posiblemente una mejor evaluación de las interpretaciones musicales.

3. En el estado del arte es posible encontrar trabajos que plantean técnicas para la evaluación de piezas musicales, abordando el problema de dos formas, implementando un sistema en tiempo real o un sistema no en tiempo real. Por lo anterior se pretende hacer las modificaciones correspondientes para que el sistema desarrollado de la evaluación automática de interpretaciones sea realizada en tiempo real. Para ello se necesita conectar directamente el sistema de evaluación automática con el teclado musical MIDI, mediante una interface MIDI.

4. En la música se utilizan una variedad de instrumentos musicales, dichos instrumentos pueden ser fusionados en una misma melodía, es decir teniendo en una sola interpretación musical sonidos de ambos instrumentos (simulando una orquesta musical). Se propone realizar modificaciones al sistema desarrollado para evaluar no solo a un estudiante, sino a varios alumnos tocando otros instrumentos simultáneamente, ejemplo: bajo eléctrico, teclado, guitarra MIDI, etc.

Apéndice A

# Formato MIDI

MIDI, es un estándar de comunicación entre instrumentos musicales digitales y ordenadores. Permite, por ejemplo, la conexión e intercambio de datos entre un teclado y un ordenador, o entre un teclado y un módulo de sonido, o entre dos teclados, de acuerdo a Licencia Creative Commons Atribución en [Unported16].

**Conceptos relacionados con el MIDI**

Hablar de "archivo midi", "teclado midi", "instrumento MIDI", "controlador MIDI", "cable midi", "grabación midi", "pista midi", puede resultar confuso. Por lo tanto se introduce a las principales características y elementos asociados al MIDI.

1. Teclado o Piano MIDI: aquel teclado que soporta MIDI, incluye terminales o conectores MIDI (MIDI IN / OUT / THRU) y por tanto podemos conectarlo (vía MIDI) al ordenador, a un módulo de sonido, o a otros dispositivos MIDI. Actualmente la mayoría de teclados y pianos digitales son MIDI, así como los Silent e híbridos, que mezclan en un solo piano la mecánica acústica con la tecnología digital.

2. Conectores MIDI (IN / OUT / THRU): obviamente los conectores "out" son para salida de datos, y los "in" para entrada. El conector "thru" es también de salida pero envía sólo lo que le llega por el conector "in", es decir, hace que los datos "pasen" a través del dispositivo.

3. Instrumento MIDI: Los instrumentos de teclas como los teclados musicales son los principales y los que más se usan como controladores MIDI, pero no los únicos. Existen también guitarras MIDI, parches de percusión, e incluso instrumentos de viento con capacidad de transmisión vía MIDI. Todos ellos son instrumentos MIDI.

4. Controlador MIDI: es el instrumento con el que generamos los mensajes MIDI (pulsando notas, etc.), normalmente un teclado al que se llama controlador.

5. Cable MIDI: es el cable con el que conectamos los dispositivos MIDI entre sí, o con el ordenador (vía MIDI). El cable MIDI clásico es un cable de 5 pines.

Muchas tarjetas de sonido pro o semiprofesionales vienen con los conectores MIDI que se define arriba. Sin embargo, actualmente muchos teclados MIDI vienen con conector USB, con lo que se pueden conectar al puerto USB del equipo para intercambiar datos MIDI sin necesidad de cable MIDI. Además, existen cables MIDI-USB que hacen de mini-interfaz, con lo que podemos conectar nuestro teclado MIDI con un equipo aunque el teclado no tenga USB y el equipo no disponga de conector MIDI. También se puede utilizar el puerto de juegos (game port) para conectar dispositivos MIDI, con el cable correspondiente, aunque se usa más frecuentemente el puerto USB, más extendido.

Por tanto, hay distintas posibilidades para conectar el teclado al ordenador vía MIDI, no siempre son necesarios los conectores y cables MIDI clásicos, aunque sí suelen ser necesarios para otros usos, como conectar un teclado a un módulo de sonido, o conectar un teclado con otro.

