Universidad de Extremadura
Departamento de Tecnología de los
Computadores y de las Comunicaciones

Doctor of Philosophy Dissertation

Una Aproximación Genética a la Transcripción Automática de Música

A Genetic Algorithm Approach to Automatic Music Transcription

by

Gustavo Miguel Jorge dos Reis

Spain 2012

Thesis for the Degree of Doctor of Philosophy
This Thesis was submitted to the University of Extremadura
in accordance with the criteria necessary for the award
of the Doctorate Degree with European Mention.

Organization:

Departamento de Tecnología de los Computadores y de las Comunicaciones; Universidad de Extremadura; Mérida; España

Title:

Una Aproximación Genética a la Transcripción Automática de Música
A Genetic Algorithm Approach to Automatic Music Transcription

Author:

Gustavo Miguel Jorge dos Reis

Supervisors:

Francisco Fernández de Vega (Universidad de Extremadura, España)
Aníbal João de Sousa Ferreira (Universidade do Porto, Portugal)

Dr. D. Francisco Fernández de Vega, profesor titular de la Universidad de Extremadura, España

CERTIFICA:

que la presente memoria, titulada "Una Aproximación Genética a la Transcripción Automática de Música" *("A Genetic Algorithm Approach to Automatic Music Transcription")* ha sido realizada por D. Gustavo Miguel Jorge dos Reis bajo mi dirección en el Departamento de Tecnología de los Computadores y de las Comunicaciones de la Universidad de Extremadura.

Y para que conste, y en cumplimiento de la legislación vigente, firmo la presente.

Dr. D. Francisco Fernández de Vega

Dr. D. Aníbal João de Sousa Ferreira, profesor titular de la Universidade de Porto, Portugal

CERTIFICA:

que la presente memoria, titulada "Una Aproximación Genética a la Transcripción Automática de Música" *("A Genetic Algorithm Approach to Automatic Music Transcription")* ha sido realizada por D. Gustavo Miguel Jorge dos Reis bajo mi dirección en el Departamento de Tecnología de los Computadores y de las Comunicaciones de la Universidad de Extremadura.

Y para que conste, y en cumplimiento de la legislación vigente, firmo la presente.

Dr. D. Aníbal João de Sousa Ferreira

To Helena

Contents

Acknowledgements

First of all, I would like to thank to my supervisors, Prof. Francisco Fernandéz de Vega and Prof. Aníbal Ferreira, for their support, attention, guidance, insight, constructive comments, suggestions and encouragement through these years.

I would also like to thank to Nuno Fonseca for pushing me during my first steps on the Signal Processing field and, specially, on the first implementation of the piano synthesizer and collaborative research. I would also like to thank Nuno for introducing me to Aníbal Ferreira, who became my supervisor.

I am also deeply grateful to Patrício Domingues, for reducing my work ours, so I could focus on the writing of this dissertation.

A special thanks to Valentin Emiya for sharing his results, and to all the researchers that shared their algorithms so that we could perform the reported comparisons.

To Daniel Lombraña Gonzeléz and Paco Chavez for their hard work on configuring the blade machines so that we could perform all the tests.

To the Spanish Ministry of Science and Innovation for the support under project ANYSELF (TIN2011-28627-C04), Gobierno de Extremadura, under projects GRU09105, GR10029 and Municipality of Almendralejo.

I cannot end without thanking my daughters Maria and Sofia for their patience and understanding. Lastly, and most importantly, I wish to deeply thank my wife Helena for her constant encouragement, love and most of all, patience, throughout this journey. To them I dedicate this dissertation.

Resumen

La transcripción de música es un proceso que pretende extraer una notación musical legible por las personas, tal como una partitura, a partir de una señal acústica. De este modo, la transcripción automática de música es el proceso en el que un computador extrae la notación deseada de la fuente audio. La transcipción automática de música es un área de investigación que además de encuadrarse en ciencias de la computación, toca varias disciplinas, incluyendo procesamiento digital de señales, aprendizaje máquina, psicoacústica, percepción de tono, teoría musical y también conocimiento musical y teoría cognitiva.

La transcripción automática de música es una tarea extremadamente difícil, que ha sido ya abordada en varias tesis doctorales. A pesar del número de intentos de resolver el problema, un sistema de propósito general para transcripción automática, práctico y aplicable no ha sido desarrollado hasta la fecha. Más aún, los sistemas disponibles en la actualidad no alcanzan el nivel de eficacia y flexibilidad desarrollado por músicos profesionales.

Entendemos aquí el problema de la transcripción automática de música como un problema de optimización combinatoria, cuyo objetivo es encontrar la combinación de notas musicales que mejor representa la señal acústica observada. Nuestra propuesta hace uso de codificación dispersa y algoritmos genéticos, que son una muy buena herramienta en para problemas de búsqueda. Utilizando esta aproximación, nuestro método puede trabajar con sonidos que incluyen varias componentes armónicas.

Esta tesis presenta las etapas de investigación que se han desarrollado con el fin de aplicar los Algoritmos Genéticos al problema de la Transcripción Automática de Música. Hemos utilizado varias aproximación basadas en los algoritmos genéticos para resolver el problema de estimación múltiple de frecuencias (tonos), comenzando primero con un modelo simplificado (sintetizado) de instrumentos musicales, para después pasar a trabajar con grabaciones realizadas con instrumentos musicales reales. En cada experimento hemos aplicado diferentes modelos y herramientas teóricas (espectros logarítmicos y lineales, filtros, cepstrum, cepstral híbrido y análisis espectral, autocorrelación, etc.) para buscar medidas de similaridad y de error (pasando por distancias Hamming y distancias de áreas de intersección Itakura-Saito, correlación y otras variantes).

Hemos trabajado también en el problema de "Sobreentrenamiento armónico" que tiene su origen en las diferencias de timbre, y hemos propuesto un modelo de envoltura espectral

para resolverlo. Además hemos empleado esta aproximación en señales musicales que utilizan diferentes instrumentos, para mostrar la capacidad para trabajar en problema multi-timbre.

También presentamos un nuevo modelo para estimación múltiple de frecuencias fundamentales en grabaciones de piano. Proponemos un esquema basado en los algoritmos genéticos para analizar los solapamientos de los armónicos durante la búsqueda de la combinación más correcta de frecuencias fundamentales. El proceso de búsqueda es ayudado por un proceso de modelado adaptativo de la forma del espectro y de una estimación dinámica del nivel de ruido en las muestras de piano, para que coincida con el piano real presente en la señal de entrada, ayudando así al proceso de búsqueda. Hemos comparado nuestros resultados con los obtenidos por varios algoritmos del "estado del arte" utilizando varias piezas musicales ejecutadas sobre diferentes pianos. El nuevo algoritmo propuesto obtiene un meritorio primer y segundo lugar en las comparativas, dependiendo de la medida de comparación utilizada. También se comparó la nueva propuesta con una aproximación anterior basada en algoritmos genéticos y se observa las significativas mejoras aportadas tanto en calidad como en tiempo de cómputo.

Esta tesis doctoral también presenta contribuciones útiles de modo genérico para la Computación Evolutiva. En concreto, la técnica que denominamos "Competición de Fragmentos Genéticos (Gene Fragment Competition, GFC)" puede ser aplicada en problemas descomponibles tanto en procesamiento de señales audio como imágenes. Se realizó un estudio completo para mostrar la utilidad de la técnica en problemas descomponibles. Haciendo uso del modelo modular y jerárcico de las funciones tipo "Royal Road" se han hecho tests que muestran como la nueva técnica puede superar problemas de correlacion. Mostramos empíricamente que GFC en general sobrepasan al algoritmo genético estandar, al modelo coevolutivo y también al método de ascenso de colina. La aplicación de descomposición de problemas en bloques es una técnica útil para evitar los problemas de correlación mencionados. A pesar del hecho que la mutación aleatoria asociada a métodos de ascenso de colina han probado en el pasado ser ideales para funciones tipo "Royal Road" hemos mostrado que el nuevo método presentado puede explorar más eficientemente el espacio de búsqueda en estas funciones.

Palabras clave: Transcripción Automática de Música, estimación múltiple de frecuencias fundamentales, estimación muti-tono, análisis de señales acústicas, recuperación de información musical, perecpción de tono, algorítmos genéticos, competición de fragmentos genéticos.

Abstract

Music transcription is the process of extracting human readable notation, like a music score, from an acoustical signal. This way, automatic transcription of music is the process in which a computer program extracts notation from an audio signal. Automatic transcription of music is a research area that, besides computer science, encompasses several disciplines including digital signal processing, machine learning, psychoacoustics and pitch perception, music theory and also music cognition.

Automatic transcription of music is an extremely difficult task, which has already been addressed in several doctoral theses. Despite these number of attempts to solve the problem, a practical and applicable, general-purpose transcription system still does not exist at the present time. Furthermore, current available systems fall behind skilled human musicians in both accuracy and flexibility.

We depict the problem of automatic music transcription as a combinatorial optimization problem where the goal is to find the combination of musical notes that best represents the observed signal. We extend the sparse coding with genetic algorithms, which are a very good tool on search problems. By using sparse approximation, along with evolutionary algorithms, our method is able to cope with harmonic sounds with varying harmonic components.

This dissertation presents the several steps of our research on addressing Genetic Algorithms to the problem of Automatic Transcription of Music. We have employed several genetic algorithm approaches to address the problem of multi-pitch estimation, first starting with simple synthesized models of instruments, and then, moving to real audio recordings, performing several experiments. These experiments included different domains and tools (log spectra, linear spectra, filter banks, real cepstrum, hybrid cepstral and spectral analysis, auto correlation and summary auto correlation functions, etc.) for audio similarity measurement and several error measurements (from Hamming and Itakura-Saito distances to area intersection, correlation and other variations). We faced the problem of Harmonic Overfitting, which is related to timbre differences, and proposed a spectral envelope modeling technique to address this issue. Furthermore, we have also employed this approach on musical signals with different audio instruments to show the feasibility of the approach on multi-timbral music.

We present a new method for multiple fundamental frequency (F0) estimation on piano recordings. We propose a framework based on a genetic algorithm in order to analyze

the overlapping overtones and search for the most likely F0 combination. The search process is aided by adaptive spectral envelope modeling and dynamic noise level estimation: while the noise is dynamically estimated, the spectral envelope of previously recorded piano samples (internal database) is adapted to best match the piano played on the input signals and aid the search process for the most likely combination of F0s. For comparison, several state-of-the-art algorithms were tested on various musical pieces played by different pianos and then compared using three different metrics. The proposed algorithm ranked second place on both Onset Only and Onset-Offset metrics and ranked first place on Hybrid Decay/Sustain Score metric, which has better correlation with the human hearing perception. One final comparison is made with a previous genetic algorithm approach to show how the proposed system brings significant improvements on both quality of the results and computing time.

This dissertation also presents our contributions to the field of Evolutionary Computation, namely the Gene Fragment Competition approach, which can be used on most decomposable problems in signal or image processing. An analysis of how decomposable approaches are suitable to decomposable problems is presented. We took advantage of the modular and hierarchical structure of the Royal Road functions to use them as test functions and show how single-population decomposable approaches, such as the Gene Fragment Competition, can overcome the spurious correlation or hitchhiking. We show empirically that both Parisian approach and Gene Fragment Competition show, in general, better behavior than not only the standard genetic algorithm and the multiple-population co-evolutionary approach but also the random mutation hill-climber. Hitchhiking is known to be, in general, one of the major bottlenecks of the genetic algorithms performance. Therefore, avoiding hitchhiking has the potential to boost the performance of the algorithm. Applying problem decomposition in building blocks is an advantageous optimization technique, since this avoids the hitchhiking phenomena. Despite the fact that the random mutation hill-climber algorithm has proved in the past to be the ideal for the Royal Road functions, we have shown that single population decomposable approaches can explore more efficiently the search space on Royal Road functions.

We show empirically that both Parisian approach and Gene Fragment Competition show, in general, better behavior than not only the standard genetic algorithm and the multiple-population co-evolutionary approach but also the random mutation hill-climber.

Keywords: Automatic music transcription, multiple fundamental frequency estimation, multi-pitch estimation, acoustic signal analysis, music information retrieval, pitch perception, genetic algorithms, gene fragment competition.

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The question:

> *How can we automatically transcribe the musical score of an acoustic song or audio signal?*

is known as the *automatic music transcription problem*.

Music transcription is the process of extracting human readable notation (e.g. a music score) from an acoustical signal. This way, *automatic transcription of music* is the process in which a computer program extracts musical notation from an audio signal when it is represented using a non-semantic format such as PCM. Automatic transcription of music is a research area that, besides computer science, transverses several disciplines including digital signal processing, machine learning, psychoacoustics and pitch perception, music theory and also music cognition. Also, it is a music information retrieval (MIR) task. Other MIR tasks include:

- Audio Genre Classification;
- Audio Music Mood Classification;
- Audio Classical Composer Identification;
- Audio Cover Song Identification;
- Audio Tag Classification;
- Audio Music Similarity and Retrieval;
- Symbolic Melodic Similarity;
- Audio Onset Detection;
- Audio Key Detection;
- Real-time Audio to Score Alignment (a.k.a Score Following);

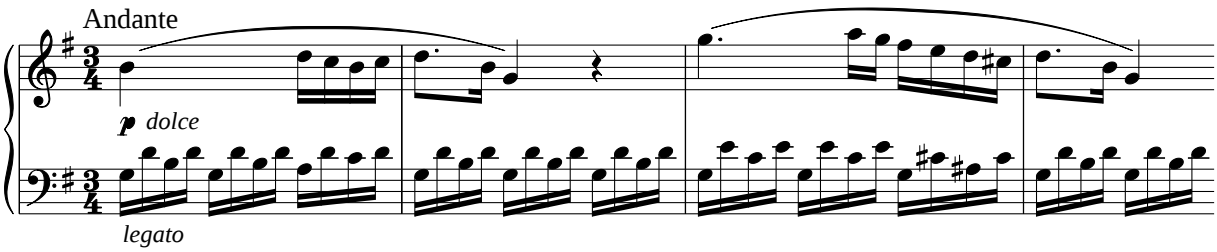

Figure 1.1: Music score of the second movement of Piano Sonata No. 16 in C major, K. 545, by Wolfgang Amadeus Mozart.

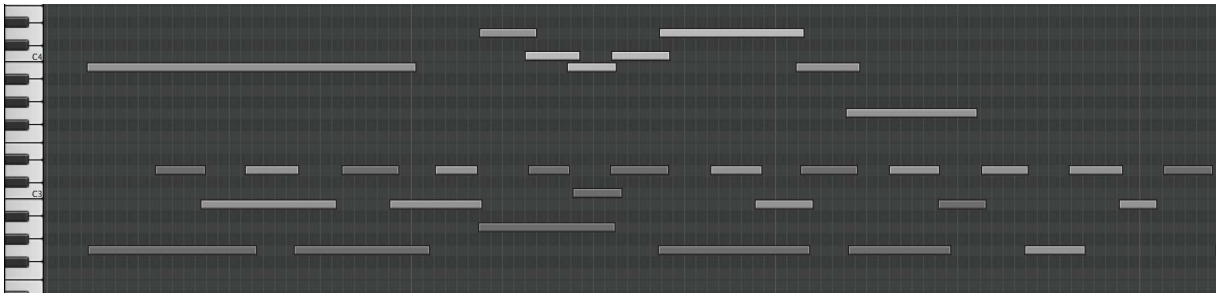

Figure 1.2: Piano-roll of the second movement of Piano Sonata No. 16 in C major, K. 545, by Wolfgang Amadeus Mozart.

- Query by Singing/Humming;
- Audio Melody Extraction;
- Multiple F0 Estimation & Tracking;
- Audio Chord Estimation;
- Query by Tapping;
- Audio Beat Tracking;
- Structural Segmentation;
- Audio Tempo Estimation.

The main goal of music transcription systems is helping a musician to write down the music notation of a performance from an audio recording. This is a time consuming task when done by hand and very hard problem as well, that only the most skilled musicians can address. Automatic transcription systems can also be used as input in other MIR systems like, for instance, plagiarism detection, artist identification, genre classification and also music composition assistance.

As previously mentioned, the goal of automatic music transcription systems is to extract human readable representation, like music notation, from an acoustic signal. In this context, a score is a guide to perform a piece of music, and it can be represented in several and different ways. The most used representation is the modern notation used in Western tonal music: the **score** (see Figure 1.1).

To extract a score from an acoustic signal, several problems must be also addressed. Besides the estimation of the pitches of each source, it is also necessary to infer onset[1] times and durations of the notes, the tempo. Depending on the process, the meter and the tonality of a musical piece might also need to be inferred. In general, the transcription process is separated into two main steps: the extraction of a **piano roll** representation (see Figure 1.2) from an acoustic signal, and the subsequent conversion of the inferred piano roll to the score.

In general, automatic music transcription is only addressed as the stage of audio to piano roll conversion, whereas the further conversion of the piano roll to the score is considered to be a separate problem (Cemgil et al., 2006b). Moreover, a music transcription system cannot obtain the exact score that the musician reads while performing. This happens because music is a way to express feelings and emotions: a particular score can be performed by the same musician in several different ways, according to his mood. This way, musical sheets are nothing but guidelines for the performers to play a musical piece, rather than simple mechanical translations of notes read on a score. Furthermore, the process involved in audio to piano roll representation, besides pitch estimation, also requires the temporal segmentation of musical notes, which is a challenging task on its own. On the other hand, piano roll to score conversion involves more high-level tasks like, for instance, tempo estimation, rhythm quantization, key detection or pitch spelling. This step is more related to the generation of human readable notation. This is why, in general, music transcription methods generate only the piano roll as output. The generated piano roll can later be used as input for symbolic music algorithms that will generate the corresponding musical notation.

If we discard all the high-level features, like the metrical structure or tempo information, the conversion from an acoustic signal into a piano roll representation only depends on the waveform. This way, a piano roll can be seen as a note oriented representation which shows the musical notes that are playing at each time. The conversion from an acoustic signal into a piano roll representation is done by a **multiple-fundamental frequency (F0) estimation** method, which is the core of a music transcription system: it estimates the number of sources sounding at each time and their corresponding pitches.

The conversion of a piano roll into a music sheet, besides multiple-F0 estimation, must also address several harmonic and rhythmic features and other tasks, such as: tonality, source separation and timbre classification, metrical structure and tempo. Tonality is the tonic character of a piece of music determined by the "center" key, on which specific hierarchical pitch relationships are based. Source separation and timbre classification allow the segmentation of the music according to each different instrument identified in the music signal, allowing the generation of the respective scores (one score per different instrument). Metrical structure is the pattern of the beats in a music piece and includes several rhythmic aspects like: meter and tempo. It also specifies the time signature, i.e. how many beats are in each measure and what note value constitutes one beat. This way, bars can be added to the score. Tempo is a measure to describe how fast or slow a musical piece is.

[1]Onset refers to the time where the musical note starts.

In comparison to the conversion of a piano roll into a music sheet, converting a performance played by a musician into a score is even harder to address. When the performer plays, there are slight temporal deviations (fractions of a second) between the notes on the original score and the notes in the resulting acoustic signal. This stresses the need of note onsets and durations to be adjusted (quantized) to obtain a readable score, which means that performance will most certainly not match exactly the original timing information existing on the score. This way, piano roll is considered as the note oriented representation of the acoustic signal.

1.1 Objectives and Scope of the Thesis

The goal of this dissertation relies on the conversion of musical performances into their note oriented representation i.e. piano roll by means of multiple-F0 estimation. Multiple-F0 estimation is an extremely difficult task, which has already been addressed in several doctoral theses, such as: Moorer (1975); Maher (1989); Marolt (2002); Hainsworth (2003); Bello (2003); Cemgil (2004); Vincent (2004); Klapuri (2004a); Zhou (2006); Yeh (2008); Ryynänen (2008); Emiya (2008) and Pertusa (2010). Despite these number of attempts to solve the problem, a practical and applicable, general-purpose transcription system does not exist at the present time. Moreover, current available systems fall behind skilled human musicians in both accuracy and flexibility.

Music transcription is a very difficult problem, not only from the computational point of view, but also in a musical perspective since it can only be addressed by the most skilled musicians. Usually, only pitched musical instruments are considered. Recognizing drum instruments or the sounds of the singer is not discussed in this dissertation.

1.2 Thesis Contributions

The main contributions contained within this dissertation are summarized below:

- A new method of automatic transcription of polyphonic piano music using Genetic Algorithms, Adaptive Spectral Envelope Modeling and Dynamic Noise Level Estimation (presented in Chapter 9).

- New developments on using evolutionary algorithms, namely Genetic Algorithms, for automatic music transcription (presented in Chapters 6 and 7).

- A new evolutionary approach, the Gene Fragment Competition, which improves the performance of evolutionary algorithms in sound processing applications (presented in Chapter 8).

- Some developments on the performance optimization as a result of applying decomposable approaches, namely the Gene Fragment Competition, to decomposable problems by exploiting their modular and hierarchical structure (presented in Section 8.4).

1.3 Outline of the Thesis

This dissertation is organized as follows.

Chapter 2

This chapter starts presenting a brief explanation of several background topics, from waves and signal sampling to more advanced digital signal processing. Throughout this chapter, relevant terminology is defined. Finally, a theoretical background addressing the frequency estimation of acoustic signals is presented.

Chapter 3

In this chapter a literature review of previous studies on multiple-F0 estimation is presented. These methods are categorized according to their complexity and also according to their scope. The reviewed studies are also discussed.

Chapter 4

This chapter presents the concept of Algorithm and its main purpose: problem solving. Several types of problems are presented (decision, search, optimization and counting) and it is discussed how can all the problems be viewed as decision problems. Then, according to their computational complexity, four classes of decision problems are presented: P, NP, NP-Complete and NP-Hard problems. Bio-inspired Algorithms are introduced as a mean of addressing NP-Complete and NP-Hard problems and, finally, Genetic Algorithms are presented. Both biological background behind a Genetic Algorithm and its terminology are explained.

Chapter 5

In this chapter, the problems of Automatic Transcription of Music and Multi-Pitch Estimation are presented from the search space or optimization points of view. Both problems are presented as an optimization problem, and an idea about the size of the search space is given. The computational complexity of the problem is studied and the NP-Completeness of the problem is discussed. Then, it is discussed how to address combinatorial optimization problems with Genetic Algorithms and, finally, a discussion is presented on how to address the problem of Automatic Transcription of Music with Genetic Algorithms.

Chapter 6

This chapter describes the first genetic algorithmic approaches to the problem, namely addressing synthesized audio signals and simple mathematical models. It is also discussed how each approach found in the literature addresses each of these topics: genotype, fitness evaluation, selection, recombination, mutation, creation of the initial population and survivor selection. The problems of each method are also discussed.

Chapter 7

In this chapter we describe our first Genetic Algorithm approaches to the problem of Automatic Transcription of Music on real audio recordings. Considering that polyphonic real audio recordings have different spectral envelopes for different sources and inharmonic partials, spectral envelope modeling is also introduced.

Chapter 8

This chapter presents our contributions to the field of Evolutionary Computation, namely the Gene Fragment Competition, which can be used on evolutionary algorithm for signal and image processing. Our proposed approach sits as a trade-off between classical Genetic Algorithms and traditional Memetic Algorithms, performing a quasi-global/quasi-local search by means of gene fragment evaluation and selection. The applicability of this hybrid Genetic Algorithm to the signal processing problem of Polyphonic Music Transcription is shown. The results obtained show the feasibility of the approach. We also present an analysis of how decomposable approaches are suitable to decomposable problems. Moreover, we have taken advantage of the modular and hierarchical structure of the Royal Road functions in order to use them as test functions and see how single-population decomposable approaches can overcome the spurious correlation or hitchhiking.

Chapter 9

This chapter presents our main contribution: a new method for multiple fundamental frequency (F0) estimation on piano recordings. We propose a framework based on a genetic algorithm in order to analyze the overlapping overtones and search for the most likely F0 combination. The search process is aided by adaptive spectral envelope modeling and dynamic noise level estimation. For comparison, several state-of-the-art algorithms were run across various musical pieces played by different pianos and then compared using three different metrics. The proposed algorithm ranked first place on Hybrid Decay/Sustain Score metric, which has better correlation with the human hearing perception and ranked second place on both Onset-only and Onset-Offset metrics. A previous genetic algorithm approach is also included in the comparison to show how the proposed system brings significant improvements on both quality of the results and computing time.