1. Archivo MIDI: es un tipo de archivo con extensión .mid, que contiene datos, no sonido. Lo podemos reproducir y escuchar en prácticamente cualquier equipo porque los sistemas operativos suelen incluir un sintetizador MIDI que “pone sonido” a esos datos MIDI. Windows por ejemplo incluye su sintetizador MIDI de Microsoft.

2. El tipo de datos que contiene un archivo MIDI son principalmente eventos MIDI, que pueden ser notas, controles, cambios de programa, señales de tiempo, etc., pero no hace falta entender esto en profundidad, sólo que contiene datos y no el sonido como tal, al contrario que otros tipos de archivos de música como los MP3. Contiene la

información necesaria para que suenen las notas correspondientes en su lugar, tiempo, intensidad, con el instrumento correspondiente, etc. Te lo puedes imaginar como si fuera una partitura legible para los ordenadores y aparatos MIDI.

3. Grabación MIDI: hace referencia al hecho de enviar y/o guardar datos MIDI, no sonido (lo que sería una grabación de AUDIO). Si grabamos vía MIDI, después podemos asignarle (a lo que hayamos tocado y grabado) el sonido/instrumento que queramos.

4. Pista MIDI: en un secuenciador existen diferentes pistas, para poder superponer los distintos instrumentos y voces de una canción. La pista MIDI es un tipo de pista con un tipo de datos distintos a los de la pista AUDIO. Como siempre, la pista MIDI no contiene sonido, podemos asignarle el que queramos, y tiene unas posibilidades de edición enormes (nota por nota, transporte, velocidad, etc.), diferentes de las posibilidades de edición de AUDIO.

**Características / Elementos principales del MIDI**

1. Eventos MIDI: los eventos MIDI pueden ser notas, controles, cambios de programa (instrumento), y otros tipos de mensajes, es la información que contiene el archivo (si hablamos de un archivo), o que se transmite en tiempo real de un dispositivo a otro (si hablamos de una ejecución).

2. Notas: hasta 128, aunque en la práctica sólo se suelen utilizar las 88 que se corresponden con las 88 teclas del piano, desde el la hasta el do. Cada nota lleva aparejada un valor de "velocidad" (velocity), que se traduce en más volumen y a veces distinto timbre cuanto mayor es ese valor. Aunque se llame velocidad no tiene que ver con el tiempo, sino con la dinámica, la fuerza con la que se ejecuta cada nota. Valores altos = Forte, valores bajos = piano. Y aunque lo parezca, no es absurdo que se le llame velocidad, ya que en la realidad dicha dinámica sí depende de la velocidad con la que se ataca la nota en el piano, o la velocidad con la que se pasa el arco en los instrumentos de cuerda, la presión de aire en los de viento, etc. Este parámetro de velocidad es muy importante porque es el que determina que la ejecución sea musical o no (junto a otras cosas, como el correcto uso del pedal en el piano y otras). Si las

notas tienen idéntica velocidad, sonará monótono, inexpresivo y robótico. Si varía la velocidad sin sentido sonará extraño. Para que suene musical la velocidad debe ir variando de acuerdo con el fraseo de las melodías y del carácter de la obra, tal y como debe hacerlo una buena ejecución humana.

3. 128 instrumentos (programas) estándar en la especificación General MIDI, divididos en grupos. En MIDI, a los instrumentos se les llama programas.

4. 16 Canales MIDI. Permiten el uso de distintos intrumentos a la vez, usando pistas. A cada pista se le asigna un canal, y a cada canal un instrumento. El canal 10 tradicionalmente se reservaba para la percusión, pero en los secuenciadores actuales se puede usar cada canal para lo que se quiera.

5. Controladores o Controles MIDI. También podrían ser hasta 128 dependiendo del dispositivo, pero en la práctica son muchos menos. Aunque vayan numerados del 0 al 127, muchos no tienen ningún controlador asociado, están "vacíos". Algunos de los más habituales o más usados son:

    - 0 : Bank Select / Selección de banco. Los dispositivos como módulos de sonido separan los sonidos en bancos para poder ofrecer más de 128 instrumentos.
    - 1 : Modulation Wheel (rueda de modulación).
    - 7 : Main Volume (volumen principal).
    - 10 : Pan (balance, posición en el estéreo, centro, derecha o izquierda).
    - 11 : Expression (expresión, es también un volumen pero sin afectar al volumen principal).
    - Los controles correspondientes a los pedales del piano: 64 (sustain pedal, pedal derecho), 66 (sostenuto, pedal central), y 67 (soft pedal / corda / pedal izquierdo).
    - 123 : All Notes Off, silencia todas las notas.

# Referencias


[Baeza99] Baeza, R. Y. y Navarro, G. Faster approximate string matching. *Algorithmica*, 23(2):127–158, 1999.

[Baily06] Baily, J. y Collyer, M. Introduction: Music and migration. *Journal of Ethnic and Migration Studies*, 32(2):167–182, 2006.

[Bar-Yossef04] Bar-Yossef, Z., Jayram, T., Krauthgamer, R., y Kumar, R. Approximating edit distance efficiently. *En Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, págs. 550–559. IEEE, October 2004.

[Cano99] Cano, P., Loscos, A., y Bonada, J. Score-performance matching using hmms. *In Proc. of ICMC: International Computer Music Conference*, págs. 441–444, 1999.

[Cont06] Cont, A. Realtime audio to score alignment for polyphonic music instruments using sparse non-negative constraints and hierarchical hmms. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5:245–248, May 2006.

[Cvejic07] Cvejic, N. *Digital Audio Watermarking Techniques and Technologies: Applications and Benchmarks: Applications and Benchmarks*. IGI Global, 2007.

[Dixon05] Dixon, S. Live tracking of musical performances using on-line

time warping. *Proc. of the 8th. Int.conference on Digital Audio Effects*, págs. 1–6, September 20-22 2005.

[García07] García, J. F. Métricas de similitud para búsqueda aproximada. *Journal of Technology*, 6(2):13–25, Julio - Diciembre 2007.

[Grabowski04] Grabowski, S. y Navarro, G. O(mn log $\sigma$) time transposition invariant lcs computation. *Supported by Millenium Nucleus Center for Web Research*, 2004.

[Guo04] Guo, A. y Siegelmann, H. Time-warped longest common subsequence algorithm for music retrieval. *International Conference on Music Information Retrieval*, October 2004.

[Gómez07] Gómez, O. A. C. y Becerra, R. L. Convertidor de archivos con formato wave a archivos con formato midi. 2007.

[Ibarrola10a] Ibarrola, A. C. y Chávez, E. Real time tracking of musical performances. *En Mexican International Conference on Artificial Intelligence*, págs. 138–148. Springer, 2010.

[Ibarrola10b] Ibarrola, J. A. C. Seguimiento en tiempo real de interpretaciones musicales. *Congreso Nacional Ingenieria-Arquitectura*, págs. 201–206, Noviembre 2010.

[Ibarrola15] Ibarrola, A. C. y Morales Pintor, S. Automatic evaluation of music students. *Advances in Artificial Intelligence and Its Applications. Springer International Publishing*, 9414:220–231, December 2015.

[Java SE Documentation16] Java SE Documentation, O. Package javax.sound.midi. https://docs.oracle.com/javase/7/docs/api/javax/sound/midi/package-summary.html, 2016. Consulta: Julio 2016.

[Lemström04] Lemström, K., Navarro, G., y Pinzon, Y. Practical algorithms for transposition-invariant string-matching. *Journal of Technology*, 3(2):267–292, September 2004.

[Manzo11] Manzo, A. y Camarena, J. A. Una nueva y eficiente técnica de alineamiento para evaluar la similitud entre series de tiempo. *Proceedings of the 17th International Congress on Computer Science Research CIICC'11*, págs. 255–266, Octubre 26-28 2011.