Chapter 10

In this chapter, we present the results of our methods on the Music Information Retrieval eXchange (MIREX). The method presented in Chapter 9 was also submitted to the Humies competition and was selected by the SIGEVO executive board as one of the 10 humies finalists competing for a $5.000 dolar prize.

Chapter 11

Finally, this chapter presents our main conclusions. Possible lines of research about the evolution of the harmonic model and about the Gene Fragment Competition are also

presented as future work.

Chapter 2

Sound, Signals and Fundamental Frequency Estimation

This chapter presents a brief explanation of several background topics, from sound waves and signal sampling to more advanced digital signal processing (DSP). This chapter is aimed at readers that are not so familiarized with digital signals and systems. Throughout this chapter, relevant terminology is defined and theoretical background towards the frequency estimation of acoustic signals is presented.

2.1 Audio: Sound Waves

Everything what we call sound are vibrations in a medium, like air. The string of a guitar or even our vocal chords move and create pressure oscillations in the air. One vibration or cycle occurs when the air is compressed, rarefacted and returns to its original state. The human ear can hear vibrations occurring between 20 and 20 000 times per second. The physician Hertz gave his name to the measure of *"number of cycles per second"*. Thus we can hear, from low to high, vibrations from 20 Hz to 20 kHz, approximately.

The graphic with the sound waves or the oscillogram, in many recording computer programs, shows both compression and decompression of air as a function of time. In the vertical axis, the variation of the sound pressure indicates the sound volume. The horizontal axis shows the time evolution of the sound wave.

2.1.1 Analog Audio

Microphones have a membrane that moves according to the air vibrations and a circuit to generate an electrical signal. The voltage of that electrical signal changes according to the vibrations of the membrane. Therefore, all the sound vibrations that are sensed by the microphone's membrane are transformed into voltage oscillations, which are transmitted to an amplifier. When the electric audio signal arrives into a speaker, the speaker acts on the opposite way of the microphone: its cone vibrates, according to the electric variations,

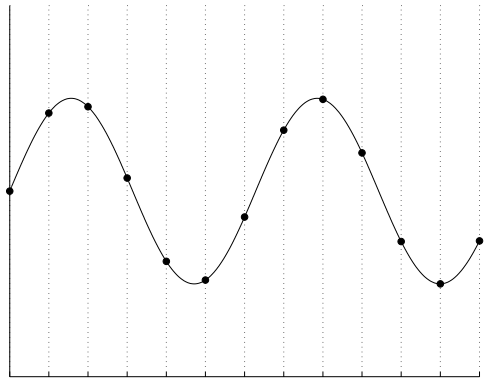

Figure 2.1: The ADC collects periodic samples of the sound pressure.

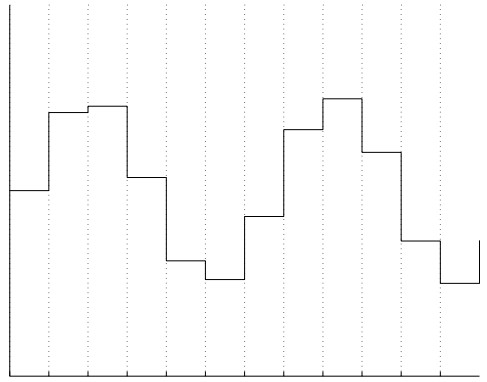

Figure 2.2: Digitized sound wave.

making the air particles to move, creating again the mechanical sound. Between the microphone's membrane and the speaker's cone we have the electrical audio signal or analog sound. It is called analog because it is a continuous function of time and thus denotes an *analogy* between the air vibrations and voltage oscillations.

2.1.2 Digital Audio

When an analog sound is sent to a computer sound card or to a digital audio mixer, the electrical signal must be digitized, i.e. converted to numeric values. The input of the device where the cable is connected to has an analog/digital converter (ADC) to convert voltage values into numbers. The digital/analog converter (DAC) does exactly the opposite: it recreates the analog sound after being processed by the computer making it possible for a loudspeaker to reproduce the sound.

The ADC converter converts the sound vibrations into numbers by a process of sampling (see Figure 2.1). This process happens thousands of times per second: the current state of the oscillation is converted into a digital number. When that sequence of digital

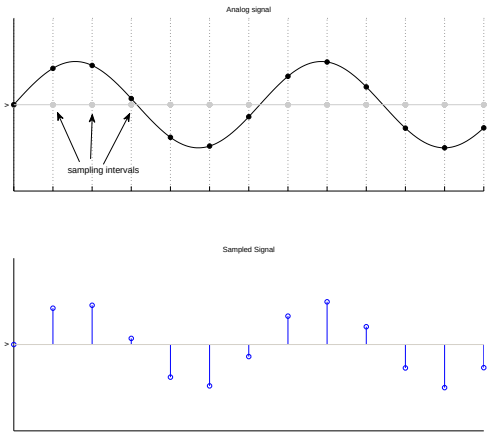

Figure 2.3: Sampled signal.

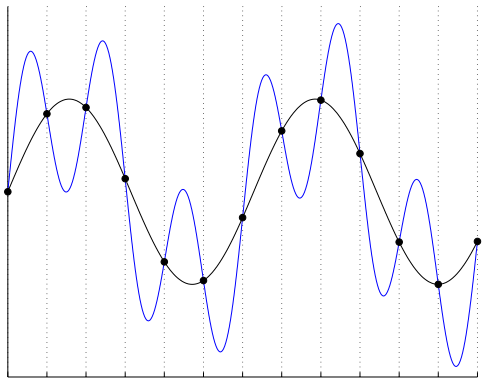

Figure 2.4: A complex wave with lost frequencies due to a low sample rate.

numbers is represented graphically it has the wave shape similar to the original analog sound. But, if we look closer, we will see that waves of the oscillation vary in a "stair" shape (see Figure 2.2). In fact, the digital sound is not continuous in time or amplitude (it is discrete). If we increase the number of samples per second (i.e. the sample rate) a higher density of samples results making the digitized signal more similar to the original sound.

2.1.3 Signal Sampling

Signal sampling is the acquisition of a continuous signal (for instance: an analog sound) in discrete time intervals. The output of the sampler varies only in periodic intervals of time, when it assumes the instant value of the input signal (see Figure 2.3). Any variation that might occur between the sampling intervals is completely ignored (see Figure 2.4). Sampling is the key concept for the real-time digital signal processing.

Sampling Theorem

The sampling theorem, also known as the Nyquist theorem, specifies the sampling rate at which an analog signal should be sampled so that all relevant information in the signal is preserved in the sampling process. This theorem states that: "If a low pass-band signal has the highest frequency equal to F_{max}, then for its exact reconstruction it must be sampled at least at F_s sample rate, where F_s is equal to or greater than the double of F_{max}". Equation 2.1 shows this theorem.

$$F_s \geq 2F_{max} \tag{2.1}$$

Therefore, so that a sound preserves frequencies between 20 Hz and 20 kHz, the sampling rate should be equal or greater than 40 kHz. This is why industry adopted 44.1 kHz (44100 samples per second) for the CD since it covers all the audible spectrum.

Nowadays, the technology operates with higher frequency rates to achieve the best possible fidelity to real sound.

Sampling Depth

Each digital sample is a point with the wave or signal value on that instant. From time to time (for instance, in a CD, at each $\frac{1}{44100}$ of second or 0.000023 seconds) the analog/digital converter expresses the wave amplitude with a numeric value. Consecutive values denote the wave variations which typically are represented as vertical variations in a wave graph where the abscissa axis denotes time. The number of possible values (or levels) indicating the amplitude of each wave sample is expressed in 'bits' - sequences of 0's and 1's (binary system).

A sound sample with 8 bits only has 256 possible levels, but the dynamics of the human ear is much more sophisticated since, for example, it can distinguish between the flapping wings of a fly to an airplane's turbine, which is millions of times stronger. With 16 bits - the format used in audio CDs - we have 65,536 possible levels, which implies a substantial gain in quality compared to the 8 bits resolution. The 24 bits used on the DVD give 16,777,216 amplitude variations. Figure 2.5 shows the digitalization of a sound wave with 2 bit depth.

2.2 Music

Music is the art of expressing feelings, values and ideas through a sequences of sounds and silences. One sound is essentially characterized by four fundamental properties: *dynamics*, *duration*, *timbre* and *pitch*.

Dynamics: volume or sound pressure refers to the perception of the amplitude of the sound wave, it indicates how "loud" or how "soft" a sound is. It is often referred to as note velocity due to the MIDI terminology Association (2008).

Duration: a sound produced by a musical instrument has a duration - we can hear it during a certain period of time. In western music notation several symbols were

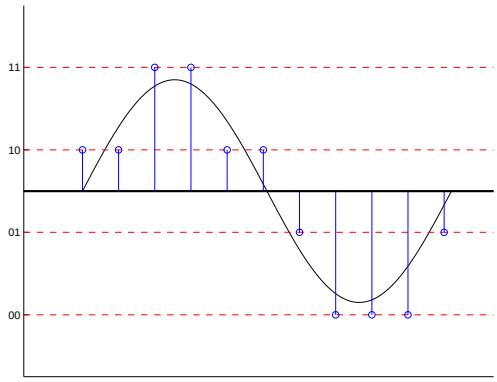

Figure 2.5: A bit depth of 2 (which makes a sound just barely intelligible) does not provide a very accurate representation of the original sound.

created to represent units or fractions of time - *tempo*. The *tempo* is related with the beat time (number of beats per second - bpm) or the music's rhythm.

Timbre: is mainly defined by the spectral envelope of a sound. In time-domain, it is related to the waveform (e.g. square wave, sawtooth wave, etc.), which corresponds to a specific spectral organization, i.e. a harmonic series in the frequency-domain. It is by the timbre that human listeners recognize which instrument plays a specific musical note.

Pitch: is the tonal height of a sound. It indicates how "high" or "low" a note sounds. The physical correlate of pitch is the **fundamental frequency** - F0. Although the F0 may be obtained throughout physical measurement, this can be different from the perceived pitch due to the harmonics and overall loudness of the sound.

In general, pitched musical instruments are based on a harmonic oscillator, which in turn is a system that, when displaced from its equilibrium position, applies a restoring force proportional to the displacement. When a string, for instance, is plucked, it experiences a restoring force proportional to the displacement from its equilibrium position, causing the string to oscillate. Harmonic oscillators like strings or columns of air can oscillate at numerous frequencies. These resonant frequencies, by traveling in both directions along the string or the columns of air, are self-filtered by reinforcing and also canceling each other to form standing waves. This way, due to the typical spacing of the resonances, the resulting frequencies are integer multiples (or harmonics) of the lowest (fundamental) frequency. These multiples, along with the fundamental, constitute the **harmonic series**.

The fundamental frequency (F0) is the lowest frequency in a harmonic series and is defined for periodic or nearly periodic sounds only. For these classes of sounds, F0 is defined as the inverse of the period. In ambiguous situations, the period corresponding to the perceived pitch is chosen (Klapuri, 2004b).

Each pitch value is associated to a musical note (C, D, E, etc.) or even a letter and a number. For instance: an "A" has the F0 of 440 Hz if it is played somewhere in the

Table 2.1: Numeric values of the pitches. C0 is the pitch 12 and B8 is the pitch 119.

Note	**Octave**								
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
C	12	24	36	48	60	72	84	96	108
C#	13	25	37	49	61	73	85	97	109
D	14	26	38	50	62	74	86	98	110
D#	15	27	39	51	63	75	87	99	111
E	16	28	40	52	64	76	88	100	112
F	17	29	41	53	65	77	89	101	113
F#	18	30	42	54	66	78	90	102	114
G	19	31	43	55	67	79	91	103	115
G#	20	32	44	56	68	80	92	104	116
A	21	33	45	57	69	81	93	105	117
A#	22	34	46	58	70	82	94	106	118
B	23	35	47	59	71	83	95	107	119

Table 2.2: Corresponding frequency of each MIDI pitch, from C0 to B8. C0 has the frequency 16 Hz (more precisely 16.3515978313 Hz) and B8 has 7,902 kHz (more precisely 7902.1328200980 Hz).

Note	**Octave**								
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
C	16	33	65	131	262	523	1047	2093	4186
C#	17	35	69	139	278	554	1109	2218	4435
D	18	37	73	147	294	587	1175	2349	4699
D#	20	39	78	156	311	622	1245	2489	4978
E	21	41	82	165	330	659	1319	2637	5274
F	22	44	87	175	349	699	1397	2794	5588
F#	23	46	93	185	370	740	1475	2960	5920
G	25	49	98	196	392	784	1568	3136	6272
G#	26	52	104	208	415	831	1661	3322	6645
A	28	55	110	220	440	880	1760	3520	7040
A#	29	58	117	233	466	932	1865	3729	7459
B	31	62	124	247	494	988	1976	3951	7902

middle of a piano keyboard. This way, that note can also be represented by A440 or A4. The A5 is also an A but it is an octave higher (the next A from the lower notes to the higher notes) and its F0 is 880 Hz. The human perception of pitch is approximately logarithmic regarding the F0: the distance perceived between the pitches A220 and A440 (220 Hz difference) is the same between the pitches A440 and A880 (440 Hz difference).

In order to avoid possible problems, the pitch is usually represented in a numeric scale based on the F0's logarithm. This way, we can adapt the MIDI standard Association

(2008) to map the F0 f to a simple integer value p (see Equation 2.2).

$$p = 69 + 12 \times \log_2 \left(\frac{f}{440Hz} \right) \tag{2.2}$$

This creates a linear pitch distribution, where the octaves have size of 12, and the semitones the size of 1. For instance: the pitch A440 has the value 69.

Table 2.1 shows the numeric scales of the pitches and Table 2.2 shows the corresponding frequencies.

2.3 Digital Signal Processing

Digital Signal Processing consists in the application of several computational methods on digital signals, or digitized versions of natural signals (sound, image, electrocardiogram potentials, seismic vibrations, etc.), to analyze, classify, recognize or even transform them.

Nowadays, every way of representation and communication of voice, sound and image are digital and apply Digital Signal Processing techniques. For instance: the JPEG, MPEG, MP3 and GSM are no more than techniques of Digital Signal Processing.

Digital Signal Processing can also create digital signals from mathematical models and computational methods, like the image synthesis and the computer animation, speech synthesis and music composition by computer, where the MIDI format (Association, 2008) is very popular.

Basic Concepts

A signal is said to be **periodic** if it repeats itself at a regular time interval: its **period**. A periodic signal is mathematically defined as a function $\tilde{x}(t)$, where $\tilde{x} \in \mathbb{R}$, which follows the Equation 2.3, where T is a real number and $m \in \mathbb{Z}$.

$$\tilde{x}(t) = \tilde{x}(t + mT) \tag{2.3}$$

The lowest positive value of T where the expression is true is called the period of the fundamental component and is designated as T_0. The fundamental frequency ω_0 is defined, according to the period of the fundamental by:

$$\omega_0 = \frac{2\pi}{T_0}. \tag{2.4}$$

Figure 2.6 shows a periodic signal $\tilde{x}(t) = \sin(\pi t)$, which has the fundamental frequency $\omega_0 = \pi$, which corresponds to the period $T_0 = 2$.

Given the fact that the F0 of a speech or music sound source varies with time, one assumption is made: it is assumed that the signal is **stationary** in a very short time duration. This way, the F0 of a non-stationary periodic signal can be determined through the approximation $\tilde{x}(t) \approx \tilde{x}(t + T_0)$ for the concerned duration. If a signal can be approximated by using this method, it is called a **quasi-periodic** signal. Figure 2.7 illustrates

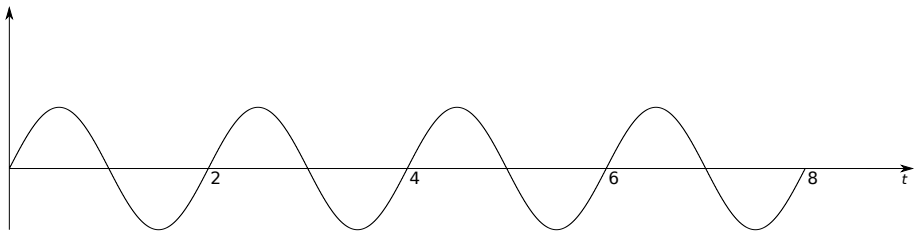

Figure 2.6: Periodic signal.

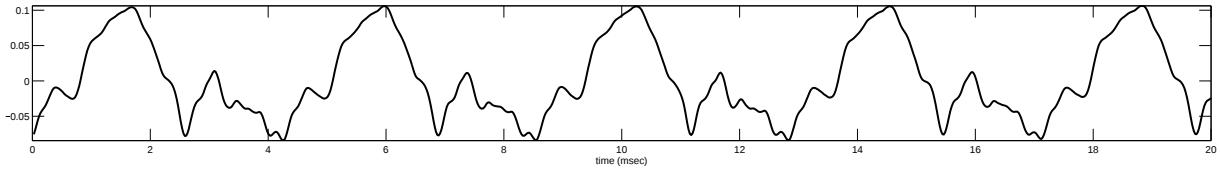

Figure 2.7: Waveform of a quasi-periodic signal, generated by a saxophone, with F0 = 237Hz ($T_0 = 4.2$ms)

an example of a quasi-periodic signal, with distinctive periods. On the contrary, periodic non-stationary signals have several and distinctive periods, which in turn may have competitive fits to the signal, resulting in an ambiguity in the determination of the F0.

2.3.1 Fourier Analysis

Sinusoids are probably the most important of all periodic signals. In general, a sinusoid is represented by a cosine function with a specific amplitude A, frequency w_0 and initial phase ϕ:

$$\tilde{x}(t) = A\cos\left(w_0 t + \phi\right). \tag{2.5}$$

Jean-Baptiste Joseph Fourier (1768 - 1830) was the first to have the insight to see the potential for representing a signal as a *sum of harmonically related sinusoids*, where each component is called a **harmonic** and has a frequency that is a multiple of the fundamental frequency. According to Joseph Fourier, any periodic signal can be represented by a **Fourier series**. This was later demonstrated by P. L. Dirichlet (Oppenheim et al., 1997).

The study of signals and systems using the sinusoidal representation is known as **Fourier analysis**, in honor to Joseph Fourier. Fourier analysis is a family of mathematical techniques based on the sinusoidal signal decomposition (each function is treated as an infinite sum of sines). Signals can be classified according to two criteria: discrete or continuous and periodic or non-periodic. The four elements of the family of the Fourier transforms came from these two criteria, being the most important for this thesis the Fourier Series - FS (applies to periodic and continuous signals) - and the Discrete Fourier Transform - DFT (applies to periodic signals with discrete time). The Discrete Fourier Transform is the only class of these two representations that can be used in Digital Signal Processing, since computers can only deal with discrete information and with finite size.

Fourier Series - FS

Any periodic signal represented by a real function $\tilde{x}(t)$ can be constructed as a sum of a number (possibly infinite) of harmonically related complex exponentials:

$$\tilde{x}(t) = \sum_{k=-\infty}^{\infty} a_k e^{jkw_0 t}, k \in \mathbb{Z} \tag{2.6}$$

where $w_0 = 2\pi F_0 = \frac{2\pi}{T_o}$. An important property of the Fourier series representation is that the sinusoidal functions form an orthonormal basis for the space $L_2([-\pi, \pi])$ of square-integrable functions of $[-\pi, \pi]$. The Fourier series coefficients can thus be efficiently computed as follows:

$$a_k = \frac{1}{T_0} \int_{T_0} \tilde{x}(t) e^{-jkw_0 t} dt. \tag{2.7}$$

Equation 2.6 is referred to as synthesis equation and Equation 2.7 as the analysis equation. By rearranging Equation 2.6, we have:

$$\tilde{x}(t) = a_0 + \sum_{k=1}^{+\infty} \left(a_k e^{jkw_0 t} + a_{-k} e^{-jkw_0 t}\right) = a_0 + \sum_{k=1}^{+\infty} \left(a_k e^{jkw_0 t} + a_k^* e^{-jkw_0 t}\right) \tag{2.8}$$

implying that $a_k^* = a_{-k}$. By further expressing a_k in polar form as $a_k = \frac{A_k}{2} e^{j\phi_k}$, we have the following trigonometric equation:

$$\tilde{x}(t) = a_0 + \sum_{k=1}^{+\infty} A_k \cos(kw_0 t + \phi_k). \tag{2.9}$$

This last equation (Equation 2.9) is commonly used for the Fourier series representation of periodic signals. Any sound that can be represented by this equation is called a **harmonic sound**. Since the harmonics of a quasi-periodic signal do not have frequencies that are exactly multiples of its F0, they are often referred to as **partials**. For practical use, a finite number of harmonic components **H** is generally used for quasi-periodic signals approximation:

$$\tilde{x}(t) \approx a_0 + \sum_{k=1}^{H} A_k \cos(kw_0 t + \phi_k). \tag{2.10}$$

Fourier Transform - FT

Waveforms are signal representations in the **time-domain** and, thus, can be used directly for several tasks such as beat detection. Also, by applying correlation to the time-domain signal, it is possible to search for repetitive patterns and, furthermore, detect periodicities on monaural signals. However, the information in the time-domain is not practical for some approaches that require a different kind of information.

By using the **Fourier transform** (FT), a waveform can be mapped (transformed) into the **frequency-domain**. A time-domain graph shows how a signal changes with time,

whereas a frequency-domain graph shows how much of the signal lies within each given frequency band over a range of frequencies. The FT decomposes a given signal function into a sum of sinusoids with different frequencies, showing how much of the signal lies within each frequency band. This way, FT is a widely used for frequency analysis. The FT is defined for aperiodic continuous waveforms with infinite length as follows:

$$\mathrm{FT}_{\tilde{x}}(f) = \tilde{X}(f) = \int_{-\infty}^{+\infty} \tilde{x}(t) e^{-j2\pi ft} dt. \tag{2.11}$$

Discrete Fourier Transform - DFT

When a periodic signal is discrete in time, like in digital signal processing, the FT cannot be applied. Therefore, for analyzing a discrete signal the Discrete Fourier Transform is applied. Equation 2.12 shows the DFT calculation, where k is the spectral bin corresponding to each frequency:

$$\mathrm{DFT}_{\tilde{x}}[k] = \tilde{X}[k] = \sum_{n=-\infty}^{+\infty} \tilde{x}[n] e^{-j2\pi kn}. \tag{2.12}$$

Computational problems require signals to have finite length, which stresses the need of a DFT for finite signals. The DFT for finite signals is defined as follows:

$$\mathrm{DFT}_{\tilde{x}}[k] = \tilde{X}[k] = \sum_{n=0}^{N-1} \tilde{x}[n] e^{-j\frac{2\pi}{N}kn}, \quad k = 0, \cdots, N-1 \tag{2.13}$$

where N is the length of the waveform (number of samples).

According to Nyquist-Shannon sampling theorem (Shannon, 1949), the number of useful frequencies of the DFT is limited to the Nyquist frequency $\left(\frac{F_s}{2}\right)$. Given that the N frequency bins are equally distributed, the frequency of each spectral bin k is: $f_k = k\frac{F_s}{N}$. This way, the frequency resolution of the DFT is $\Delta f = \frac{F_s}{N}$.

As an example, let us assume that a stationary quasi-periodic signal $\tilde{x}[t]$ has 4096 samples and its sampling rate is 44100 Hz (CD quality). This means that by applying the DFT, the signal $\tilde{x}[t]$ is transformed into the signal $\tilde{X}[k]$, where the frequency resolution of each bin is $\Delta f = \frac{F_s}{N} = \frac{44100}{4096} = 10.77Hz$. This means that the frequency of each spectral bin k is $f_k = k \times 10.77Hz$.

Fast Fourier Transform - FFT

The calculation of the DFT for N samples requires N^2 complex multiplications and $N^2 - N$ complex additions, to obtain N samples in the frequency-domain. This number of multiplications and additions leads to a prohibitive DFT calculation in real-time. However, if the number of samples N is a structured number such as a power of two, then the DFT can be efficiently computed using a **fast Fourier transform** (FFT) algorithm.

The Fast Fourier Transform is a fast algorithm to implement efficiently the DFT, where a number of N samples of the input signal are transformed in N frequency points. For N structured as a power of two, the required computational effort for this operation

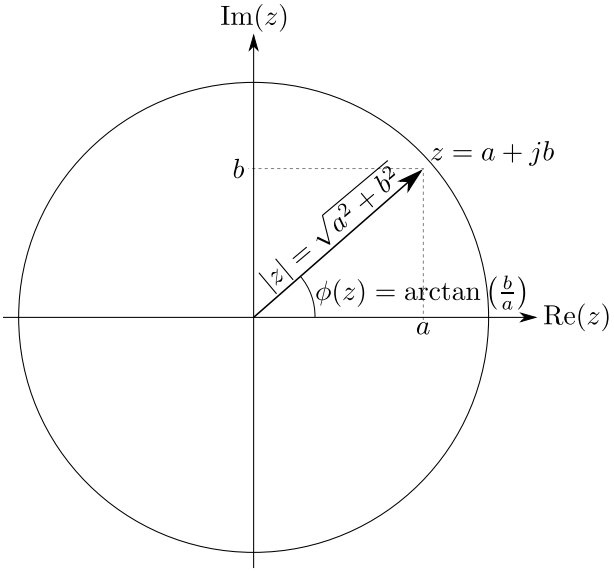

Figure 2.8: Complex plane diagram. Magnitude and phase of the complex number z are shown.

is proportional to $N \log_2 N$ in terms of multiplications and additions. This allows DFT computation in real-time.

2.3.2 Power Spectral Density - PSD

Although these Fourier transform equations (FT, DFT and FFT) were described in terms of complex exponentials, these Fourier transforms can also be expressed as trigonometric functions, as a consequence of the complex representation according to Euler's formula. Complex numbers are represented in the complex plane as $z = a + jb$ (see Figure 2.8), where a is the real part, and b is the imaginary part. j is the square root of -1: $j = \sqrt{-1}$. The radial position or magnitude $|z|$, and the angular position or phase $\phi(z)$ can be computed from the complex value: $|z| = \sqrt{a^2 + b^2}$ and $\phi(z) = \arctan_2\left(\frac{b}{a}\right)$, that is:

$$\phi(z) = \arctan_2\left(\frac{b}{a}\right) = \begin{cases} \arctan\left(\frac{b}{a}\right) & a > 0 \\ \arctan\left(\frac{b}{a}\right) + \pi & b \geq 0, a < 0 \\ \arctan\left(\frac{b}{a}\right) - \pi & b < 0, a < 0 \\ +\frac{\pi}{2} & b > 0, a = 0 \\ -\frac{\pi}{2} & b < 0, a = 0 \\ \text{undefined} & b = 0, a = 0 \end{cases} \tag{2.14}$$

The **power spectral density** (PSD) describes how the energy of a signal is distributed with frequency. PSD is the squared magnitude of the DFT of a signal $\tilde{x}[n]$. In general, it is called simply the **spectrum** of the signal. The spectrum is usually represented as a two-dimensional diagram showing the energy of a signal $|\tilde{X}[k]|^2$ as a function of frequency (see Figure 2.9). Although the spectrum in Figure 2.9 is represented using a linear scale for both magnitude and frequency, the logarithmic scale is also commonly used to show

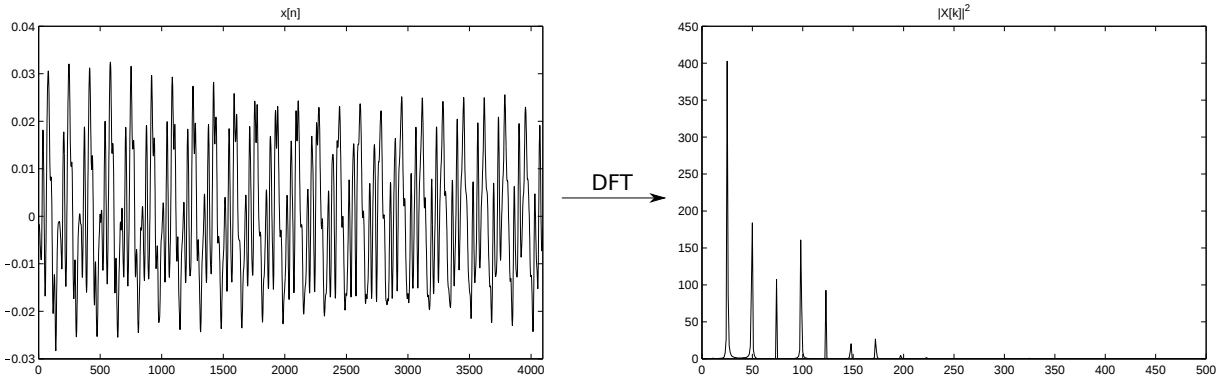

Figure 2.9: Power spectrum of a Steinberg piano (middle C note).

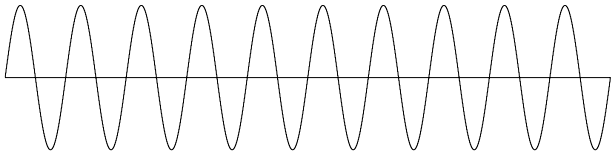

Figure 2.10: Pure sine wave in the time-domain.

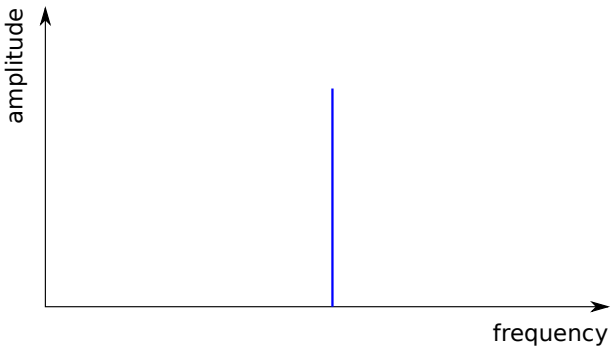

Figure 2.11: Single spectral line in the frequency-domain.

each component. A logarithmic magnitude widely used to represent the magnitudes is the deciBel (dB):

$$\mathrm{dB}(|\tilde{X}[k]|) = 20\log_{10}(|\tilde{X}[k]|) = 10\log_{10}(|\tilde{X}[k]|^2). \tag{2.15}$$

Through this dissertation, PSD will be referred to as **power spectrum**, whereas **magnitude spectrum** will be referred to the DFT magnitudes $|\tilde{X}[k]|$, also as a function of frequency.

2.3.3 Spectral Leakage

The Fourier Transform assumes that the signal is periodic. Therefore, a continuous sine wave (see Figure 2.10) is transformed into a single spectral line in the frequency-domain (see Figure 2.11).

However, in the case of the Discrete Fourier Transform a finite section of the signal history is transformed. If a pure sine wave does not repeat exactly within the time window,

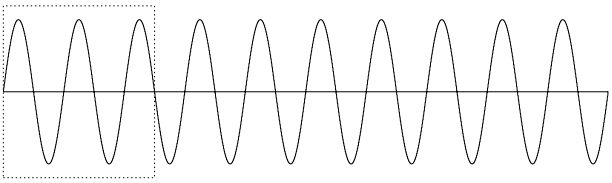

Figure 2.12: Windowed section of the signal history.

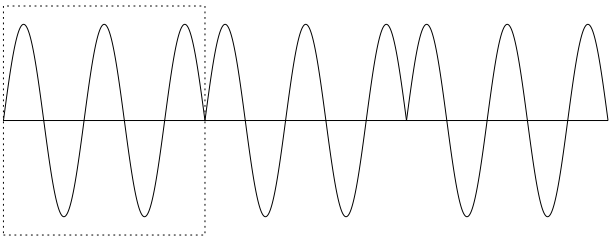

Figure 2.13: Repeated window section of the signal history.

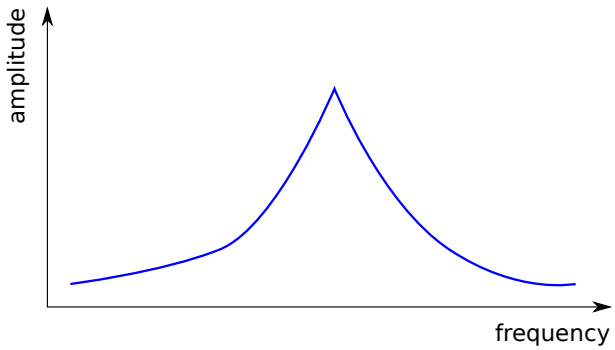

Figure 2.14: Spectral leakage on the frequency-domain.

it is truncated (see Figure 2.12) and the windowed section of the signal history is assumed to repeat (see Figure 2.13). This truncation will lead the spectrum to spread out of a single frequency (see Figure 2.14). This phenomenon is known as **leakage** due to the signal energy being leaked from a single frequency bin to adjacent frequency bins. Leakage reduces the accuracy of the spectrum: the measured level of the peaks is less accurate and also the effective frequency resolution of the analysis is strongly affected.

2.3.4 Windowing

In order to minimize the spectral leakage effect, caused by the direct truncation of a signal into a single time window (rectangular windowing), the samples in the frame can be multiplied by a smooth window shape, thus, smoothing the abrupt edges. This process is called **windowing**. The most common windows are rectangular, triangular, Hanning, Hamming, Blackman and Blackman-Harris. In general, windows have shapes that are positive, bell shaped and symmetric functions. A study on the use of different window functions, their trade-offs and comparison was presented by Harris (1978).

Windows are used to modify the frequency response of a DFT as a way to reduce the spectral leakage. The output of a DFT applied to a windowed function is the product of

two sequences given by Equation 2.16, where a_n is the applied window.

$$Y_k = \sum_{n=0}^{N-1} a_n \tilde{x}_n W_N^{nk}, \quad W = e^{-j2\pi}. \tag{2.16}$$

2.3.5 Relation between the signal's properties

There are several relevant implications of relevant parameters such as the sample rate, the number of samples, the window function and that affect the DFT analysis. For instance: suppose that we have a music piece with a duration of 2 seconds, with a sampling rate of 44100 samples per second and the DFT window size is 4096 samples, this means that we have:

- Time Record = $2 \times 44100 = 88200$ samples
- $\Delta F = \frac{44100}{4096} = 10.7$, thus each bin will correspond to 10.7 Hz.
- $\Delta t = \frac{4096}{44100} = 0.09$, thus the window of the stationary signal corresponds to 0.09 seconds.

This means that our algorithm only distinguishes accurately frequencies in intervals of about 11 Hz - ΔF -, therefore it can only distinguish the frequencies starting on pitch 54 (pitch F♯3) but it can validate pitches with duration of 0.09 seconds or greater. The DFT size is strictly related with the type of information we want to focus on concerning detail: frequency or time. If we want to increase the frequency resolution, the size of the DFT should also be increased, thus making the ΔF (frequency intervals) value to decrease. On the other hand, this will lead to a greater value of Δt which may imply the non-detection of short-duration frequencies. If we decrease the size of the DFT we will achieve better time resolution but it will be very difficult to distinguish near frequency values (the intervals - ΔF - are wider). This demands a trade-off between time resolution and frequency resolution.

2.4 Fundamental Frequency and Pitch

Pitch is an auditory sensation in which a listener perceives the tonal height of a sound: how "high" or "low" a note sounds. As already noted in Section 2.2, frequency is an objective and scientific concept that can be physically measured, while pitch is subjective and depends not only on frequency but also on spectral content and loudness. In spite of the fact that sound wave oscillations can be measured to obtain a frequency, sound waves themselves do not have pitch. It takes a human brain to map the internal quality of pitch. In general, pitches are quantified as frequencies by comparing sounds with pure tones, which are periodic, sinusoidal waveforms. On the other hand, pitch of complex tones can be ambiguous: two or more different pitches can be perceived, depending upon the observer. For instance, an audio signal consisting of two pure tones of 1000 Hz and 1200 Hz may sometimes be heard as up to three pitches: two spectral pitches as 1000 Hz and

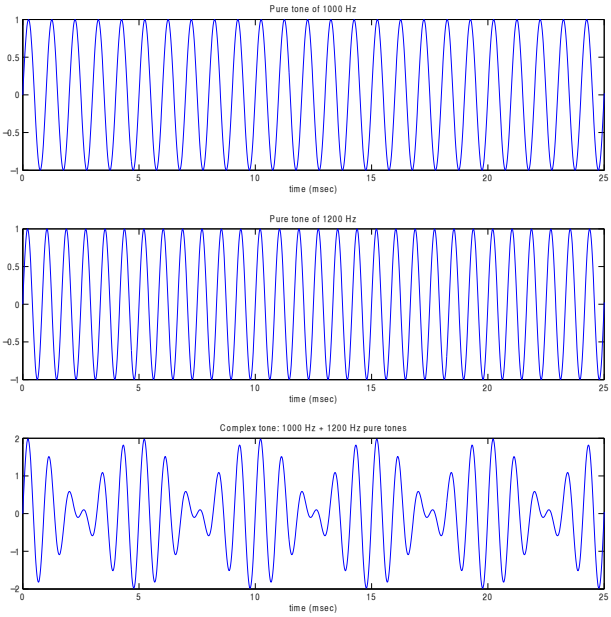

Figure 2.15: Missing fundamental resulting at 200 Hz.

1200 Hz, percept from the physical frequencies of the two pure tones, and an additional pith derived from the combination tone at 200 Hz, corresponding to the repetition rate of the waveform (see Figure: 2.15). The additional perceived pitch or *missing fundamental* is often the greatest common divisor of the frequencies present in the signal.

The American National Standard Institute (ANSI) defines pitch as "that auditory attribute of sound according to which sounds can be ordered on a scale extending from low to high". The French standards organization Association Française de Normalisation (AFNOR) defines: "pitch is associated with frequency and is low or high according to whether this frequency is smaller or greater". However, both verbal definitions are quite abstract. An operational definition is given in Hartmann (1997): "sound has certain pitch if it can be reliably matched by adjusting the frequency of a sine wave of arbitrary amplitude".

According to Bregman (1990), theories of auditory pitch analysis tend to differ in two dimensions: place coding and temporal coding. While place information is related to the **spectral pitch**, temporal coding is related to **periodicity pitch**. Hartmann (1996) states that the perception of pitch for complex tones resembles pattern matching: a sound is recognized through its spectral pattern, i.e. the series of partials that compose it. Furthermore, if some spectral components are too weak to be perceived, the human auditory system tends to reconstruct those missing partials so that the pattern matching task can be completed.

For most authors, the term **F0 estimation** is equivalent to **pitch estimation**. However, we consider F0 estimation as the extraction of exact frequency components that model the signal, and the latter as the extraction of musical tones (or perceived pitches) present in the signal. This way, throughout this dissertation, when referring to *Automatic Transcription of Music* we mean multi-pitch estimation because the goal of this work is to extract what is perceived as a musical pitch and not to extract the F0 as a parameter

of the signal model.

2.5 Single-F0 Estimation

Single-F0 applies to short-time signals that have one harmonic source (monophonic or monaural signals). Without loss of generality, and as pointed by Yeh (2008) p. 9, the observed signal can be expressed as a sum of a quasi-periodic part $\tilde{x}[n]$ and the residual $z[n]$:

$$x[n] = \tilde{x}[n] + z[n] \approx \sum_{k=1}^{H} A_k \cos(kw_0 t + \phi_k) + z[n] \tag{2.17}$$

where Equation 2.10 is used for approximation.

The single-F0 estimation problem lies on the extraction of the fundamental period or the fundamental frequency of $\tilde{x}$. Note that the goal is not to minimize the residual $z[n]$, but to extract the quasi-periodic part $\tilde{x}[n]$ with high periodicity/harmonicity. The most common errors made, when addressing this problem, are **subharmonic errors** and **super-harmonic errors**. In either case, the estimated F0s are harmonically related to the correct F0: subharmonic errors correspond to those which are *unit fractions* of the correct F0 and super-harmonic errors correspond to those which are *multiples* of the correct F0.

In general, single-F0 estimation methods can be categorized according to their domain: some are temporal domain methods, others are frequency-domain. Temporal domain methods try to find the fundamental period, whereas frequency-domain methods rely on the spectral analysis. The mathematical formulation for the problem depends on the way the periodicity in $\tilde{x}[n]$ is extracted. Most algorithms do not use an explicit source model as expressed in the approximation of Equation 2.17, but rather attempt to extract directly the periodicity in either the time-domain or the spectral domain (Yeh, 2008).

2.5.1 Spectral-location Approaches

By following the definition of periodic signals in Equation 2.3, time-domain methods try to look for a repetitive patterns in $x[n]$ through pattern matching between the signal itself ($x[n]$) and a delayed version of $x[n]$, i.e. $x[t + \tau]$. The idea is to find the delay or **lag** that gives the best correlation between the delayed signal and the signal itself. The lowest lag that has the highest correlation corresponds to the period of the observed signal. In general, pattern matching in the time-domain is carried out through multiplication between patterns to measure their correlation.

Autocorrelation

The **autocorrelation** function (ACF) is a tool allowing to measure the similarity between a signal and delayed versions of itself. Analytically, it corresponds, for a given delay (or lag), to the cross-correlation of a signal with itself. This way, quasi-periodic signals will

have high correlation measure (similarity) when the lag corresponds to the fundamental period of the signal or to multiples of the fundamental period. Analytically, the ACF is obtained as the sum of the product between a signal $x[n]$ of finite duration L and its delayed version $x[n+\tau]$, as a function of the lag τ:

$$\mathrm{ACF}[\tau] = \frac{1}{L-\tau} \sum_{n=0}^{L-\tau-1} x[n]x[n+\tau]. \tag{2.18}$$

After performing the autocorrelation in the signal, the highest and non-zero-lag peak is chosen as the estimated fundamental period. However, this method is very sensitive to formants in speech signals and also to resonances in music signals. Hence, center clipping, spectral flattening and nonlinear distortion and other special post-processing techniques were proposed by several authors (Hess, 1983) in addition to autocorrelation.

Magnitude difference

The **Average Magnitude Difference Function** (Ross et al., 1974) compares the dissimilarity of $x[n]$ and $x[n+\tau]$ by evaluating the *distance* between the two patterns:

$$\mathrm{AMDF}[\tau] = \frac{1}{L-\tau} \sum_{n=0}^{L-\tau-1} |x[n] - x[n+\tau]|. \tag{2.19}$$

For quasi-periodic signals, AMDF results in dips, showing low dissimilarity between the signal and its delayed version, when the lag equals to the fundamental period or multiples of the fundamental period. After computing the magnitude difference, the deepest non-zero-lag dip is selected as the estimated fundamental period. Besides AMDF, dissimilarity can also be measured by using the *squared distance*:

$$\mathrm{SDF}[\tau] = \frac{1}{L-\tau} \sum_{n=0}^{L-\tau-1} (x[n] - x[n+\tau])^2. \tag{2.20}$$

The **Squared Difference Function** (SDF) can be generalized into any power of the distance measure. It has been investigated that a power larger than one is appropriate for weekly stationary signals and, according to Nguyen and Imai (1977), power of two seems to be fair. This function was used by De Cheveigné and Kawahara (2002) on the YIN algorithm: SDF is normalized by its average over shorter values of τ, removing dips at τ near zero and, thus, avoiding super-harmonic errors. This process is called the **cumulative mean normalized difference function**. A study, showing how YIN outperforms several traditional methods on speech signals was also presented by De Cheveigné and Kawahara (2001).

Although both AMDF and SDF are used to measure dissimilarity between patterns, both are also related to the autocorrelation function, which computes similarity. Moreover, despite the fact that these methods are applied to the time-domain (signal waveform) they are phase-insensitive since partials are being subtracted regardless of their phases. However, as demonstrated by Hess (1983), those methods are prone to errors caused by intensity variations, noise and also low-frequency spurious signals.

Cepstrum

Cepstrum is the result of taking the Fourier transform (FT) of the logarithm of the estimated spectrum of the observed signal. There is a complex cepstrum, a real cepstrum, a power cepstrum, and phase cepstrum. The name "cepstrum" was derived by reversing the first four letters of "spectrum". Operations on **cepstra** are labeled **quefrency analysis**, **liftering**, or **cepstral analysis** (Oppenheim and Schafer, 2004).

When a signal exhibits a periodicity which is denoted by sinusoidal peaks in its spectrum, it makes sense to apply again the Fourier analysis on the observed spectrum for analyzing the underlying periodicity. The real cepstrum of a signal is given by the inverse Fourier transform of the logarithm of its power spectrum:

$$c(\tau) = \text{IDFT}\{\log\left(|\text{DFT}\left(x[n]\right)|\right)\}. \tag{2.21}$$

Cepstrum was initially proposed by Schroeder 1962 for F0 estimation based on the first cepstral analysis paper on seismic echoes resulting from earthquakes and bomb explosions. A short-time cepstrum analysis was proposed later for application to pitch determination of human speech by Noll (1967). Although **quefrency** is a measure of time, the signal is not in the time-domain. For instance, if a signal has a sampling rate of 44100 Hz, and if there is a large peak whose quefrency is 200 samples, this peak indicates a presence of a frequency component in the signal that is $\frac{44100}{200} = 220,5$ Hz. This way, while lower-**quefrency** components in the cepstrum essentially provide spectral envelope information, components that appear as cepstral peaks correspond to estimated fundamental periods.

Both ACF based and cepstrum-based single-F0 estimation methods are implicit realizations of a model which emphasizes frequency partials at harmonic locations of the magnitude spectrum (Klapuri, 2004a). This can be seen by writing the ACF in terms of the Fourier spectrum $X[k]$ of a real-valued input signal as:

$$r(\tau) = \frac{1}{K}\sum_{k=0}^{K-1}\left[\cos\left(\frac{2\pi\tau k}{K}\right)|X[k]|^2\right] \tag{2.22}$$

where K is the length of the transform frame. The above formula is equivalent to:

$$r(\tau) = \text{IDFT}\{|\text{DFT}\left[x[n]\right]|^2\}. \tag{2.23}$$

The definition of the cepstrum $c(\tau)$ of $x[n]$ (Equation 2.21) is very analogous to $r(\tau)$ (Equation 2.23) and is obtained by replacing the second power with a logarithm function. This way, the difference between the ACF and cepstrum-based F0 estimators is quantitative: raising the magnitude spectrum to the second power emphasizes spectral peaks in relation to noise but, on the other hand, further aggravates spectral peculiarities of the target sound. On the other hand, applying the logarithm function causes the opposite for both. This way, ACF-based F0 estimators have been reported to cope with noise but are prone to errors due to formant structures in speech: especially the first and the strongest formant may mislead the algorithm (Rabiner et al., 1976; Talkin, 1995). Cepstrum-based F0 estimators perform relatively poorly in noise, but perform well for

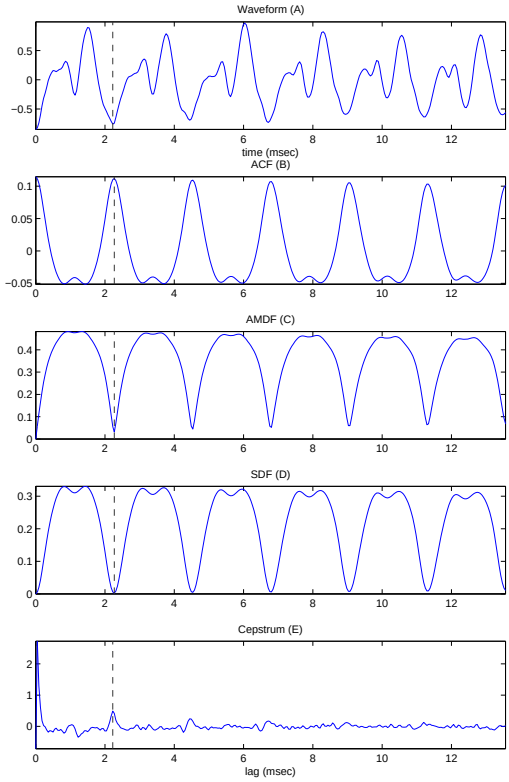

Figure 2.16: Three time-domain salience functions for a baritone sax signal of T0 = 2.3ms (A) signal waveform; (B) autocorrelation function; (C) average magnitude difference function; (D) squared difference function; and (E) cepstrum.

exotic sounds (Rabiner et al., 1976). As a trade-off, Tolonen et al. suggest using a "generalized autocorrelation function" replacing second power by real-valued exponent (0.67 in their case) (Tolonen and Karjalainen, 2000).