[Morales09] Morales, F. D. B. La música en la edad media. *Innovación y Experiencias Educativas*, ISSN 1988-6047 DEP. LEGAL: GR 2922/2007(14):1–10, Enero 2009.

[Murcia06] Murcia, H. C. Buscador musical. *Universidad Carlos III de Madrid, Escuela Politécnica Superior*, Diciembre 2006.

[Navarro01] Navarro, G. A guided tour to approximate string matching. *ACM Computing Surveys (CSUR)*, 33(1):31–88, March 2001.

[Navarro02] Navarro, G. y Raffinot, M. *Flexible Pattern Matching in Strings. (Practical online search algorithms for texts and biological sequences.)*. 2002. ISBN ISBN 0-521-81307-7.

[Orio01a] Orio, N. y Déchelle, F. Score following using spectral analysis and hidden markov models. *In Proc. of ICMC: International Computer Music Conference*, págs. 1–5, 2001.

[Orio01b] Orio, N. y Schwarz, D. Alignment of monophonic and polyphonic music to a score. *Proceedings of the ICMC.ICMA*, págs. 1–4, October 2001.

[Percival07] Percival, G., Wang, Y., y Tzanetakis, G. Effective use of multimedia for computer-assisted musical instrument tutoring.

*En Proceedings of the International Workshop on Educational Multimedia and Multimedia Education*, págs. 67–76, 2007.

[Portillo08] Portillo, M. T. E. y Mendoza, J. A. S. P. P. ch. mahalanobis y las aplicaciones de su distancia estadística. *CULCyT*, (27):13–20, Julio-Agosto 2008.

[Raphael99] Raphael, C. Automatic segmentation of acoustic musical signals using hidden markov models. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 21(4):360–370, April 1999.

[Reverte06] Reverte, A. M. y Hidalgo, J. R. *Conversor WAVE a MIDI en tiempo real para guitarras eléctricas.* 2006.

[Sorensen16] Sorensen, A. y Brown, A. jmusic, music composition en java. http://explodingart.com/jmusic/, 2016. Consulta: Julio 2016.

[Turetsky03] Turetsky, R. J. y Ellis, D. P. Ground-truth transcriptions of real music from force-aligned midi syntheses. *In Proceedings of the Fourth International Conference on Music Information Retrieval*, págs. 1–7, October 2003.

[Unported16] Unported, L. C. C. A. C. Introducción al midi. http://www.creandopartituras.com/introduccion-al-midi/, 2016. Consulta: Julio 2016.

[Valenzuela96] Valenzuela, J. C. *Audio Digital, Conceptos Básicos y Aplicaciones.* Miller Freeman Books, 1996. ISBN ISBN 0-87930-430-8.

[Yang02] Yang, C. Efficient acoustic index for music retrieval with various degrees of similarity. *En Proceedings of the tenth ACM international conference on Multimedia*, págs. 584–591. ACM, 2002.

# Glosario de Términos

**DTW** *Dynamic Time Warping* - Doblado Dinámico en Tiempo.

**HMM** *Hidden Markov Model* - Modelo Oculto de Markov.

**LCS** *Longest Common Subsequence* - Subsecuencia Comun Mas Larga.

**LCTS** *Length of the Longest Common Transposition Invariant Subsequence* - Longitud de la Subsecuencia Común Más Larga Invariante a Transposición.

**MICAI** *Mexican International Conference on Artificial Intelligence.*

**MIDI** *Musical Instrument Digital Interface* - Interconexión Digital de Instrumentos Musicales.

**MIR** *Music information retrieval* - Recuperación de Información Musical.

**MP3** *MPEG-1 Audio Layer 3* - MPEG-1 capa de audio 3.

**NFA** *Non-deterministic Finite Automaton* - Autómata Finito No-determinista.

**NMF** *Non-negative Matrix Factorization.*

**PITCH** - Valor de tono correspondiente a una nota del teclado musical.

**TWLCS** *Time Warped Longest Common Subsequence.*

**WAV** *Waveform Audio File Format*