Figure 2.16 shows four time-domain functions (Autocorrelation, AMDF, SDF and cepstrum) for a piano signal with F0 = 440 Hz (T_0 = 2.3ms). The period of the signal is represented by a vertical dashed line in each plot. Although there is some diversity on the methods to measure the similarity between a short-time signal and its delayed versions (ACF, SDF, AMDF, etc.), the main problem lies in the ambiguity on the selection of the best period. This happens because all multiples of the period, i.e. subharmonics, are competitive candidates for being selected as the estimated period.

2.5.2 Spectral-interval Approaches

Periodic but non-sinusoidal signals have periodic magnitude spectra and the corresponding period is the F0. Spectral domain approaches, in general, estimate the F0 by employing either one of these methods: measuring the regular spacing of dominant peaks; or formulation of the salience of the F0 as a function of hypothetical partials. This way, F0 can

also be defined as the *greatest common divisor* of the frequencies of all harmonics.

Frequency-domain methods to F0 estimation extract periodicity from the frequency-domain, i.e. the spectral representation of the observed signal, based on the Fourier Transform. Although the cepstrum also extracts periodicity of the frequency representation of the observed signal, the cepstrum does not belong to the spectral-interval approaches because, by applying the inverse Fourier transform, the observed signal is converted from the frequency representation back to the time-domain. Moreover, cepstrum finds the fundamental period T_0, such as the autocorrelation based methods.

Spectral autocorrelation

Spectral-location F0 estimators are very sensitive to harmonic deviations from their ideal positions: **inharmonicity**. Nevertheless, ACF function can also be used, in a similar way to the search for repetitive patterns in the time-domain, on the spectral domain (Lahat et al., 1987). The periodicity search is thus attained by pattern matching between the spectrum $X[k]$ and its shifted versions $X[k+m]$. The ACF over the positive frequencies of a K-length magnitude spectrum is calculated as:

$$\mathrm{ACFS}(m) = \frac{2}{K-2m} \sum_{k=0}^{\frac{K}{2}-m-1} |X\,[k]\,||X\,[k+m]\,|. \tag{2.24}$$

F0 corresponds to the local maximum of the ACFS(m) function for $m > 0$. In fact, when the shift m equals to period of the spectrum or multiples of its period, the product results in high spectral autocorrelation coefficients, the maximum occurring when the shift equals to $m = F0\left(\frac{K}{F_S}\right)$. On the other hand, when the shift m is not equal to the period of the spectrum, the corresponding autocorrelation coefficient is attenuated since the partials are not aligned.

Spectral compression

Schroeder proposed the **Schroeder histogram**, which counts the contribution of each spectral peak over *frequency-warped* spectra, in order to infer the F0 from the higher harmonics observed as spectral peaks. Frequency-warped spectra correspond to spectra compressed by different integer factors on the frequency axis. This histogram is computed by counting the effective contribution of each spectral peak to the related F0s that are also common divisors of its frequency. This method is not robust against noise and spurious peaks in the spectrum. Schroeder (1968) further proposed to weight the spectral components according to their magnitudes: **harmonic product spectrum** uses the log power spectrum and **harmonic sum spectrum** uses the linear spectrum. By summing compressed spectra, the energy of higher partials becomes concentrated on distinct peaks. The maximum peak determines the related F0.

Pattern matching

Pattern matching, also referred to as harmonic matching, approaches compare the observed spectrum against a given *harmonic spectral pattern*. The spectral pattern used for matching can either be a specific spectral model or a **harmonic comb**, i.e. a series of equally spaced spectral pulses, defined by an F0 hypothesis, without specifying the magnitudes of the harmonics. In general, for polyphonic signals, specific spectral models are used, whereas harmonic comb is generally used on single-F0 estimation. The similarity measure for an F0 hypothesis can be evaluated by the correlation between the harmonic comb and the observed spectrum, such as in the works of Martin (1982); Brown (1992), or by the minimization of the distance between the frequencies of the harmonics and the frequencies of the matched peaks, as in Goldstein (1973); Duifhuis et al. (1982). To improve the robustness of harmonic matching, several factors have also been studied, such as: the number of harmonics Goldstein (1973), the quality of the peaks (Sluyter et al., 1982), the tolerance interval (Sreenivas and Rao, 1981), the presence of harmonics (Doval and Rodet, 1991), etc.

Spectral peak inter-spacing

According to Harris and Weiss (1963), when partials are well separated in the spectrum the F0 can be estimated by measuring the regular spacing between each pair of partials: each F0 hypothesis corresponds to a group of spectral peaks that have frequency spacing close to the F0 hypothesis; the hypothesis corresponding to the best support group is chosen. In general, the measure of support is related to energy and harmonicity.

2.5.3 Unitary model of pitch perception

Some single-F0 estimation methods use **perceptual models** of the human hearing system. Meddis and Hewitt (1991a,b) introduced the **unitary** model of the hearing system, which is able to estimate the fundamental frequency of a signal by measuring the periodicity of the time-domain envelope. This model represents a trade-off between time periodicity (time-domain) and spectral interval (spectral domain) methods: analysis methods based on time periodicity are prone to errors in F0 halving, whereas frequency-domain methods are prone to errors in F0 doubling. This happens because the waveform of stationary signals is also periodic at twice the fundamental period (half of the F0), while the spectrum is periodic at double the F0 rate. This way, the unitary model is a good compromise between both methods. The unitary model is also widely accepted as a psychoacoustically valid mid-level representation (Klapuri and Astola, 2002). The first step of the unitary model consists in applying a cochlear frequency analysis using an auditory filter bank. Then, a simulation of the hair cells, which convert cochlear movements into neural impulses, is performed through half-wave rectification (HWR), compression and low-pass filtering of the signals at each frequency channel. Figure 2.17 (Klapuri (2004a), page 27) illustrates this step. The half-wave rectifier function is given by:

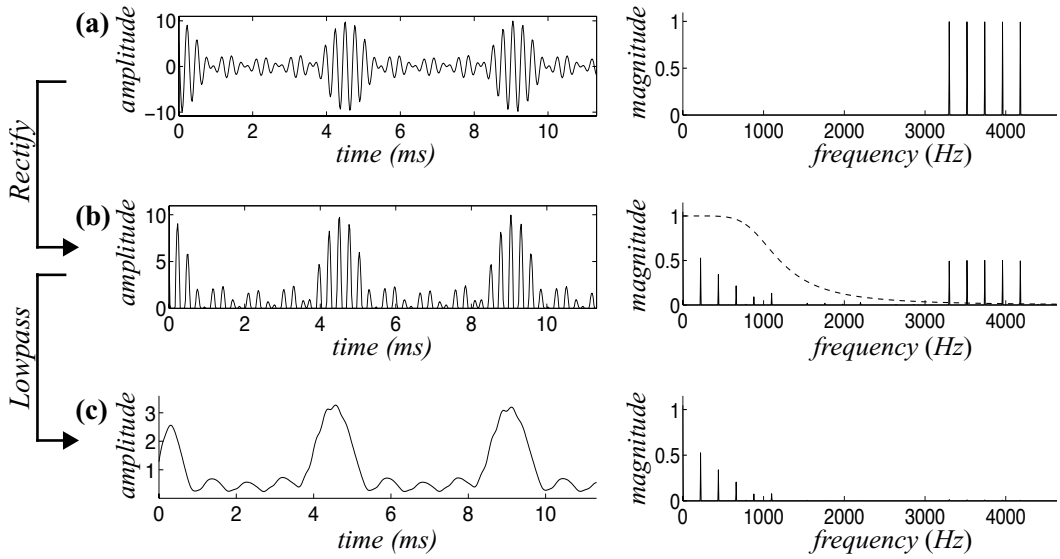

Figure 2.17: From top to bottom: (a) waveform of a signal containing the harmonics from 15 to 19 of a sound with F0 = 220Hz; (b) the same signal, after half-wave rectification; and (c) the signal after rectification and low pass filtering. The response of the lowpass filter is shown as a dashed line in (b) (from Klapuri (2004a), page 27).

$$\mathrm{HWR}(x) = \frac{x + |x|}{2} \quad = \quad \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} . \tag{2.25}$$

The periodicity at each channel is then estimated through the ACF function, generating a **correlogram** which is a three-dimensional representation of time, frequency and ACF lag. Finally, the **summary autocorrelation function** (SACF) is computed by summing the ACF across channels. The highest value given by the SACF function is considered the *perceived pitch*. For a broader review about pitch perception models see de Cheveigné (2005).

2.6 Multiple-F0 Estimation

In general, multi-pitch estimation algorithms assume that there can exist more than one harmonic source in the same short-time signal. As mentioned by Yeh et al. (2010), that signal can also be expressed as a sum of harmonic sources plus a residual[1]:

$$y[n] = \sum_{m=1}^{M} y_m[n] + z[n], \quad M > 0 \quad \text{with } y_m[n] \approx y_m[n + N_m] \tag{2.26}$$

[1]The residual - $z[n]$ - comes from components that are not explained by the sinusoids, for instance, the background noise, spurious components or non-harmonic partials.

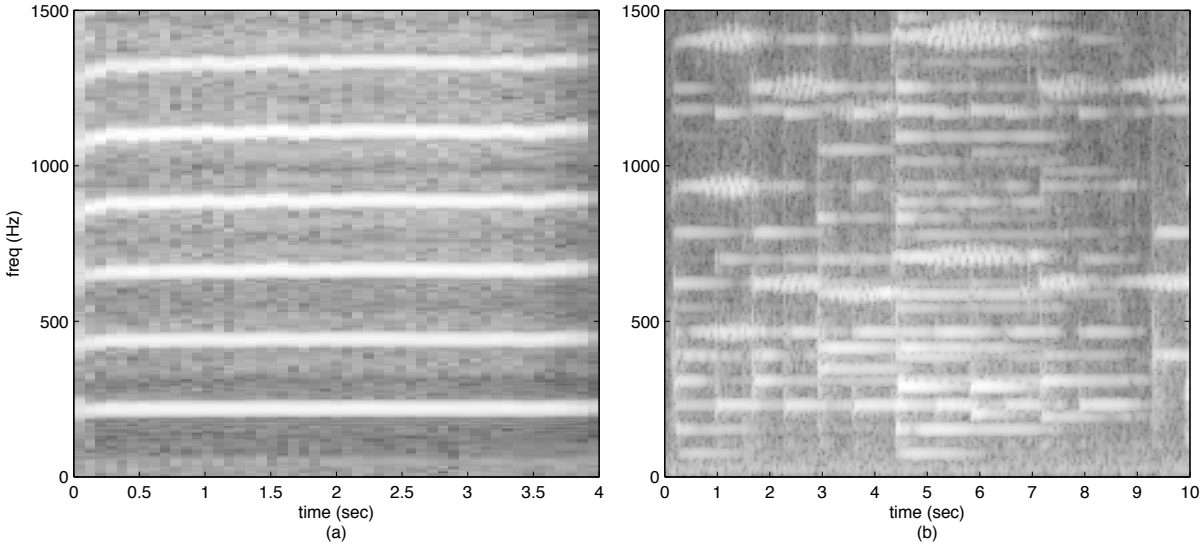

Figure 2.18: Comparison of the spectrogram of a monophonic signal with that of a polyphonic signal: (a) a trumpet note sample; (b) a piano and violin duo recording (from Yeh (2008), page 15).

where n is the discrete time index, M is the number of harmonic sources, $y_m[n]$ is the quasi-periodic part of the mth source, N_m represents the period of the mth source and $z[n]$ is the residual. Using the Fourier Series, this model can be represented as follows:

$$\begin{aligned} y[n] &= \sum_{m=1}^{M} \left\{ \sum_{k=1}^{\infty} A_{m,k} \cos(k\omega_m n + \phi_{m,k}) \right\} + z[n] \\ &\approx \sum_{m=1}^{M} \sum_{k=1}^{H_m} A_{m,k} \cos(k\omega_m n + \phi_{m,k}) + z[n]. \end{aligned} \tag{2.27}$$

The approximation on the last step of this equation is for practical use: a finite and small number of sinusoids H is commonly used to approximate a quasi-periodic signal.

2.6.1 Problem Complexity

Polyphonic signals are much more complex than monophonic signals. As an example, Figure 2.18 shows the complexity difference between the spectrogram of a monophonic signal (a) and a polyphonic signal (b). The difficulty of the problem of extracting multiple F0s from a music signal relies in the handling of overlapping partials, beating, transients, reverberation and also on the modeling of musical instruments sounds with diverse spectral characteristics. Moreover, the number of M sources must be also inferred, whereas in single-F0 estimation the only decision is to determine whether there is sound (and the corresponding pitch) or if there is silence. The noise model is also more complex than in monophonic signals and, in real polyphonic music, there can also be unpitched sounds (e.g. drums).

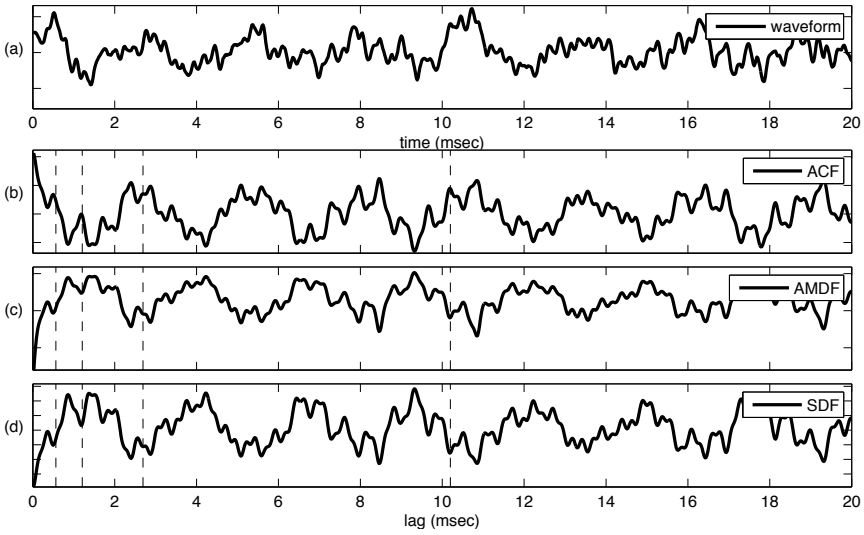

Figure 2.19: Three time-domain salience functions for a polyphonic signal containing four harmonic sources. The correct periods are marked by vertical dashed lines (from Yeh (2008), page 19).

In general, it is admitted that most single-F0 methods are not suitable for multiple-F0 estimation. Figures 2.19 (time-domain methods) and 2.20 (frequency-domain methods) depict the study performed by Yeh (2008), page 19, which shows the performance of different single-F0 estimation methods on the analysis of polyphonic signals.

Overlapping partials

For polyphonic signals, different sources with fundamental frequencies F_a and F_b are harmonically related when:

$$F_a = \frac{m}{n} F_b, \quad n, m \in \mathbb{N}. \tag{2.28}$$

When this happens, every n^{th} partial of the source a overlaps every m^{th} partial of source b (Klapuri, 1998). This way, and for equal temperament, the fundamental frequencies of most musical notes are harmonically related, resulting in a high probability of partial collision in polyphonic music signals. Also, if the fundamental frequencies of two musical notes form integer ratios (e.g. octave relation) all partials of the higher note might overlap with those of the lower note.

In musical signals, different sources often interfere with one another in such a way that their partials overlap in both time and frequency. This way, frequencies, amplitudes and phases of the overlapping partials of harmonic sources are disturbed: when two sounds are superposed, the corresponding wave functions are summed but, on the other hand, when there is harmonic overlap, two simple harmonic motions with the same frequency, but different amplitude and phases are added. This produces another simple harmonic motion with the same frequency but different amplitude and phase: when two harmonics are overlapped, two sinusoids of the same frequency are summed in the waveform, resulting

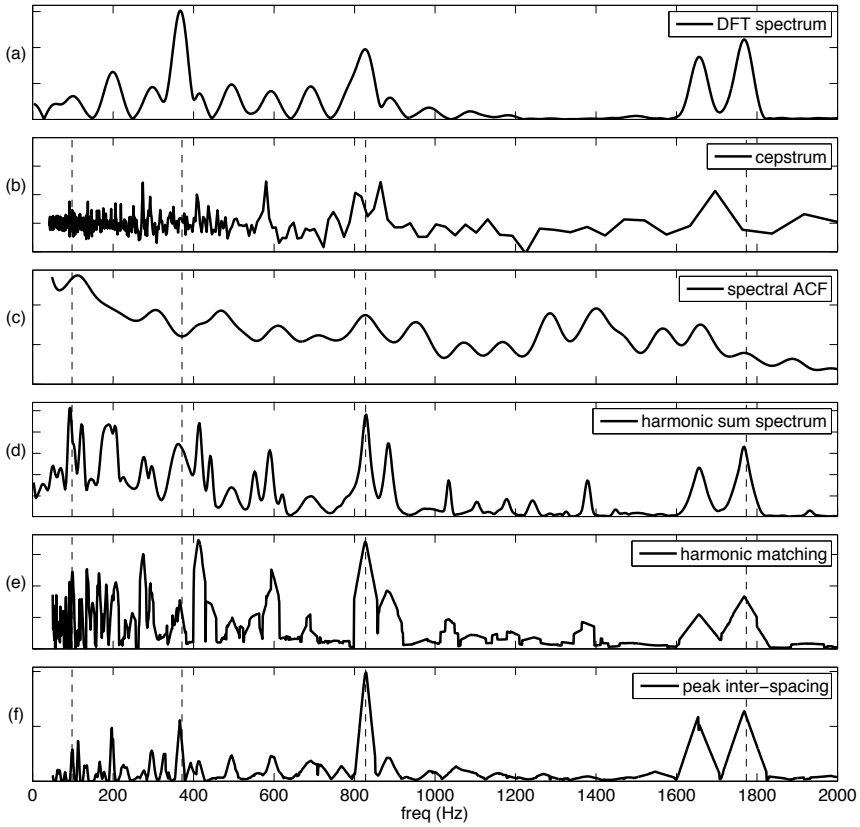

Figure 2.20: Five frequency-domain salience functions for a polyphonic signal containing four harmonic sources. The correct fundamental frequencies are marked by vertical dashed lines (from Yeh (2008), page 19).

in a signal with the same frequency and whose magnitude depends on their phase difference. This makes it difficult to locate overlapping partials.

Parsons (1976) tried to detect overlapping components by means of harmonic selection, relying on three tests: spectral peak symmetry, distance and well-behaved phase. Parsons's method relies on the sinusoidality of stationary sinusoids and is not appropriate for modulated sinusoids. Also, the maximum number of concurrent sources is limited to two, which is quite restrictive. Even when the number of concurrent sources is known, the decomposition of the overlapping partials into their original sources still remains a challenge, as highlighted by several authors: Viste and Evangelista (2002); Virtanen (2003a); Every and Szymanski (2004); Yeh and Roebel (2009).

Beating

As previously discussed, quasi-periodic signals have distinctive periods, i.e. several periods have competitive fitness to the signal, which might result in an ambiguity in the determination of the corresponding F0. Moreover, the same also happens with partials, which results in different partials with slight frequency deviations, changing over time. This way, if two overlapping partials with similar amplitude suddenly have a small frequency difference, **beats** can be perceived (see Figure 2.21).

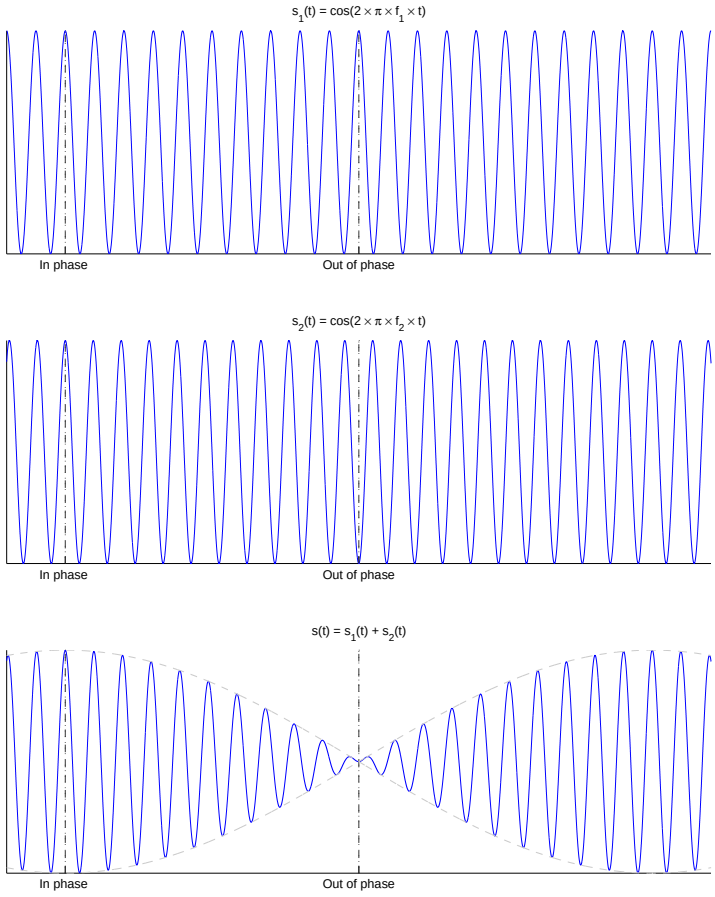

Figure 2.21: Interference tones of two sinusoidal signals of close frequencies f_1 and f_2.

According to Wood (2007), the physical explanation of dissonance is that we hear unpleasant beats. As illustrated in Figure 2.21, beats are periodic variations of loudness, and the frequency of the beats depend on the frequency difference of the two tones: $s(t) = s_1(t) + s_2(t) = \cos(2\pi f_1 t) + \cos(2\pi f_2 t)$, which is equal to:

$$s(t) = 2\cos\left(2\pi\frac{\Delta f}{2}t\right)\cos\left(2\pi\frac{f_1+f_2}{2}t\right), \tag{2.29}$$

being Δf the difference between f_1 and f_2, i.e. $f_1 - f_2$. This results in a single sinusoidal wave, with frequency $\frac{f_1+f_2}{2}$ and with a periodic variation of loudness whose frequency is Δf. This effect generates spectral components that do not belong to any original source, producing "ghost" fundamental frequencies and, also, changing the original partial amplitudes across the spectrum (see Figure 2.22).

Physical properties of musical instruments

In general, music signals are mixtures of musical notes played by various and different instruments. The diversity of spectral characteristics of musical instrument sounds increase

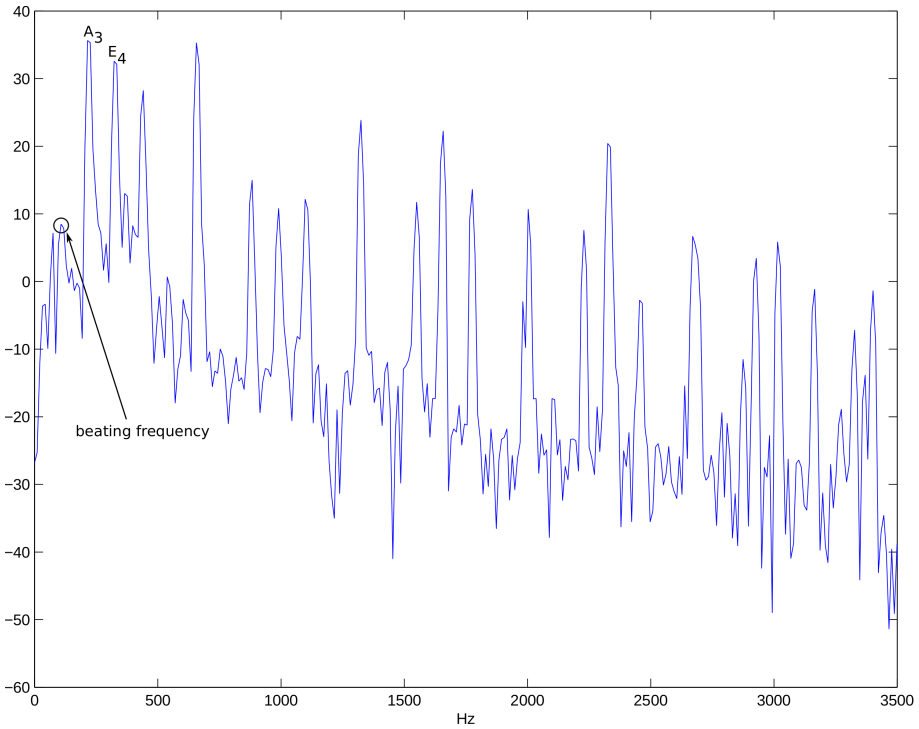

Figure 2.22: Example spectrum of two piano sounds with fundamental frequencies A3 (220.0 Hz) and E4 (329.6276 Hz). A beating component appears at frequency 110 Hz, corresponding to a A2 ghost pitch.

exponentially the complexity of the problem of multiple-F0 estimation.

1. **Spectral envelopes**: The spectral envelope of a harmonic signal shows the contour or spectral shape of the prominent spectral peaks which are, in general, the partials. Many musical instruments produce sounds with smooth spectral envelopes but, on the other hand, differ immensely in their shapes (see Figure 2.23, Yeh (2008)). Moreover, relatively weak F0s are often observed in the lower registers of some instruments like pianos, bassoons, oboes and guitars, resulting in rough spectral shapes. The spectrum of a clarinet sound, for instance, has attenuated even harmonics, which turns the spectral envelope bumpy. The spectral shape of musical instrument sounds also evolves with time in a way that partials decay at different rates. According to previous studies on modeling the spectral envelopes of musical instrument sounds, a universal model that generalizes different registers and musical instruments still has to be developed (Jensen, 1999; Loureiro et al., 2004; Burred et al., 2006).

2. **Inharmonic partials**: For an ideal harmonic sound, the frequencies of the harmonics are integer multiples of the F0. However, on musical instruments the partial frequencies are not exact integral ratios. For stretched strings, for example, the frequencies of the partials obey the formula:

$$f_h = hF\sqrt{1+\beta\,(h^2-1)}, \tag{2.30}$$

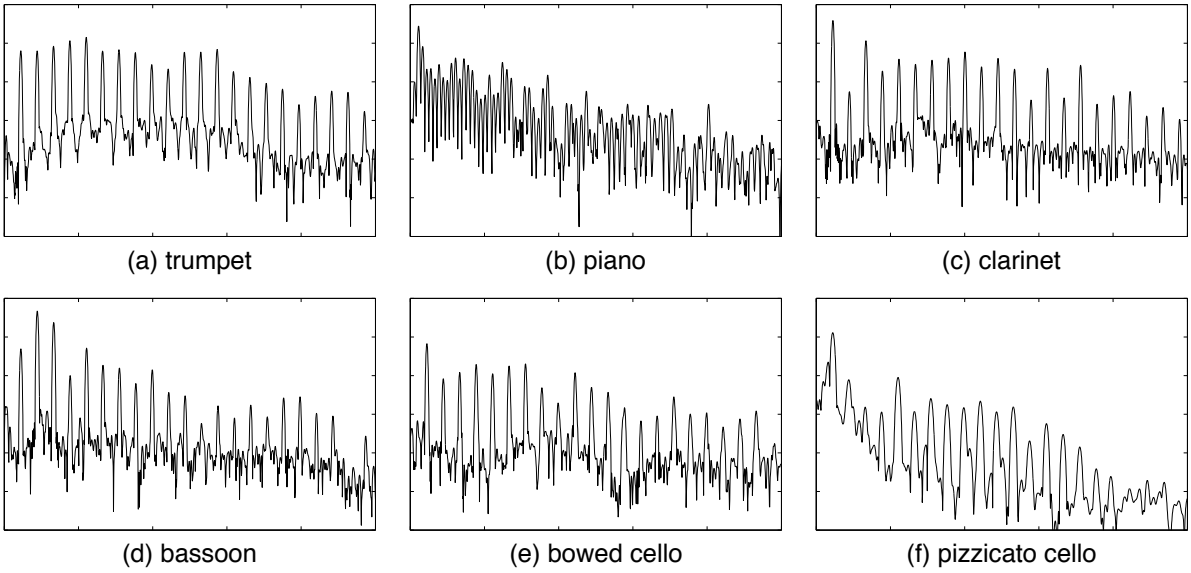

Figure 2.23: The spectra of six musical instrument sounds: (a) trumpet A3 note; (b) piano A1 note; (c) clarinet A3 note; (d) bassoon A3 note; (e) bowed cello A3 note; and (f) pizzicato cello A3 note. (Yeh (2008), page 17).

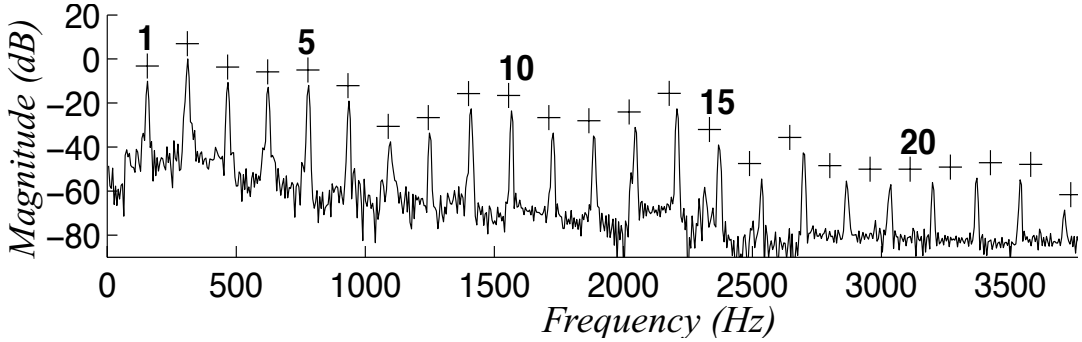

Figure 2.24: Spectrum of a vibrating piano string (F = 156Hz). Ideal harmonic locations are numbered and indicated with "+" marks above the spectrum. The inharmonicity phenomenon (i.e., non-ideal harmonicity) shifts the 24th harmonic partial to the position of the 25th ideal harmonic (from Klapuri (2004a), page 22).

where F is the fundamental frequency, h is the harmonic index (partial number), and β is the inharmonicity factor (Fletcher and Rossing (1998), page 363). The most commonly used inharmonicity factor for the middle pitch range of a piano is $B = 0.0004$, which is also the necessary to shift the 17th partial to the ideal frequency of the 18th partial. Figure 2.24 shows the spectrum of a vibrating piano string with the ideal harmonic frequencies indicated above the spectrum.

3. **Spurious components**: For some instruments, there are some dominant components excited along with the partials. According to Conklin Jr (1999), these **phantom partials** are observed in string instruments and are related to the tension variation on the plucked strings, appearing close to the frequencies of the normal

partials. Phantom partials are sometimes rather dominant compared to the partials.

Non-stationary events - transients

Although the term **transient** does not have a precise definition, it can be stated as an event or zone of short duration where a fast variation of the sound signal occurs (Rodet and Jaillet, 2001). On music signals, transients are identified as note onsets in the form of fast attacks, or at note offsets with fast releases. Due to its highly non-stationary nature, F0 estimation within transients is very hard to tackle. In some cases, as it happens on bowed and woodwind instruments, the attack transient also excites subharmonics (McIntyre et al., 1983).

Transients are often impulsive and accompanied with high energy, resulting in many spurious components that will most likely interfere with other sound sources. Recent research tends to treat the transient as a specific signal component, where the transient is detected by either a non-parametric approach (Rodet and Jaillet, 2001; Röbel, 2003; Bello et al., 2005), or a parametric approach (Daudet, 2004; Molla and Torrésani, 2004).

Reverberation

Reverberation prolongs preceding sounds in a way that they overlap with the subsequent sounds. This way, a recorded signal becomes a mixture of direct sounds, reflected sounds and also reverberated sounds. Thus, even when a monodic instrument is being recorded in a reverberant environment, the recorded signal can be polyphonic (Beauchamp et al., 2012; Baskind and de Cheveigné, 2012; Yeh et al., 2006). Furthermore, the reverberated parts are quite non-stationary, which adds even more complexity to the analysis of the recorded signal.

2.7 Summary

This chapter presented a brief explanation of several background topics, from waves and signal sampling to more advanced digital signal processing. Then, relevant terminology was defined and, finally, a theoretical background addressing the frequency estimation of acoustic signals was presented.

The next chapter will review how the presented concepts were extended by several researchers to address the multi-pitch estimation and multiple-F0 estimation problems.

Chapter 3

Related Work

This chapter presents a literature review of previous studies of multiple-F0 estimation. Several related studies (i.e. automatic music transcription, audio source separation) are also included in the review because these processes also demand F0 extraction. The reviewed studies are also discussed.

3.1 Multiple-F0 estimation

Multiple fundamental frequency (F0) estimation was initiated by Shields (1970) in his work on separating co-channel speech signals. Afterwards, the research of multiple-F0 estimation was extended to polyphonic pitch estimation in the context of automatic music transcription for polyphonic music signals by Moorer Moorer (1977) and Piszcalski & Galler Piszczalski and Galler (1977). In general, multi-pitch estimation algorithms assume that there can exist more than one harmonic source in the same short-time signal.

Although the methods for monophonic transcription or single-F0 estimation can be categorized according to time or frequency, existing methods for multiple-F0 estimation, as pointed out by Klapuri (2004a), are difficult to classify using a single taxonomy, as they are complex and usually combine several processing principles. For instance, multiple-F0 estimation systems can be categorized according to their mid-level representation (time-domain, STFT, wavelets, auditory filter banks, etc.), but also to their scope (some methods need a-priori information about the instrument to be transcribed, whereas others can be used to analyze generic harmonic sounds), to their capability for modeling varying timbres (for instance, statistical parametric methods can model varying time-frequency envelopes like those of sax sounds, whereas non-parametric methods can only handle fixed spectral patterns, like piano sounds), or by the way they can estimate the interactions between sources (iterative and joint estimation methods).

In this dissertation, we follow the categorization proposed by Yeh (2008), page 23, i.e. how different approaches estimate interactions between different sound sources: iterative estimation and joint estimation. The reason for this categorization relies on the complexity of the problem. There is a compromise between the efficiency and robustness

of each proposed algorithm. Theoretically, joint estimation approaches should handle the source interaction better than iterative estimation approaches. However, joint estimation approaches require greater computational cost. On the other hand, iterative estimation approaches have the advantage of having higher computational efficiency but are less optimal in the source interaction handling.

3.2 Iterative Estimation Approaches

Iterative estimation algorithms iterate F0 estimation of the predominant sources and the respective cancellation or suppression until a termination criteria is achieved. In general, these approaches assume that, during each iteration, there is at least one dominant source with distinct harmonic energy such that the extraction of one single F0 is reliable even if the remaining partials are fragmentary.

3.2.1 Direct Cancellation

Direct cancellation applies a single-F0 estimation algorithm to extract the predominant-F0 and then eliminates **all** harmonics of the extracted source from the input signal. Here, an assumption is made: that a complete removal of the dominant source does not affect the subsequent estimations. The term "direct" cancellation means that the partials related to the estimated F0 are completely removed before the next single-F0 estimation. This way, source interaction such as overlapping partials is not considered.

Parsons (1976) used the Schoroeder's histogram to extract predominant-F0s in a two-speaker separation problem: when the first F0 was estimated, the spectral peaks corresponding to its harmonics were removed before the calculation of the next histogram. Lea (1970) iteratively extracted the predominant peak in SACF as an F0 and canceled the estimate in the ACF array. De Cheveigné (1993) proposed a time-domain cancellation model where he performed a study on both iterative and joint cancellation. His iterative cancellation approach estimates the predominant F0 by AMDF and cancels it by comb filtering. Direct cancellation was also applied in the spectral domain. Ortiz-Berenguer et al. (2005) used spectral patterns obtained from previously recorded piano sounds to perform harmonic matching. Here, predominant-F0s were canceled iteratively using binary masks around the matched harmonics in the observed spectrum.

3.2.2 Cancellation by Spectral Models

An iterative estimation algorithm based on harmonicity and spectral smoothness was proposed by Klapuri (2003). In this approach, the observed signal is preprocessed by a RASTA-like technique (Hermansky and Morgan, 1994) on a logarithmic frequency scale, compressing the spectral magnitudes and removing the additive noise. The resulting spectrum is analyzed into a 2/3 octave filter bank, constraining the minimum bandwidth to 100Hz at the lowest bands. F0 weights are computed on each band according to the normalized sum of their partial amplitudes and are further combined by summing the

squared band-wise weights, taking inharmonicity into account. The spectral components of the fundamental frequencies that have the highest global weights are smoothed using the algorithm described in Klapuri (2001) before their subtraction from the mixture to avoid its corruption after several iterations of direct cancellation. Therefore, overlapping partials still remain in the residual after subtraction. The method, called the **bandwise smooth model**, uses the moving average over the amplitudes of the harmonic partials within one octave band to smooth out the envelope of an extracted source. After the smoothing process, the weights of each candidate are calculated again and the highest recalculated global weight determines the resulting F0. This process terminates when the maximum weight related to the *signal-to-noise ratio* (SNR) is below a fixed threshold.

A perceptually motivated multiple-F0 estimation method using an auditory filter bank was later presented by Klapuri (2005). Here, the signal at each subband is compressed, half-wave rectified and low-pass filtered. Then, similarly to the SACF, the results are combined across channels, but in this method magnitude spectra are summed across channels to obtain a summary spectrum. The predominant F0 is computed using an approximated $\frac{\mathbf{1}}{\mathbf{h}}$**smooth model**[1], to remove the source from the mixture while keeping the energy of higher partials for the next iteration. Klapuri (2006) also proposed a spectral model, where the spectral signal is flattened (whitened) to reduce timbral-dependant information, as an attempt to generalize a variety of musical instrument sounds. Here, the salience for each F0 candidate is computed as a $\frac{1}{h}$ wighted sum of its partials. The same weighting scheme is also performed by Klapuri (2008) using a computationally efficient auditory model.

The system from Klapuri (2005) was later used as a front-end by Ryynänen and Klapuri (2005) to incorporate the multiple-F0 estimation algorithm into a probabilistic framework, as in Ryynnen and Klapuri (2004) for singing transcription. Here, the transcription system applies three probabilistic models: a note event **Hidden Markov Model** (HMM), a silence model and musicological model. The note HMM uses the output of the multiple-F0 estimator (Klapuri, 2005) to calculate the likelihoods or different notes, the silence model identifies time regions where no notes are sounding and the musicological model controls transitions between the other two models (note HMMs and the silence model). As in Ryynnen and Klapuri (2004), the acoustical and musicological models are combined into a network whose optimal path is found using the token-passing algorithm from Young et al. (1989).

Bach and Jordan (2005) addressed the problem of multiple pitch tracking using factorial HMM and graphical models (Jordan, 2004). The spectral model is trained from a speech database (Plante, 1995) as a **spline smoothing model**. The predominant-F0 is obtained by maximizing the likelihood. Predominant-F0 tracking and subtraction are iterated until the designated number of F0s is achieved.

[1]Partial amplitudes are inversely proportional to the partial index.

3.2.3 Matching Pursuit

Matching Pursuit (MP) is a numerical technique for decomposing a signal into linear functions (or atoms) that are selected from a dictionary. Given a fixed dictionary, MP will first find the atom that has the biggest inner product with the signal, and then subtract the contribution of that atom from the signal. This process is repeated until the signal is satisfactorily decomposed.

Mallat and Zhang (1993) proposed an MP algorithm to approximate a solution for decomposing a signal into linear functions that are selected from the dictionary. On the first iteration, the atom which best correlates with the analyzed signal is chosen. Then, the contribution of this function is subtracted from the signal and the process is repeated on the residual. This way, the algorithm minimizes the residual energy by choosing at each iteration the most correlated atom with the residual. As a result, the signal is represented as a weighted sum of atoms of the dictionary plus a residual.

Gribonval and Bacry (2003) extended MP into **harmonic matching pursuit** (HMP) by employing a dictionary composed by harmonic atoms. This way, a Gabor atom[2] is identified as a partial and a spectral pattern (or harmonic atom) as a linear combination of Gabor atoms. Cañadas-Quesada et al. (2008) extended HMP to avoid the corruption of the residual when there is partial overlap by maximizing the smoothness of the spectral envelope for each harmonic atom. The smoothing process is similar to the one in Klapuri (2003). Reyes et al. (2009) performed a study on this method performance when dealing with harmonic related simultaneous notes.

Leveau et al. (2008) proposed a modified MP algorithm that is applied to the whole signal at once. This way, instead of the frame by frame analysis, harmonic molecules are considered as a group of several atoms of the same instrument in consecutive time windows.

3.3 Joint Estimation

While on iterative estimation approaches the predominant-F0 is estimated, followed by its cancellation or suppression, joint estimation approaches evaluate possible combinations of multiple-F0 hypothesis without any cancellation involved (except for joint cancellation). Despite the fact that the input signal is not corrupted as in iterative estimation and the resulting cancellation process, the handling of overlapping partials still remains a difficult task.

3.3.1 Joint Cancellation

Joint cancellation, besides joint estimation, also performs the suppression or cancellation of the estimated pitches. As in iterative estimation, the estimation and cancellation happen in an iterative manner, until a termination criteria is met.

[2]Gabor atoms are time-frequency atomic signal decompositions proposed by Gabor (1946, 1947). These are obtained by dilating, translating and modulating a mother generating function.

Double Diference Function

De Cheveigné (1993) proposed a method that uses the **double difference function** (DDF) that jointly cancels multiple-F0 hypotheses. These hypotheses are canceled using a cascade of filters, and the selected combination is the one that minimizes the residual. In the experiments done by de Cheveigné (2005), different iterative cancellation methods are compared with the joint approach, showing that joint cancellation performs better than iterative cancellation because a single-F0 estimation failure will lead to successive errors in an iterative manner. However, joint cancellation is computationally more demanding than iterative cancellation.

Two-way Mismatch

Maher and Beauchamp (1993) proposed a **two-way mismatch** (TWM) method to estimate two F0s jointly. TWM searches for the pair of F0s that minimize the frequency discrepancies between the predicted partials and the measured peaks on both ways: the mismatch from "*the predicted to the measured*" and the mismatch from "*the measured to the predicted*". Each match is weighted by the amplitude of the corresponding measured peak. The algorithm minimizes the residual by the best match.

3.3.2 Polyphonic Salience Function

Salience methods try to emphasize the underlying F0s by applying signal processing transformations to the input signal to ease a later peak-picking or tracking.

Licklider (1951) proposed a pitch perception model consisting of an array of bandpass filters for cochlear filtering, followed by autocorrelation. According to Lyon (1984), this model leads to the channel-lag representation of ACF in the auditory models and is often referred as **correlogram** (Slaney et al., 1990). Several salience functions follow this model. Weintraub (1986) and, later, Wu et al. (2003) presented a similar method for separating the speech signals of two simultaneous talkers by applying iterative dynamic programming to the correlogram and a Markov model to determine the characteristics of each speaker's voice. Tolonen and Karjalainen (2000) performed a two-channel SACF over the observed signal using an auditory filter bank. Here, the SACF is processed to remove peaks corresponding to harmonics and subharmonics. The resulting function is called **enhanced summary autocorrelation function** (ESACF).

Polyphonic salience functions can also be obtained by employing a combination of single-F0 estimation functions. ACF, along with AMDF were combined by Min et al. (1988) as the salience function and Peeters (2006) combined a temporal domain function (real cepstrum) with a spectral domain function (ACFS - spectral autocorrelation). This way, since temporal domain functions are prone to subharmonic errors and spectral domain functions are prone to super-harmonic errors, by combining both, Peeters (2006) provided a useful polyphonic salience function that reduces the octave ambiguities.

Zhou (2006) presented a method to extract the power spectrum above the noise floor, called **resonator time-frequency image** (RTFI) (Zhou and Mattavelli, 2007), from which **relative energy spectrum** and **relative pitch energy spectrum** are derived

for the selection of F0 candidates. RTFI selects a first order complex resonator filter bank to implement a frequency dependent time-frequency analysis, due to the flexibility with regards to time and frequency resolution, and the simplicity and computational efficiency of an implementation based on first-order filters. Then, the RTFI Average Energy Spectrum is used as input and transformed into Relative Energy Spectrum and the harmonic components are extracted. Afterwards, for a preliminary estimate of the possible multiple pitches, the RTFI Energy Spectrum is converted into a Relative Pitch Energy Spectrum. The information about harmonic components and pitch candidates is then combined to cancel pitch candidates without *enough confidence* (Zhou (2006), page 114). Finally, a spectral smoothing principle is used to validate each remaining pitch candidate.

3.3.3 Spectral Matching by Non-parametric Models

Non-parametric models or *static models* assume that the spectral pattern of a harmonic structure is *fixed*. Hence, these methods can only handle fixed spectral patterns like, for instance, piano sounds. On the other hand, these models do not cope well with instruments that do not have a fixed spectral profile, like the saxophone. In general, the sound of real saxophones contains varying dynamics and expressive alterations, like breathing noise, causing several notes with same pitch to sound differently.

Sparseness

Sparse approximation or sparse decomposition accounts for most or all information of a signal with a linear combination of a small number of elementary signals called **atoms**, chosen from a **dictionary**. The technique of finding a representation with a small number of significant coefficients is often referred to as **sparse coding**. Decoding merely requires the summation of the relevant atoms, appropriately weighted.

Consider the observed signal x, where $x = D\alpha$. D is an $m \times p$ matrix ($m \ll p$) and $x \in \mathbb{R}^m, \alpha \in \mathbb{R}^p$. D is the *dictionary* or the design matrix. The idea is to estimate the signal α, subject to the constraint that it is sparse. **Sparsity** implies that only a few components of x are non-zero and the rest are zero. This implies that x can be decomposed as a linear combination of only a few $m \times 1$ vectors in D, called *atoms*: the basis of x. The sparse decomposition problem is represented as:

$$\min_{\alpha \in \mathbb{R}^p} ||\alpha||_0 \quad \text{such that} \quad x = D\alpha, \tag{3.1}$$

where $||\alpha||_0 = \#\{i : \alpha_i \neq 0, i = 1, \cdots, p\}$ is a pseudo-norm, l_0, which counts the number of non-zero components of $\alpha = [\alpha_1, \cdots, \alpha_p]^T$. This problem is NP-Hard with a reduction to NP-complete subset selection problems in combinatorial optimization (NP-Hard, NP-complete and combinatorial optimization problems will be discussed in the next chapter).

A convex relaxation of the problem can instead be obtained by taking the l_1 norm instead of the l_0 norm, where

$$||\alpha||_1 = \sum_{i=1}^{p} |\alpha_i|. \tag{3.2}$$

The l_1 norm induces sparsity under certain conditions (Donoho, 2006).

Bello et al. (2002) proposed a time-domain approach for piano transcription using as the dictionary a database of waveforms. This method uses linear algebra to decompose the original signal into a sum of signals present in the database. One particular feature of this approach is that the waveforms in the database can be known a priori, but they can also be inferred from the observed signal. The adaptive dictionary is built using a simplified version of the fundamental frequency estimator of the **blackboard method** described in Bello and Sandler (2000). For each estimated pitch, its spectral information is resynthesized using the inverse Fourier transform into an audio signal corresponding to that pitch. Then the signals missing in the dictionary database are created from the obtained signals using pitch-shifting by standard phase-vocoder techniques. Once the database is complete, the time-domain method can estimate the pitches.

Groble (2008) presented a system where each pitch has a model associated with it. The model for each pitch is processed a-priori by computing scaled averages of stored audio samples, i.e. the model for each particular pitch is generated from a large number of sample frames taken from an instrument playing that note. This training data contains samples at multiple note attack levels and multiple distances between the note onset and the frame start. The model for each pitch is processed by normalizing each individual feature vector at that pitch to have a unit height, averaging the unit height vectors together, and normalizing the resulting average vector to have unit length. The feature vectors extracted from the audio frames of the observed signal are scored against these pitch models to estimate which pitches are present in the frames.

Lee et al. (2010) assume that the Fourier coefficients of an input frame are a linear combination of the Fourier coefficients of previously recorded waveforms of individual piano notes (Bello et al., 2002) and defined a matrix of Fourier coefficients of segments of those waveforms as the dictionary. The computational complexity of the problem is reduced by l_1 minimization.

Benetos and Dixon (2011b) proposed a model that extends the **shift-invariant probabilistic latent component** (PLCA) method of Smaragdis et al. (2008). This model is able to support the use of multiple pitch templates extracted from multiple sources. Using a log-frequency representation and frequency shifting, detection of notes that are non-ideally tuned, or that are produced by instruments that exhibit frequency modulations is made possible. Sparsity is also enforced in the model, in order to further constrain the transcription result and the instrument contribution in the production of pitches. Finally, a hidden Markov model-based note tracking method is employed in order to provide a smooth piano-roll transcription.

Non-negative Matrix Factorization

Non-negative Matrix Factorization (NMF) is a technique for data analysis where a non-negative matrix $\mathbf{V}$ is factorized into two non-negative matrices $\mathbf{W}$ and $\mathbf{H}$: given an $n \times m$ non-negative matrix $\mathbf{V}$ and a positive integer $r < \min(n, m)$, NMF tries to factorize $\mathbf{V}$ into an $n \times r$ non-negative matrix $\mathbf{W}$ and an $r \times m$ non-negative matrix $\mathbf{H}$ such that:

$$\mathbf{V} \approx \mathbf{WH}. \tag{3.3}$$

This way, the observed power spectra $\mathbf{V}$ can be decomposed into spectral models (basis functions in $\mathbf{H}$) of each note with its intensity change along time (weightings in $\mathbf{W}$) as in Smaragdis and Brown (2003). The cost function is designed to favor the minimization of the residual with specific constraints like **sparseness** (Cont, 2006a), **harmonicity** (Raczyński et al., 2007), beta-divergence (Dessein et al., 2010a) and **ERB-scale time-frequency** representation (Vincent et al., 2007). Although fast algorithms have been proposed for multiple-F0 estimation (Sha and Saul, 2004; Cont, 2006a), the challenge remains in the modelling of the time-varying spectra of sound sources (Virtanen, 2003b; Abdallah and Plumbley, 2004). NMF methods have also been used for music transcription by Plumbley et al. (2002), Virtanen (2007), Vincent et al. (2007), Bertin et al. (2007).

Independent Component Analysis

The **independent component analysis** (ICA) (Jutten and Herault, 1991; Comon, 1994) finds the independent components (factors) of a signal by maximizing the statistical independence of the estimated components. The signal model is factorized as $\mathbf{x} = \mathbf{Wh}$, being $\mathbf{x}$ and $\mathbf{h}$ n-dimensional real vectors, and $\mathbf{W}$ a non-singular matrix. Citing Virtanen (2006), page 274: *"ICA attempts to separate sources by identifying latent signals that are maximally independent. In practice, this usually leads to the separation of meaningful sound sources."*.

ICA is closely related to NMF and, as pointed by Schmidt and Larsen (2008), their differences rely on the different constraints placed on the factorizing matrices. In ICA, rows of $\mathbf{W}$ are maximally statistically independent, whereas in NMF all elements of $\mathbf{W}$ and $\mathbf{H}$ are non-negative. Both ICA and NMF have been investigated by several authors (Plumbley et al., 2002; Abdallah and Plumbley, 2003, 2004; Virtanen, 2007) for polyphonic transcription. According to Virtanen (2007), for spectrogram factorization, NMF algorithms performed better separation results than ICA.

ICA has also been successfully used for drum transcription (FitzGerald, 2004; Virtanen, 2006), since most percussive sounds have a fixed spectral profile and, thus, can be modeled using a single component.

Neural Networks

The method from Marolt (Marolt, 2004a,b) uses a connectionist approach for automatic transcription of polyphonic piano music. The first stage of the partial tracking algorithm transforms the acoustical waveform into time-frequency space using an auditory

model, which emulates the functionality of human ear. The auditory model consists of two parts: an auditory Patterson-Holdsworth gammatone filterbank (Patterson and Holdsworth, 1996) and the Meddis hair cell model (Meddis, 1986). The filterbank is first used to split the signal into several frequency channels, modeling the movement of basilar membrane in the inner ear and the output of each gammatone filter is processed by the Meddis' model of hair cell transduction. Then, the auditory model outputs a set of frequency channels containing quasi-periodic firing activities of inner hair cells. Instead of using a correlogram, a modified version of the Large and Kolen (1994) adaptive oscillators is employed to detect periodicities in the output channels of the auditory model. If the oscillators synchronize with their stimuli (outputs of the auditory model), then the stimuli are periodic, meaning that the respective partials are present in the input signal. This scheme is able to track partials, even in the presence of vibrato or beating. This model for tracking individual partials was extended into a model for tracking groups of harmonically related partials. This was done by joining adaptive oscillators into networks. Each network consisted of up to 10 interconnected oscillators with their initial frequencies set to integer multiples of the frequency of the first oscillator. A total of 88 oscillator networks, with initial frequencies corresponding to all 88 piano tones (A0-C8) were created. Then, a set of 76 neural networks is employed to perform note recognition. The inputs of each network are taken from the outputs of the oscillator networks, combined with the outputs from the auditory model. Each of the 76 neural network is trained to recognized one piano note (from A1 to C8). The used network model was the **time delay neural network** (TDNN) (Waibel et al., 1989). This method was further integrated into a system called SONIC[3] to transcribe piano music. This system also includes an onset detector, a module for detecting repeated notes and simple algorithms for length and loudness estimation. SONIC is constrained to piano transcription since the system was trained by using only piano sounds.

Support Vector Machines

Like Neural Networks, **Support Vector Machines** (SVMs, also support vector networks Cortes and Vapnik (1995)) are supervised learning models with associated learning algorithms that can be used for: data analysis, pattern recognition, classification and regression analysis.

The method proposed by Poliner and Ellis (2007b,a) employs SVMs on a frame-by-frame spectral analysis and then performs **Hidden Markov Models** (HMM) to refine the estimated pitches. SVMs were previously trained with spectrograms of synthesized MIDI notes. The SVMs consist in a total of 87 note classifiers trained to detect the presence of the corresponding musical note in a given time frame. The input time frames consist on a 255 feature vector consisting of normalized spectral bins. The output of the SVM classifier (referred as *posteriogram*) is fed into a two state (on/off) HMM to improve the temporal coherence of the estimate. The HMM performs temporal smoothing by inferring the temporal structure from the musical signal. This way, the HMM post-processing stage

[3] `http://lgm.fri.uni-lj.si/SONIC/`

is applied to the unlabeled data classification, resulting in class assignment updates (in some cases), due to temporal context. This system is constrained to polyphonic piano transcription and its experimental results outperformed those of Marolt (2004a,b) using Neural Networks.

Zhou (2006) also used SVMs on Method II, employing 88 binary classifiers, each corresponding to one piano note. The classifier inputs are the peaks extracted from the RTFI energy spectrum.

Specmurt Analysis

Specmurt Analysis is a method proposed by Sagayama et al. (2004) that enables "*pitch likelihood distribution visualization*", unlike standard methods for pitch estimation Saito et al. (2008). Specmurt Analysis, like short-time spectrum analysis, is a frame-wise signal processing method. On the contrary to the Cepstral analysis, which is the inverse Fourier Transform of the short-time power spectrum with log-scaled magnitude, it is the inverse Fourier transform of the short-time power spectrum with log-scaled frequency. The deconvolution of the spectrum by a common harmonic pattern is achieved by division. In this method, the input signal is first processed by a **constant-Q** transform (Brown, 1991) so its power spectrum is on the log-frequency scale. The log-frequency spectrum is then transformed into Specmurt by inverse Fourier Transform.

Bispectrum

According to Nikias and Mendel (1993) and Nikias and Raghuveer (1987) the **bispectrum** is a bidimensional frequency representation capable of detecting nonlinear harmonic interactions, which are typically present in musical audio signals. Furthermore, in the presence of interfering sound partials, the bispectrum ideally enables to resolve the contribution of each single harmonic by performing a bidimensional cross-correlation procedure with a fixed 2D-harmonic pattern. The bispectrum belongs to the class of Higher-Order Spectra (HOS, or polyspectra), used to represent the frequency content of a signal. An overview of the theory of HOS can be found in Nikias and Mendel (1993) and Nikias and Raghuveer (1987). The bispectrum is defined to be the third-order spectrum, being the amplitude spectrum and the power spectral density, respectively, the first and second-order ones.

In Nesi et al. (2009a) method, multiple-F0 estimation has been performed, on a frame by frame basis, through a joint constant-Q and bispectral analysis of the input audio signal. The bispectrum was computed using by using Nikias and Mendel (1993) *direct method*. F0-tracking has been obtained by detecting note onsets and durations through the analysis of the signal spectrogram.

Hypothetical Partial Sequence

The method proposed by Yeh (2008); Yeh et al. (2010) is a frame-based system for joint estimation of multiple fundamental frequencies, based on the STFT representation. The first stage of this method consists on applying an adaptive noise level estimation algorithm (Yeh and Röbel, 2006) to provide a probabilistic classification of the spectral peaks

into sinusoids or noise. The sinusoidal peaks are considered the partials of the quasi-periodic sources such that a combination of harmonic patterns related to F0 hypotheses will match (Yeh et al., 2005). To reduce the computational cost of the approach, a set of F0 hypotheses is selected from the spectral peaks using a harmonic matching technique. Each hypothesis is related to its **hypothetical partial sequence** (HPS). The correct combination of HPSs will lead to their partials matching as many sinusoidal peaks as possible in the observed spectrum. This way, the HPS is a source model with estimated frequencies and amplitudes obtained by partial selection and overlapping partial treatment. Partials are identified with spectral peaks within a tolerance deviation from their ideal position. If two or more peaks are within the tolerance range, the peak performing a smoother HPS envelope is selected. Amplitudes of overlapped partials in the combination are estimated by using linear interpolation, as in Maher (1990), and a set of rules (Yeh, 2008). The HPS is flattened by exponential compression. A score function is then used on the estimated HPSs to select the plausible sets of F0 hypotheses. To infer the best combination, hypothetical sources are progressively combined and iteratively verified. A hypothetical source is considered valid if it either explains more energy than the noise, or improves significantly the envelope smoothness once the overlapping partials are treated. The score function for a given hypothesis is computed considering the following aspects, for each hypothetical source: spectral match with low inharmonicity, spectral smoothness and synchronous amplitude evolution of the partials. The score function is a weighted sum of these four criteria, and the weights are optimized using the evolutionary algorithm Schwefel (1995) using a large dataset. The estimation of the number of concurrent sounds is finally done by iterative score improvement (Chang et al., 2008; Yeh et al., 2006), based on the explained energy and the improvement of the spectral smoothness: the polyphony hypothesis is progressively increased and all possible combinations are evaluated. Finally, a post-processing stage can be added by tracking the F0 candidates trajectories, using a high-order HMM and a forward-backward tracking scheme proposed by Chang et al. (2008).

Gaussian Mixture Models

Pertusa and Inesta (2008a) proposed a multiple-F0 estimator using Gaussian smoothness. This method selects a set of fundamental frequency candidates at each time frame, generating all the possible candidate combinations. The combination with highest salience is selected, taking into account the sum of the harmonic amplitudes and the spectral smoothness of its candidates. An interpolation method is introduced in Pertusa and Inesta (2008a) to deal with some overlap situations. Each frame is analyzed, yielding a combination of F0s that maximizes a salience measure. This approach was later extended in Pertusa and Inesta (2008b) to deal with low frequency resolution by adding short context information to get the combination of pitches at each frame: instead of selecting the combination with highest salience at each time frame, short context information is taken into account to get the salience of each combination of pitches, performing a temporal smoothing. For grouping similar information across time, a set of F0 combinations are generated at each time frame. In order to have unique combinations in each time frame,

if more than one combination with the same pitches is found in a single frame, only the combination with the highest salience is kept, removing duplicates with lower saliences. This method has a high computational efficiency, which is the main handicap of joint estimation approaches.

The method from Canadas-Quesada et al. (2009) also assumes that a polyphonic sound can be modeled by a sum of weighted **Gaussian mixture models**. In this approach, an adaptive logarithmic threshold is used to select the possible partials. Then, given a pitch range, the F0 candidates are chosen from the previous selected partials and, for each candidate, a harmonic pattern is built in the log-frequency domain, considering one semitone bandwidth for partial search. All the possible candidate combinations are modeled by a sum of weighted GMMs. The corresponding weights are composed by non-overlapped partials and/or colliding partial magnitudes. Overlapped partial amplitudes are estimated by means of linear interpolation using the nearest neighboring non-overlapped partials, as in the work of Pertusa and Inesta (2008a).

3.3.4 Statistical Modeling using Parametric Models

Contrarily to the non-parametric models, which assume that the spectral pattern of a harmonic structure is fixed, statistical parametric methods can model varying time-frequency envelopes like, for instance, saxophone sounds. Saxophone cannot be considered to have fixed spectral profile, since real saxophone sounds, in general, contain varying dynamics and expressive alterations, like breathing noise, causing several notes with same pitch to sound differently.

Statistical approaches are generally formulated with Bayesian models. Bayesian statistical methods provide a complete paradigm for both statistical inference and decision making under uncertainty.

Waveform Models

According to Davy (2006), page 203: *"tonal music can be exploited to build a Bayesian model, that is, a mathematical model embedded into a probabilistic framework that leads to the simplest model that explains a given waveform"*. These models are also known as **generative models** because they can be used to generate data by changing its parameters and the noise. Some multiple-F0 estimation systems rely on these kind of models for modeling the acoustic waveform. Most of these generative models assume that the F0 belongs to a fixed set of values, associated to the pitches. **Waveform Models** try to adaptively match the observed waveform in the time domain.

The method from Walmsley et al. (1999) segments the observed signal into 20ms audio frames assuming that, during each frame, the signal is stationary. The model is employed as a sum of an unknown number of concurrently sounding notes, the parameters of each source being: the fundamental frequency, number of partials, partial amplitudes and the residual variance. These parameters are estimated jointly across a number of adjacent frames by means of **Markov chain Monte Carlo** (MCMC) method. Davy and Godsill

(2003) extended this method by introducing a prior distribution of the inharmonicity factor.

The generative music signal model proposed by Cemgil et al. (2003, 2006b) introduces a higher-level parameter related to tempo, with several modifications. This method is based on a generative model formulated as a dynamical Bayesian network. This probabilistic model assumes that the partials have harmonic frequency relationships and also an exponentially decaying spectral envelope. This approach allows to write noisy sum-of-sines models into a sequential form. The model relies on damped sinusoids with constant frequency. The piano-roll is inferred by assigning to each of the grid frequencies the state "*on*" or "*off*" at each instant. The algorithm for estimating the most likely piano-roll is based on Greedy Search and Kalman filtering on a sliding window over the audio signal. This technique can be considered as a time-domain method, since the discrete Fourier transform is not being explicitly computed, and, thus, can be used to analyze music to sample precision, although with a very high computational cost.

Spectral Models

While Waveform Models try to adaptively match the observed waveform in the time domain, **Spectral Models** try adaptively to match the observed signal in the frequency domain. The phase information is often discarded.

Vincent and Rodet (2004) proposed a generative model based on nonlinear **Independent Subspace Analysis** (ISA) and factorial HMM. Linear ISA describes the short-time power spectrum of a musical excerpt as a sum of power spectra (or partials) with time-varying weights, where the modeling error is a Gaussian noise. In order to overcome several known limitations of linear ISA (Plumbley et al., 2002; Eronen, 2003; Eggink and Brown, 2003; Mitianoudis and Davies, 2002) a nonlinear ISA model was derived considering both summation of power spectra and of log-power spectra. This method is based on creating specific instrument models by learning its parameters on solo excerpts and use them to transcribe polyphonic excerpts. The instruments model is defined as a collection of parameters: the means and variances of partial amplitudes, partial frequencies and residuals. To transcribe a signal, the spectrum is considered as a sum of spectral models whose weights are optimized using an approximation to the second order Newton method. The HMM is used for adding temporal continuity and modeling note duration priors.

Goto (2000) proposed the **PreFEst** method to detect melody and bass lines in musical signals. This system assumes that the melody line has the most predominant harmonic structures in middle and high frequency regions and the bass line has the most predominant harmonic structure in a low frequency region. It also assumed that melody and bass lines tend to have temporally continuous trajectories. The first stage consists on calculating the instantaneous frequencies by using a multirate filterbank and extracts candidate frequency components on the basis of an instantaneous-frequency-related measure. Then, two bandpass filters are used to separate the spectral components of the bass and melody lines. For each set of filtered frequency components, a **probability density function** (PDF) of the F0 is formed, representing the relative dominance of every possible harmonic structure. The observed PDF is considered as being generated by a weighted mixture of

harmonic-structure tone models. These model parameters are estimated using the EM algorithm. Finally, to consider a continuity of the F0 estimate, the most dominant and stable F0 trajectory is selected, by tracking the peak trajectories in the temporal transition of the F0's PDFs. This step is achieved by multiple agents that track the temporal trajectories of salient promising peaks in the F0's PDF and the output F0 is determined on the basis of the most dominant and stable trajectory. This system works in real time.

The method proposed by Vincent (2004) models the observed spectrum according to the means and variances of partial amplitudes, frequencies and residuals. This way, the spectrum can be interpreted as a sum of the spectral models with the related weighting optimized by Newton's method. A factorial model is also applied for constraining the temporal continuity and for adapting the duration.

Kameoka et al. (2005a) formulates the multi-pitch estimation problem as a time-space clustering of harmonic sounds, which is named **harmonic temporal clustering** (HTC) method (Kameoka et al., 2007). This approach decomposes the power spectrum time series into distinct clusters corresponding to single sources. This way, HTC method models the pitch, the relative partial amplitudes, the intensity, the onset and the duration of each underlying source. The input of the system is the observed power spectrum with log-frequency scale. The HTC model assumes that all sources are periodic signals having smooth power envelopes with decaying partial amplitudes. Using this model, a goodness of the partitioned cluster is calculated using the Kullback-Liebler (KL) divergence. The model parameters are estimated using the expectation-constrained maximization (ECM) (Meng and Rubin, 1993; Kameoka et al., 2005b). Partials of a harmonic source are modeled as Gaussian distributions with initial spectral envelopes. The evolution of partial amplitudes is modeled by Gaussian mixtures across frames such that the synchronous evolution is constrained and the duration is adaptively modeled. In the evaluation done by Kameoka et al. (2007), the HTC system outperformed the PreFEst results.

The approach proposed by Li and Wang (2007) is somehow related to Ryynänen and Klapuri (2005) in the sense that the preliminary pitch estimate and the musical pitch probability transitions are integrated into a HMM. Nevertheless, for pitch estimation, this method uses statistical tone models that characterize the spectral shapes of the instruments. Instrument models are built using Kernel density estimation. This method is intended for single instrument transcription only.

The method introduced by Emiya et al. (2008a) consists in applying the onset detector from (Alonso et al., 2005) on the observed signal and, for each segment between two consecutive onsets, a set of candidates is selected using the **probabilistic spectral smoothness principle** (Emiya et al., 2010). The most likely combination of pitches within each segment is selected using an HMM, embedding a spectral maximum likelihood. Notes with the same pitch in two consecutive segments are merged if they are still present at the end of the first segment and if the increase of an energy criterion is below a predefined threshold. Apart from that, notes are considered as repeated.

The method proposed by Duan et al. (2009a) models spectral peaks and non-peak regions. Given pitch estimates for individual frames, pitch trajectory formation is cast as a constrained clustering problem, where each cluster corresponds to a trajectory. Har-

monic structure is used as the feature of each pitch in clustering. Finally, note formation happens after pitch trajectories are formed, instead of forming notes and then placing them in streams. The problem is addressed in two stages: the first stage is multi-pitch estimation, where pitches and polyphony in each frame are estimated, and then refined using estimates in neighboring frames; the second stage is pitch trajectory formation, where initial pitch trajectories are formed by grouping pitch estimates across frames according to pitch height. Within each initial trajectory, pitch estimates that are close in frequency and contiguous in time are grouped to form *notelets*. Final pitch trajectories are obtained through constrained clustering of pitch estimates, where must-link constraints are imposed on pitch pairs in each *notelet* and cannot-link constraints are imposed on pitch pairs of concurrent *notelets*. From the view of Auditory Scene Analysis, the first stage is simultaneous grouping and the second stage is sequential grouping.

3.3.5 Blackboard Systems

Blackboard Systems (Engelmore and Morgan, 1988) are artificial intelligence applications based on the blackboard architectural model, where a common knowledge base - the "blackboard" - is iteratively updated by a group of specialists or knowledge sources that start by specifying the problem into the blackboard and end with a solution. Each specialist updates the blackboard with a partial solution when its internal constraints match the blackboard state, thus, working together to solve the problem. As a metaphor, one can say that there is a group of specialists, seated in a room with a large blackboard. These specialists are working as a team to brainstorm a solution to a problem, using the blackboard as the workplace to cooperatively develop the solution. The session begins when the problem specifications are written into the blackboard. Then, the specialists watch the blackboard, looking for an opportunity to apply their expertise into the development of a solution. When someone writes something on the blackboard, that allows another specialist to apply their expertise, the second specialist records their contribution on the blackboard, hopefully enabling other specialists to apply their expertise as well. This process continues until the problem is solved.

Blackboard systems consist of three major components: the knowledge sources, the blackboard and control shell. The knowledge sources typically consist on a set of rules, the blackboard is a shared repository of problems, partial solutions, suggestions and contributed information. The blackboard can also be thought as a dynamic "library" of contributions to the current problem that have been recently "published" by other knowledge sources. The control shell, or the scheduler, determines the order in which knowledge sources are allowed to act. The system converges when the knowledge sources are satisfied with the hypothesis in the blackboard, given an error margin.

In general, the blackboard architectures used in music transcription are similar to the one proposed by Martin (1996). The blackboard hierarchy is ordered by increasing abstraction, being the observed signal at the lowest level: power spectrum, tracks, partials, notes, intervals, chords and tonality).

Bello and Sandler (2000) extended Martin (1996) approach by including top-down processing by means of a Neural Network to detect the presence or absence of a chord. The

output of the Neural Network is fed into the blackboard system reshaping the *hypothesis matrix*, changing its structure and adding a new level of information to the system: chords. This way, the knowledge sources that interact with the mentioned matrix are structurally modified and urged to link strong note hypotheses present in the blackboard to produce hypothetic chords.

Other blackboard systems have been also proposed by Ellis (1996); Monti and Sandler (2002); Plumbley et al. (2002). For a more detailed review on these methods please refer to McKay (2003).

3.4 Discussion

Multiple-F0 estimation is a very complex task and although there are many and different approaches, as we reviewed during this chapter, it is a problem that still remains unsolved. Moreover, there is no method suitable for all the variety of musical sounds. The advantages and drawbacks among the different methods previously described are discussed in this section.

3.4.1 Spectral Representation: Multi-resolution or Fixed-resolution?

Most multiple-F0 estimation and multi-pitch estimation methods involve analysis in the frequency domain. This representation has the advantage of providing an intuitive representation of the harmonic structure and the spectral envelope of a sound source, which eases the modeling of sound sources. Spectral representations can either be multi-resolution or fixed-resolution.

Multi-resolution representations, like constant-Q transform, wavelets or filter-banks, have the advantage of representing the signal with different resolutions for distinct frequency bands. Although several authors claim that by using a multi-resolution spectral representation a benefit results from its similarity to the equal tempered music scale or the human auditory system (Chafe and Jaffe, 1986; Keren et al., 1998; Fernandez-Cid, 1998; Chien and Jeng, 2002; Kobzantsev et al., 2005; Sagayama et al., 2004; Patterson and Holdsworth, 1996; Wu et al., 2003; Marolt, 2004a; Klapuri, 2005), there are still no physical reasons why multi-resolution is better for representing the structure of a harmonic sound. As discussed by Hainsworth (2003) and Yeh (2008), page 27, multi-resolution representations do not really solve the time-frequency main issue and, as a matter of fact, the advantage of multi-resolution also happens to be its disadvantage: wavelets sacrifice frequency resolution at high frequencies, which can be a major drawback for distinguishing individual partials of concurrent sources, and constant-Q and wavelets loose temporal precision in the lower frequencies. For these reasons most of the multiple-F0 and multi-pitch estimation methods still rely on the fixed-resolution representation. In this dissertation, the STFT is chosen for signal representation, despite being criticized for its fixed-resolution.

3.4.2 Computational Efficiency or Greater Accuracy: Iterative or Joint Estimation?

Acoustic signals containing more than one concurrent source are said to be polyphonic. The main issue when dealing with polyphonic signals is that the components of concurrent sources may overlap. In general, iterative estimation algorithms attenuate the predominant source at each iteration. Since the F0s that will be extracted in future iterations are unknown upon the attenuation or cancellation time, it is almost impossible to estimate which partials overlap and to prevent over-attenuation of partials.

On the other hand, joint estimation methods have the advantage of inferring which partials overlap from a set of hypothetical sources. This way, a better treatment of overlapping partials can, thus, be expected. Nevertheless, the downside of the joint estimation approach is its computational cost because the number of hypothetical combination grows exponentially with the polyphony hypothesis.

Although there is no significant proof that joint estimation has better results than iterative estimation, theoretically joint estimation is better since it can handle partial collision. This way, we decided to adopt a joint estimation approach.

3.4.3 Which Joint Estimation Method?

Joint cancellation approaches perform better than iterative cancellation (de Cheveigné, 2005) because a single-F0 estimation failure will lead to successive errors in an iterative manner. However, joint cancellation is computationally more demanding than iterative cancellation and since it also has both iterative estimation and cancellation stages, it has the same disadvantages as the iterative estimation approaches.

Polyphonic salience functions are computationally efficient and they also provide a mid-level representation, which is useful for F0 estimation. Some of these functions are just enhanced representations which need a posterior methodology (e.g. peak picking) to estimate the fundamental frequencies. The main disadvantage of these salience functions is that, in some situations, they can lose relevant information or even produce spurious components in the signal transformation process (Pertusa (2010), page 76). As an example, SACF performs half wave rectification. The output spectrum of a half wave rectified signal (see Klapuri (2004a), page 38) consists of a DC-component, of the original power spectrum scaled down by four, and of a convolution of the original power spectrum by itself. As a matter of fact, the convolution of a spectrum by itself produces spectral components centered at the locations that are multiple of the original spectral intervals, emphasizing partial beating, which might be very useful for single-F0 estimation. However, for polyphonic signals, all the intervals between partials of different sources (and also between different fundamental frequencies) generate prominent components at beating frequencies in the half-wave rectified spectrum, adding spurious data for the subsequent analysis.

Non-parametric models or static models assume that the spectral pattern of a harmonic structure is fixed. These methods can only handle fixed spectral patterns like, for instance, piano sounds. Despite these methods have been successfully applied considering

specific instrument transcription (usually piano). Modeling harmonic sounds with varying harmonic components still remains a challenge (Abdallah and Plumbley, 2004).

Statistical parametric methods can model varying time-frequency envelopes like, for instance, saxophone sounds. However some of those methods, like the Bayesian waveform models, although they provide an elegant way for modeling the acoustic signal, they are mathematically complex and have high computational cost. Statistical spectral models, despite being also complex, are computationally efficient.

Blackboard systems are, to some extent, general architectures. This means that these systems depend on other methods to use them as knowledge sources (Martin, 1996; Bello and Sandler, 2000).

Among all these previously stated approaches, the non-parametric methods are the less complex. Sparse coding is a non-parametric technique for finding a representation of the observed signal as a linear combination of elementary signals called atoms. This can also be seen as a combinatorial optimization problem where the goal is to find the combination of atoms that best resembles the observed signal. Although this approach is limited to fixed spectral patterns, this can be extended by employing an adaptive mechanism in the atoms. By extending sparse coding with Evolutionary Algorithms, namely Genetic Algorithms, which are adaptive by nature, the harmonic structure of each atom can adapt so that their linear combination best fits the observed signal. This way, by using sparse approximation, along with evolutionary algorithms, our non-parametric method will be able to cope with harmonic sounds with varying harmonic components.

Chapter 4

Genetic Algorithms

To describe what a Genetic Algorithm is, one must explain first what an Algorithm is. This chapter presents the concept of Algorithm and its main purpose: problem solving. Several types of problems are presented: decision, search, counting and optimization and it is discussed how search, counting and optimization problems can be viewed as decision problems. Several classes of decision problems are then presented, according to their computational complexity: P, NP, NP-Complete and NP-Hard. Bio-inspired Algorithms are introduced as a mean of addressing NP-Complete and NP-Hard problems. Finally, Genetic Algorithms are presented. Both biological background behind a Genetic Algorithm and its terminology are explained.

4.1 Algorithm

There are several definitions of the term **algorithm**: "A mathematical rule or procedure for solving a problem." (Jančařík, 2007); "A set of well-defined rules or procedures for solving a problem in a finite number of steps." (Lemon et al., 2007) or even "A finite and sorted set of operations to achieve a solution of one problem." (Gersting and Schneider, 1995). Nevertheless, all definitions state that the goal of the algorithm is to solve a given problem, usually with the requirement that the procedure terminates at some point.

The process of applying an algorithm to an input in order to obtain an output its called a **computation**.

4.2 Problem

A problem is an exercise whose solution is desired. However, theoretical computer science defines the term **computational problem** as a mathematical object representing *a collection of questions that computers might solve.* As an example, the problem of factoring *"Given a positive integer n, find a non-trivial prime factor p of n."* is a computational problem.

A computational problem can also be viewed as an infinite collection of instances, together with a solution for every instance. For example, in the factoring problem, the instances are the integers n, and solutions are prime numbers p that describe non-trivial prime factors of n. Both instances and solutions are conventionally represented as binary strings, namely elements of $\{0,1\}^*$. That is: numbers can be represented as binary strings using the binary encoding[1]. During this chapter, for improved readability, numbers will be identified instead of their binary encodings.

Computational problems are one of the main objects of study in theoretical computer science and include **decision problems**, **search problems**, **counting problems** and **optimization problems**.

4.2.1 Decision problems

A problem is said to be a decision problem if its output should be a simple "YES" or "NO" (or derivatives like "TRUE/FALSE", "0/1", "Accept/Reject", etc.). An example of a decision is the primality testing problem: *"Given a positive integer n, determine if n is prime."*.

A decision problem is typically represented as the set of all instances for which the answer is "YES". For the primality testing example, it is represented as the infinite set $L = \{2, 3, 5, 7, 11, \ldots\}$.

4.2.2 Search problems

A search problem is represented as a relation encompassing all the instance-solution pairs, called a search relation. For example, factoring can be represented as the relation $R = \{(4,2),(6,2),(6,3),(8,2),(9,3),(10,2),(10,5),\ldots\}$ which consist of all pairs of numbers (n,p), where p is a non-trivial prime factor of n.

In a search problem, the answers can be arbitrary strings. For example, factoring is a search problem where instances are string representations of positive integers and solutions are string representations of collections of prime numbers.

Search problems and decision problems

A relation R can be viewed as a **search problem**, and an algorithm which calculates R is also said to solve it. Every search problem has a corresponding decision problem, namely $L(R) = \{x \mid \exists y R(x,y)\}$.

This definition may be generalized to n-ary relations using any suitable encoding which allows multiple strings to be compressed into one string (for instance by listing them consecutively with a delimiter).

[1] Traditional Genetic Algorithms also use this encoding to represent problem instances and solutions.

4.2.3 Counting problems

A counting problem asks for the number of solutions to a given search problem. For example, the counting problem associated with factoring is "Given a positive integer n, count the number of non-trivial prime factors of n."

A counting problem can be represented by a function f from $\{0,1\}^*$ to the non-negative integers. For a search relation R, the counting problem associated to R is the function $f_R(x) = |\{y : (x,y)R\}|$.

4.2.4 Optimization problem

An optimization problem asks for finding the "best possible" solution among the set of all possible solutions to a search problem. One example is the maximum independent set problem: "Given a graph G, find an independent set of G of maximum size." Optimization problems can be represented by their search relations.

An optimization problem is the problem of finding the best solution from all feasible solutions and can be divided into two categories depending on whether the variables are continuous or discrete. An optimization problem with discrete variables is known as a **combinatorial optimization problem**. In a combinatorial optimization problem, we are looking for an object such as an integer, permutation or graph from a finite (or possibly countable infinite) set.

Decision Problems and optimization problems

Many of the practical problems are optimization problems, where the goal is to minimize or maximize a given metric or cost function. Each combinatorial optimization problem can be expressed as a decision problem that asks whether there is a feasible solution for some particular measure m_0. For instance: *"What is the lowest number of colors that can be used to paint a graph?"* can be expressed as: *"Given a graph G and an integer k, is it possible to paint G with k colors?"*. Moreover, the Music Transcription problem *"Which are the musical notes that are present in an acoustic signal?"* can also be expressed as a decision problem: *"Given an acoustic signal s and a set of musical notes M, are those musical notes the same set of musical notes present in s?"*. This problem can be answered with a simple "YES" or "No".

In the real world or industry, algorithms are designed to find near-optimal solutions to hard problems. The usual decision version is then an inadequate definition of the problem since it only specifies acceptable solutions. Even though we could introduce suitable decision problems, the problem is more naturally characterized as an optimization problem (Ausiello et al., 1999).

4.3 Polynomial Time as a Reference

The field of Computational Complexity (Hartmanis and Stearns, 1965) attempts to explain why certain computational problems are intractable for computers, i.e. why they cannot be

solved by an algorithm in polynomial time. Essentially, there are several types and classes of problems. However, most problems are divided into two main categories: *easy* problems and *hard* problems. Here, easy stands for *problems that can be solved in polynomial time by an algorithm.*

The polynomial time is a reference that both defines and separates the class of problems that can be solved efficiently. Therefore, if a problem can be solved efficiently, its execution time is polynomial. This evaluation is generally measured in terms of algorithm execution time, using the complexity of the worst possible case, as a function of n, which is the input size of the problem.

In some practical cases, some algorithms can solve simple problems in reasonable time (e.g. $n \leq 20$). But when the number of inputs is larger (e.g. $n \geq 100$) the algorithms performance considerably decreases. These kind of algorithms could be executing in exponential time, in the order: $n^{\sqrt{n}}, 2^n, 2^{2^n}, n!$, or even worse.

For some classes of problems, it is hard to see if there is any paradigm or algorithm which can lead to their solution, or if there is some way to prove that the problem is intrinsically hard, not being possible to find an algorithmic solution whose performance is sub-exponential. For some classes of hard problems it is possible to show that if one of these problems can be solved in polynomial time, then all these problems can also be solved in polynomial time.

A polynomial time algorithm has an execution time in the order of $O(n^k)$, where k is a constant independent of n. A problem is said to be *solved in polynomial time* if there is an algorithm that can solve it in polynomial time.

Some functions might not appear polynomial but can be treated like that, for instance: $O(n \log(n))$ has a superior limitation by order $O(n^2)$. Some other functions might appear to be polynomial but, if we look closely, they are not. For instance: $O(n^k)$, is not polynomial if k is a function of n, the input size. Figure 4.1 represents the most common execution times.

4.3.1 P and NP classes

The **P** class of problems is defined by all decision problems that can be solved in polynomial time. There are also other kind of problems, such as the **NP** class, that cannot be solved in polynomial time but when an answer is provided it is possible to verify that answer in polynomial time.

Consider the subset problem, an example of a problem that is easy to verify, but whose solution cannot be computed in polynomial time: *Given a set of integers, does some empty subset of them sum to 0?*. For instance: does a subset of the set $\{-2, -3, 15, 14, 7, -10\}$ add up to 0? The answer is *"YES, because* $\{-2, -3, -10, 15\}$ *add up to 0"* can be verified with four additions. However, there is no known algorithm to find such a subset in polynomial time: there is one in exponential time, which consists of $2^n - 1$ tries, i.e. complexity $O(2^n)$. In this case, the subset $\{-2, -3, -10, 15\}$ is said to be a solution. If it is possible to evaluate a solution for a problem in polynomial time, it is said that the problem is *verifiable in polynomial time.* Not all the problems have the property of being easily verified. For instance, if the problem is to identify if the set has only one subset

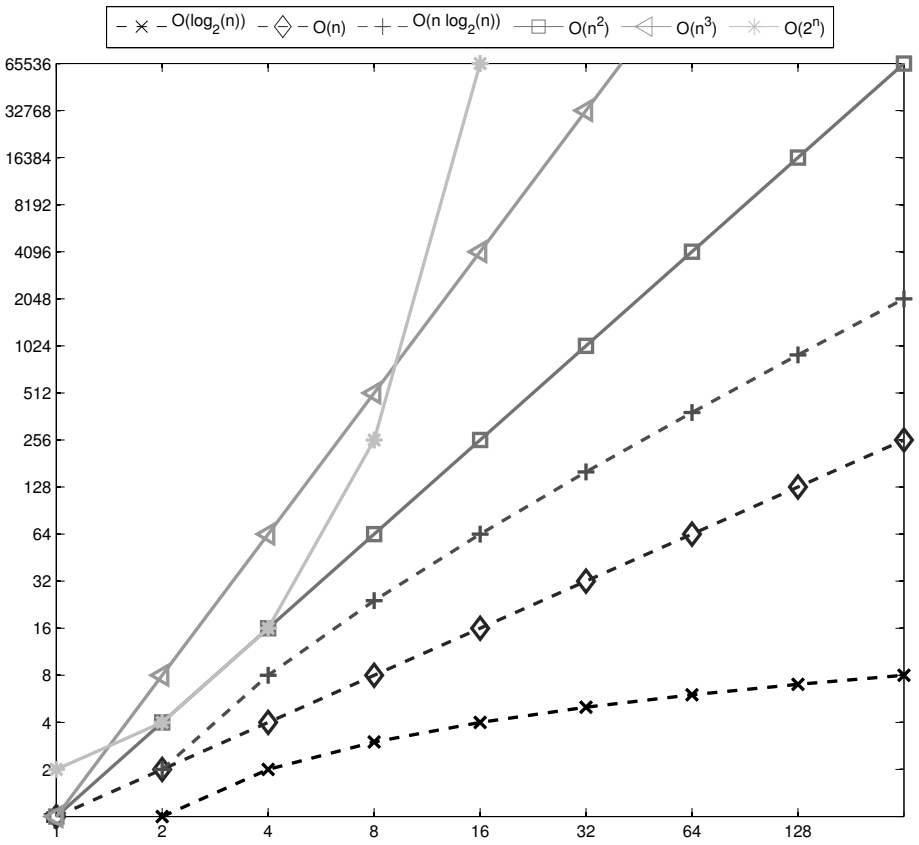

Figure 4.1: Most common execution times.

with zero sum. It is easy to verify is there is at least one subset, but it is not easy to demonstrate that there are no other subsets with zero sum.

The class of NP problems is defined by all problems that can be verified by an algorithm in polynomial time. NP stands for *Nondeterministic Polynomial Time*, which is related to a program being executed on a non-deterministic computer, with guessing capabilities. Basically this architecture could be able to non-deterministically find the value of the solution and verify, in polynomial time, if the solution really solves the problem.

Recall that the P class is defined by all problems that are solvable in polynomial time. This way, if a problem is solvable in polynomial time then it can certainly be verified in polynomial time if the solution is correct: $P \subseteq NP$.

No one knows if $P = NP$ (Baker et al., 1975). Hence, being able to verify if a solution is correct in polynomial time does not help to find an algorithm to solve the problem in polynomial time. Many authors believe that $P \neq NP$ but there is still no proof (Cook, 1971; Baker et al., 1975; Ladner, 1975; Fortnow, 2009).

If any NP problem can be transformed in polynomial time into a certain NP problem, this means that this certain problem is **NP-Complete**, i.e. it is at least as hard as any NP problem (Cook, 1971). This way, if it could be possible to solve an NP-Complete problem in polynomial time, it would be possible to solve all NP problems in polynomial time (Michael and Johnson, 1979). Also, if an NP-Complete problem can be reduced in polynomial time into other problem, this problem is proven to also be NP-Complete. The

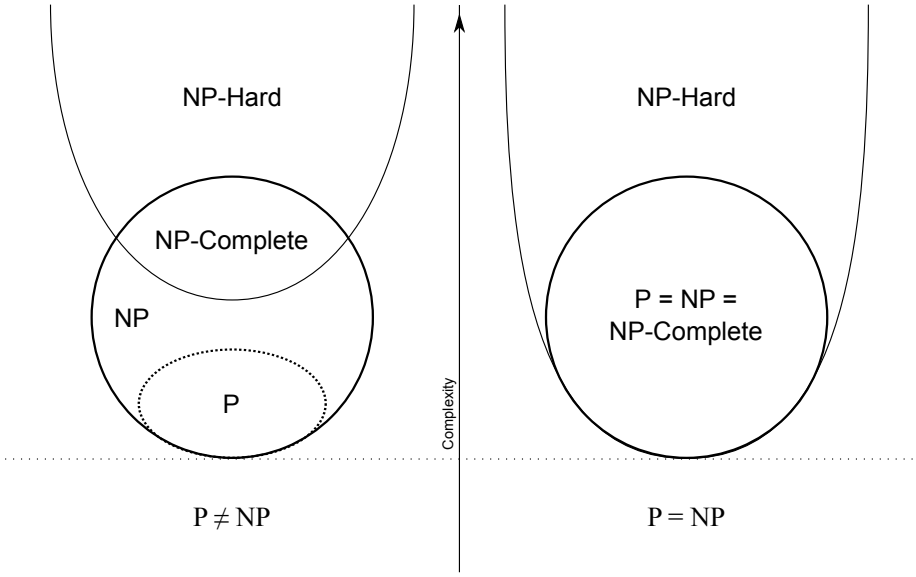

Figure 4.2: Classes of problems according to their computational complexity.

first problem known as NP-Complete was the Boolean satisfiability (SAT) problem (Cook, 1971).

According to Cook (1971), NP-Complete problems are those which are at least as hard as the hardest problems in NP. That is: any NP-Complete problem is both NP and **NP-Hard**. However, an NP-Hard problem might not belong to the NP-Complete class: there are NP-Hard problems that are not NP-Complete, indeed, they may not even be decision problems.

4.4 Bio-inspired Algorithms

Bio-inspired Algorithms (BAs) are based in analogies with natural or social systems and are used for the design of heuristic and non-deterministic methods for the resolution of complex problems, like the NP-Complete and NP-hard problems. Usually, BAs have a parallel structure (multiple agents) and are adaptive.

The brain, the immune system of living beings, even simple living beings as a whole, and the process of organic evolution, led to models or systems worthy of imitation. These models are of broad interest for creating such intelligent machines, like neural networks, cellular automata, animats, etc Schwefel (1994). The most well known bio-inspired computational models are: Evolutionary Algorithms, Neural Networks, Immune Systems, Cellular Automata and Swarm Intelligence (e.g.: Ant Colony Optimization, Particle Swarm Optimization, etc.).

4.4.1 Evolutionary Computation

Evolutionary Computation (EC) is a field based in Darwin's theory of Evolution Darwin (1859) and in Mendel's laws of inheritance (Mendel and Bateson, 1925): evolution of the species is made by natural selection where the fittest will breed - survival of the

fittest -, thus contribute to evolution, and by elimination of the less fit individuals. This is based on three assumptions:

Outbreed There are more descendants ($\lambda + \mu$) than the ones who can survive (μ).

Variability There is variability regarding structure and corporal function in the whole species.

Inheritance Many characteristics of the individual will pass to their descendants by means of genetic transmission.

Evolutionary Process

Natural evolution is an evolving process over a population of reproductive individuals who have inheritable characteristics, which may change the individual's fitness value (chance to breed). Therefore, to have evolution, the individuals should be able to reproduce, their survival depends from their characteristics (which may be affected by variations - mutation), those characteristics pass from fathers to sons by inheritance and all individuals are competing for the same resources.

The evolutionary process happens due to some mechanisms, which are not totally known, but only a few of their characteristics:

- Evolution is a process which operates more over the chromosomes than over the structures of life encoded on them.
- Natural selection is the linkage between the chromosomes and behaviors of their decoded structures.
- Reproduction is the process where evolution appears.

Biological evolution does not have memory. Species are created, evolve and then disappear if they don't adapt. Only the best, the fittest, those who best adapt to their environment will survive to perpetuate their skills.

Computation sees in this a clear process of optimization: we have the fittest individuals - the best temporal solutions -, they reproduce, breed new born individuals - new solutions -, which contain part of the genetic code - information - of their parents and the average fitness value turns into a better value.

Hence, Artificial Evolution consists on modeling an evolution process based on populations whose elements are candidate solutions to the problems. The simulation of this process in a computer turns to be a stochastic optimization technique, which turns to be a better option than other classic methods for hard problems, like the NP-hard problems. It is an alternative approach for complex search and learn problems using computational models of evolutionary processes (De Jong and Spears, 1989).

Evolutionary Computation arises from a fact observed in nature: the living organisms possess dexterity in the resolution of the faced problems, acquiring their abilities by natural evolution. Evolution processes in all living organisms as a consequence of two primary

processes: natural selection and reproduction. Therefore, any Evolutionary Program must have the following attributes Michalewicz (1994):

- A genetic representation of possible solutions for the problem.
- Some kind of mechanism to crate the initial population to contain potential solutions.
- A Fitness Function to simulate the role of the environment.
- Genetic Operators to change the composition of new born solutions.
- Values for the different parameters used by the algorithm.

Evolutionary Computation as a mechanism for solving optimization and search problems

Evolutionary Computation started in the 50's with works of Anderson (1953); Fraser (1957); Bremermann (1962); Friedberg (1958), among others (Schwefel, 1981). The field remained in the unknown for almost three decades due to the absence of a robust and computational platform and also methodological defects in the first methods (Fogel). The works of Holland (1975); Rechenberg (1973); Schwefel (1975); Fogel et al. (1966) changed slowly this scenario. Nowadays, the growth of scientific production in this field is exponential.

The benefits on using Evolutionary Computation are gains in flexibility and adaptability, with robust performance and general characteristics. Its main concept can be applied in problem resolution, specially hard optimization problems. An optimization problem requires the fulfillment of a set of parameters in order to achieve a certain quality criteria: $\max(f(x))$ or $\min(f(x))$. All kinds of hard or unsolvable problems like: high number of dimensions, multi modalities, strong non-linearity, non-differentiable, noisy and time dependent functions are perfect candidates of Evolutionary Algorithms.

In a few words, Evolutionary Computation is a term for referring systems for solving optimization or search based problems on a computer. These systems apply computational models of a known evolution mechanism as the key in its design and implementation.

Why do they work? Evolutionary Algorithms are solution generation methods which start from an initial set of candidate solutions and then employ several search operators that refine that set. This refinement is made by several gradient follow techniques with biological-based mechanisms of exploration: population of solutions and genetic operators (Rudolph, 1998; Oliveto and Witt, 2012). The basic operations are: evaluation, selection, reproduction and mutation. There are two tasks implicitly made: exploration of the search space and exploitation of "good" zones. The basic paradigms of Evolutionary Algorithms are:

Evolutionary Programming (EP) which emphasizes the behavior changes among species. Its first roots were laid by Fogel et al. (1966) and designed by Fogel (1992) in its currently practiced form;

Evolutionary Strategies (ES) which emphasize the behavior changes among individuals. ES were first introduced by Rechenberg (1973) and further developed by Schwefel (1975);

Genetic Algorithms (GAs) which use genetic operators over chromosomes and the concept of sexual reproduction (recombination). GAs were first introduced by Holland (1975) and were first used for optimization tasks by De Jong (1975);

Genetic Programming (GP) which is one of the subbranches of the GAs and applies the genetic operators to programs or mathematical expressions represented as trees of operands and operators (Koza, 1992).

Evolutionary Algorithms

Evolutionary Algorithms start with a population of μ individuals and then generates λ offspring individuals. The next generation is composed by the best μ individuals from the previous generation $\lambda + \mu$ individuals. Algorithm 4.1 shows the pseudo code of a classic Evolutionary Algorithm:

Algorithm 4.1: Algorithm $((\mu + \lambda)\, EA)$

1: $t \leftarrow 0$;
2: Initialize P_0 with μ individuals chosen uniformly at random;
3: **repeat**
4: **for** i = 1 to λ **do**
5: choose $x_i \in P_t$ uniformly at random;
6: flip each bit in x_i with probability p;
7: **end for**
8: Create the new population P_{t+1} by choosing the best μ individuals out of $P_t \cup \{x_1, \ldots, x_\lambda\}$;
9: $t \leftarrow t + 1$;
10: **until** a stop condition is fulfilled.

$p = \frac{1}{n}$ is generally considered as best choice (Bäck, 1993; Droste et al., 1998). If $\mu = \lambda = 1$, the resulting Evolutionary Algorithm is a (1+1)-EA (see Algorithm 4.2). In this case, if only one bit is flipped per iteration, the resulting algorithm is Random Local Search (RLS).

When not to use Evolutionary Algorithms

As pointed by Schwefel (1994), *"Nobody should make use of Evolutionary Computation in cases where other methods like linear and dynamic programming, quasi-Newton, or other well known approaches work. None of the Evolutionary Algorithms (EAs) would perform the job better not even as good as those. Evolutionary Computation should only be taken in consideration if and only if classical methods for the problem at hand do not exist, are not applicable or, obviously, fail. Even at this stage, at least two alternatives should be discussed: either total enumeration or other brute force methods, when the*

Algorithm 4.2: Algorithm $((1+1)\,EA)$

1: $t \leftarrow 0$;
2: Initialize P_0 with $x \in \{0,1\}^n$ by flipping each bit with $p = \frac{1}{2}$;
3: **repeat**
4: create x' by flipping each bit with $p = \frac{1}{n}$;
5: **if** $f(x') \geq f(x)$ **then**
6: $x' \in P_{t+1}$
7: **else**
8: $x \in P_{t+1}$
9: **end if**
10: $t \leftarrow t + 1$;
11: **until** a stop condition is fulfilled.

necessary computation power is at hand, and, last but not least, the development of a specific method, which makes full use of the knowledge of the problem's structure, like expert systems. Evolutionary Algorithms are weak methods, which should be handled as last resort.".

4.5 Genetic Algorithms

Genetic Algorithms (GAs) are a subclass of the Evolutionary Algorithms. Genetic Algorithms are optimization, search and learning algorithms inspired in the Natural Evolution and in Mendel theory of inheritance (Mendel and Bateson, 1925). GAs model the evolutionary process as a succession of gene changes, with candidate solutions analogous to chromosomes. The entire search space is explored by applying transformations to these candidate solutions, just as it happens in the living beings: recombination, selection and mutation.

Genetic Algorithms constitute the most complete paradigm on the Evolutionary Computation, since they resume in a natural way all the fundamental ideas of natural evolution. GAs are also very flexible, thus it is easy to adopt new ideas that may occur from the evolutionary computation field. They are also easy to hybridize with other paradigms which are not related to Evolutionary Computation, like local search. For instance: Memetic Evolutionary Algorithms Moscato (1989) are hybrid Genetic Algorithms with local search operators.

4.5.1 Biological Background

The Deoxyribonucleic acid - DNA - is the fundamental genetic material of all living organisms. It is a double-stranded macro-molecule shaped like a double helix. The two strands are linear molecules of nucleic acid without ramifications, formed by alternate molecules of deoxyribose (sugar) and phosphate. The four bases of nucleotide: Adenine (A), Thymine (T), Cytosine (C) and Guanine (G) are the alphabet of the genetical information. The

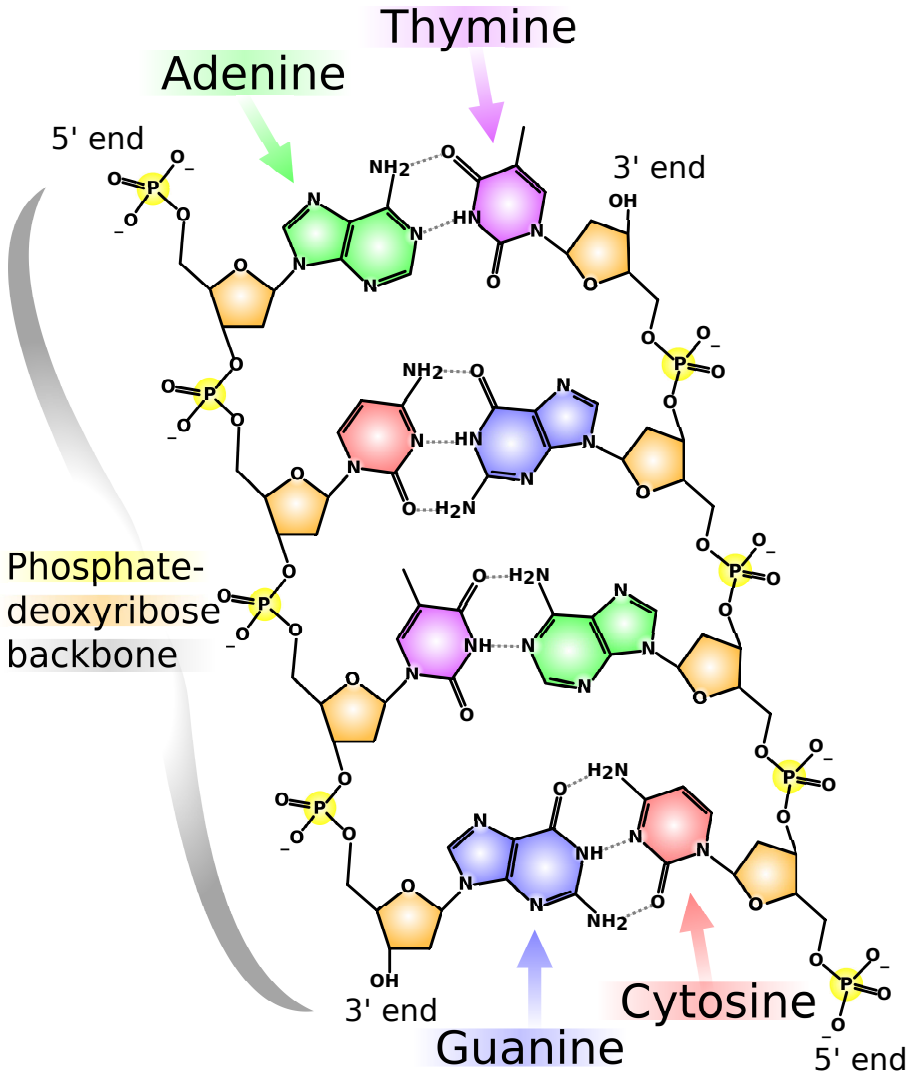

Figure 4.3: DNA chemical structure (from: `http://en.wikipedia.org/wiki/DNA`).

sequences formed in the molecule by those four bases will specify the constructive plan of any organism (see Figure 4.3).

A gene is one DNA section, which encodes a certain biochemical function, usually defined in the production of a protein. It is mainly a heritage unit. The DNA of an organism may contain from one dozen of genes (like the viruses) to thousands of genes (like the humans).

Chromosomes are the structure of the DNA and protein found in the cell nucleus. Each chromosome contains hundreds or thousands of genes that form our hereditary blueprint. Humans have 23 pairs of chromosomes (each parent contributes with one chromosome in each pair), containing a total of 50,000 to 100,000 genes. The chromosomes are responsible for the genetic information transmission. Each gene only occupies one particular region of a chromosome - locus. For each specific place, there can be several alternative forms of the gene, among the population. These alternatives are called alleles.

The Genome is the entire collection of genes (therefore, chromosomes) of an organism. A particular set of genes of the genome is the genotype. The genotype is the basis of the

Genetic Algorithm Terminology	Meaning
Chromosome (Individual)	Solution (code)
Gene (bit)	Part of the solution
Locus	Gene position
Allele	Gene value
Genotype	Encoded solution (internal appearance)
Phenotype	Decoded solution (external appearance)

Table 4.1: Genetic Algorithm terminology

phenotype of an organism: its physical and mental characteristics (for instance: the eye color, intelligence, etc.).

During the reproduction or recombination the genes of the parents are combined to form a new chromosome. The new descendant (offspring) may suffer mutations: the DNA changes slightly due to errors when copying the genes of the parents. The aptitude (fitness) of an organism is measured by its success in surviving (selection).

4.5.2 Terminology

In biology, a cell contains chromosomes, which contain genes. Each gene is in a locus (its place on the chromosome) and the alleles are the values that a gene can have.

In Genetic Algorithms terminology, there is the chromosome, which is the individual or the genotype. The individuals are candidate solutions to the problems, which are usually encoded as a binary string. In this case the alleles are 0 and 1. Table 4.1 shows the main terms in the Genetic Algorithm terminology and their meaning.

4.5.3 Definition

A Genetic Algorithm is a highly parallel algorithm which transforms a set of individual objects, using modeled operations according to the Darwin's theories of reproduction and survival of the fittest (Darwin, 1859). The transformation occurs after several genetic operations, from which we emphasize the recombination. Each individual may be a set of characters (letters or numbers) with fixed length, which fits to the model of the set of chromosomes, and is associated with a certain mathematic function to reflect its aptitude (fitness).

The algorithm starts with an initial population of individuals (chromosomes), which are candidate solutions to the problem. Some solutions are used to form a new population hoping that its individuals are better than the individuals from the old population (previous generation). The solutions are selected by their aptitude (fitness) to form the next population (generation). This process is repeated until a satisfactory condition is achieved (number of generations, improvements of the best solution, etc.). Algorithm 4.3 presents the Standard Genetic Algorithm (SGA).

The main idea behind Genetic Algorithms is to have a set of candidate solutions (individuals) which evolve towards the desired solution. In each iteration (generation) those candidate solutions are evaluated according to their quality (fitness). The worst ones are

Algorithm 4.3: Genetic Algorithm

1: $t \leftarrow 0$
2: generate initial population(P_t)
3: evaluate(P_t)
4: **while** stopping criteria does not meet **do**
5: $P'_t \leftarrow$ select(P_t)
6: recombine(P'_t)
7: mutate(P'_t)
8: $P_{t+1} \leftarrow$ create next population (P_t, P'_t)
9: $t \leftarrow t + 1$
10: **end while**
11: **return** best individual found

discarded and the best will generate new candidate solutions, which result by merging (recombination) their parents characteristics (genes) and applying minor variations (mutation). Candidate solutions with best quality will tend to live longer and to generate better solutions, improving the robustness of the algorithm.

Genetic Algorithms model the evolution process as a succession of gene changes, with solutions analogous to chromosomes. The entire search space is explored by applying transformations to these candidate solutions, just as it happens in the living beings: recombination, selection and mutation (Goldberg, 1989). The evolution usually starts from an initial population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on their fitness), and modified (recombined and possibly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

Representation of Individuals

Genetic Algorithms started and still mainly operate with binary strings for representing candidate solutions for the given problem (their genotype). This encoding follows the computation problem description in Section 4.2: both problem instances and solutions are conventionally represented as binary strings, namely elements of $\{0, 1\}^*$.

Generation of Initial Population

The initial population is usually generated with individuals chosen uniformly at random.

Sexual Reproduction - Recombination

Genetic Algorithms emphasize the role of recombination. To generate the offspring (next generation), individuals are selected for breeding and are then recombined. The selection

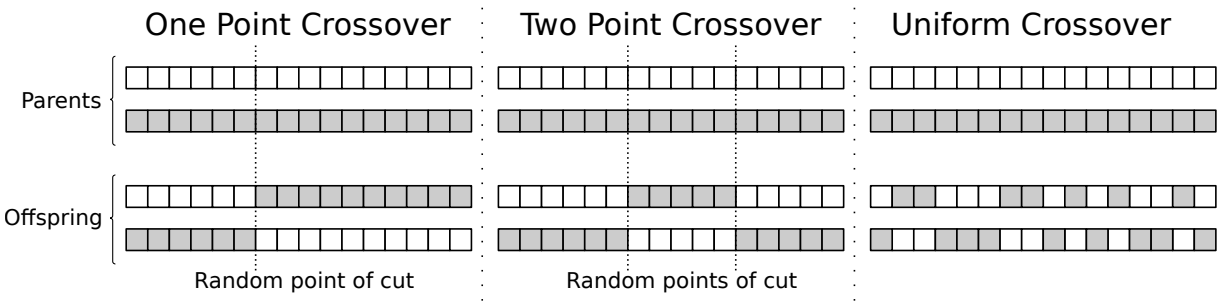

Figure 4.4: One point crossover, two point crossover and uniform crossover.

for breeding is proportional to the fitness of each individual: the fittest have higher chance of being selected to breed.

There are two main proportional selection mechanisms: roulette wheel and tournament Goldberg (1989). In the roulette wheel, a roulette wheel slot is assigned to each individual in the population, whose size is proportional to their fitness. The roulette wheel consists in an array of partial cumulative fitnesses:

$$f_c = \sum_{j=1}^{s} f(j) \tag{4.1}$$

where an uniformly distributed random number r between zero and the total cumulative fitness $f_c(M)$ is drawn and the minimum string index s that satisfies the $f_c(s) > r$ condition is chosen.

The tournament selection operator consists in randomly selecting n (where n is the tournament size) individuals and then returning the one who has the highest fitness value.

After the parent selection, both parents are recombined to form two new individuals (offspring). There are three main crossover operators: one-point crossover, two-point crossover and uniform crossover Goldberg (1989). Both one-point and two-point crossover exchange genetic material from both parents according to the random generated point of cuts, while the uniform crossover swaps random selected genes. These crossover operators are illustrated in Figure 4.4.

Mutation

Mutation consists of random bit inversions in the binary encoding of each individual, generally occurring with low frequencies.

Selection

The selection procedure decides which individuals will survive (pass to the next generations). There are elitist variants, where good parental individuals cannot get lost (this is good for proving global convergence Rudolph (1998)) and the classical one, where the offspring generation simply replaces the parent generation.

4.6 Summary

During this chapter, we presented the concept of Algorithm and its main purpose: problem solving. Different types of problems have been introduced (decision, search, counting and optimization) and how they can be viewed as decision problems. We have introduced different classes of decision problems, according to their computational complexity, namely: P, NP, NP-Complete and NP-Hard problems.

Bio-inspired algorithms were introduced as mean of addressing NP-Complete and NP-Hard problems. Genetic Algorithms were finally presented as a kind of Evolutionary Algorithms. Both biological background and Genetic Algorithm terminology were explained.

Chapter 5

Automatic Transcription of Music and Multi-Pitch Estimation - A Deeper Analysis

This chapter presents the problem Automatic Transcription of Music from the search space or optimization point of view. Automatic Transcription of Music is presented as an optimization problem, and an idea about the size of the search space is given. The computational complexity of the problem is studied and the NP-Completeness of the problem is discussed. Then, a discussion is presented on how to address combinatorial optimization problems with Genetic Algorithms and, finally, the problem of achieving Automatic Transcription of Music with Genetic Algorithms is addressed.

5.1 Automatic Transcription of Music as Optimization or Search Space Problem

Formally, a Combinatorial Optimization optimization problem A is a quadruple (I, f, m, g), where:

- I is a set of instances;
- given an instance $x \in I$, $f(x)$ is the set of feasible solutions;
- given an instance x and a feasible solution y of x, $m(x, y)$ denotes the measure of y, which is usually a positive real.
- g is the goal function, and is either **min** or **max**.

The goal is to find for some instance x an *optimal solution*, that is, a feasible solution y with $m(x, y) = g\{m(x, y')|y' \in f(x)\}$.

5.1.1 Search Space Size

If we consider an audio signal with a single audio frame, and polyphony-order k (k notes played at the same time), the size of the set $f(x)$ (feasible solutions) is

$$\binom{127}{k} = \frac{127!}{k!\,(127-k)!} \tag{5.1}$$

if we consider the 127 musical notes specified in the MIDI specification. On the other, hand if we consider only a single audio frame, with l musical instruments, playing k musical notes, the set of feasible solutions will contain

$$\binom{127}{k}^{l} = \left[\frac{127!}{k!\,(127-k)!}\right]^{l} \tag{5.2}$$

elements. Finally, if we consider n time frames, the size of the search space will increase to

$$\left[\binom{127}{k}^{l}\right]^{n} = \left(\left[\frac{127!}{k!\,(127-k)!}\right]^{l}\right)^{n}. \tag{5.3}$$

Recall that Multi-Pitch Estimation consists in estimating the musical notes that are present in each time frame. On the other hand, Automatic Transcription of Music consists in discrete note tracking, that is, besides estimating the pitch of each musical note in the acoustic signal, the respective start time, end time and timbre, among many others features, should also be inferred. This way, when compared to Multi-Pitch Estimation, the problem complexity of Automatic Transcription of Music is increased exponentially almost until infinity, since much more dimensions are taken into account and it should also be considered that a musical note can occur at any time and have any duration.

5.1.2 Computational Complexity: NP-Complete or NP-Hard?

The question may arise whether Multi-Pitch estimation is NP-Complete or NP-Hard, given the almost infinite size of the search space. Until now, no one can state that Multi-Pitch Estimation is NP-Complete, since there is no agreed-upon algorithm (at any computational cost) for determining whether the problem can been solved. In fact, Multi-pitch Estimation is a problem that remains unsolved. We also believe that it cannot be shown that it is NP-Hard, since no one has proved that a polynomial-time algorithm to solve Multi-Pitch Estimation does not exist.

As a matter of fact, NP-Completeness or NP-Hardness of Automatic Transcription of Music is indeed an interesting question, but Automatic Transcription of Music is not a well posed problem such as factoring, sorting, network flow or subset problems. While it sounds easy to state *"In a given polyphonic music recording, find the musical notes that best describe the input signal."*, some research has focused in formulating the problem as a proper computational problem (Smaragdis and Brown, 2003; Cemgil et al., 2006a; Févotte et al., 2009).

Current approaches typically have an underlying model like, for instance, some kind of factorial Hidden Markov Models or Non-negative Matrix Factorization (see Chapter 3), and the Multi-Pitch Estimation is cast as an inference/optimization problem in this context. For such formulations, one could speak about the computational complexity of exact optimization but most of the time there are NP-Hardness results (specially for NMF or HMM). In practice, authors work with approximations, i.e. approximate algorithms are used for approximate models, so computational complexity results are not much of focus in Multi-Pitch Estimation or Automatic Transcription of Music.

In this context, the problem with underlying models like NMF is that there is typically a large computational step excluded from the analysis. Unless there is a way to figuring out appropriate initialization matrices, there is absolutely no guarantee that the decomposition of the spectrogram will lead to a decomposition that will be found useful (e.g. into notes, into single sources, or into spectral elements that come exclusively from single sources). Therefore, there is a lot of "back end" work to group the resulting decomposed elements into something useful, as well as a lot of "front end" work to seed the matrices just right. In our opinion, any reasonable analysis of the time/space effort required to make NMF practically useful should include the cost of this aspect.

5.2 Addressing Combinatorial Optimization Problems with Genetic Algorithms

The main idea behind a Genetic Algorithm (Holland, 1992b) is to have a set of candidate solutions (individuals) to a problem evolving towards the desired solution. In each generation those individuals are evaluated according to their quality (fitness). The worst individuals are then discarded and the best will generate new individuals resulting from the combination of their parent's characteristics (genes) and minor variations (mutation). Therefore, individuals with better quality tend to live longer and to generate better and fitter offspring, thus improving the robustness of the algorithm. This way, there is a natural relation between these concepts and the formal definition of Combinatorial Optimization problems (defined in Section 5.1):

- I is a set of instances;
- given an instance $x \in I$, $f(x)$ is the set of feasible solutions: $f(x)$ is the search space of the Genetic Algorithm for x (input problem);
- given an instance x and a feasible solution y of $x, m(x, y)$ denotes the measure of y, which is usually a positive real: $m(x, y)$ is the fitness value (quality) of the y individual for the input problem x;
- g is the goal function, and is either **min** or **max**: the goal of the genetic algorithm is to minimize or maximize the fitness value.

The goal to find for some instance x an *optimal solution*, that is, a feasible solution y with $m(x, y) = g\{m(x, y')|y' \in f(x)\}$: the objective of the Genetic Algorithm is to find

best possible solution - the one who has the biggest or the lowest fitness from all the feasible solutions.

5.3 Addressing Automatic Transcription of Music and Multi-Pitch Estimation with Genetic Algorithms

When addressing Genetic Algorithms to the problem of Automatic Transcription of Music or Multi-Pitch estimation, there are several aspects that must be taken into account, mainly: genotype, fitness function, selection, recombination, mutation, creation of the initial population, and the survivor selection method.

5.3.1 Genotype

As explained in the previous chapter, the Genotype specifies how each individual or feasible solution is encoded. Given that each individual represents a candidate solution to the problem, for the Automatic Transcription of Music, the problem should be encoded as a sequence of discrete note events (transcription). One must be aware that the number of musical notes that are present in the input signal is not known beforehand, furthermore, different input signals have, most likely, different number of musical notes. This, way, the number of genes (note events) varies from individual to individual and will probably change during the evolutionary process. For discrete note tracking, each note encoded in the individual's genome should have, at least, the following information: start time (onset), duration, note pitch and other information like the dynamics of the musical note (loudness or MIDI velocity). It might also be useful to encode the harmonic model of each musical note, specially in case of instrument classification or recognition.

Multi-Pitch Estimation, takes only into account the frequencies or musical notes which are present in an audio signal. This way, each individual should encode at least the set of frequencies or pitches. Here, the temporal information (onset time and offset time) are not taken into account. To best cope with the harmonic collision, it might also be useful to include the harmonic model of each musical note into the individual.

When transcribing audio signals, the residual noise should also be taken into account, so that spurious frequency components are not considered as musical notes. This way, one can also take into account noise estimation and encode this information within the individual's genome.

5.3.2 Fitness Evaluation

Fitness evaluation allows us to measure the quality of each candidate solution. That is: how close each candidate transcription is to the desired solution. Fitness evaluation plays the major role in evolutionary algorithms, since it is what tells the algorithm what fitter solutions are.

In Automatic Transcription of Music, fitness evaluation will measure the similarity between the generated transcriptions and the original audio. This way, the algorithm

must be able to look at each note sequence and measure how close it is, from the musical point of view, to the original audio. Since the algorithm does not know what is the solution (the correct transcription), one question arises: *how can a feasible transcription be assessed in terms of how much close it is to the desired solution?*. To answer this question, there are two possible choices, one of them being to apply some sort of correlation coefficient between the input signal spectrum and a comb spectrum generated by the Fundamental Frequencies encoded in each individual. The other possible answer to this question relies on synthesis. It is possible to synthesize each candidate solution into an acoustic signal, and then compare the generated signal with the input signal: the closer the two signals are, the better the transcription is. On the other hand, this raises new questions:

- What kind of synthesis (additive synthesis, subtractive synthesis, FM synthesis, physical modeling synthesis, sample based techniques, etc.) should we use?
- In which domain should we perform the comparison: temporal domain or frequency domain?
- What happens if the synthesizers used for rendering each individual into an audio signal have spectral envelopes (timbres) very different from those played in the input signal? This way, transcriptions assessed as good will be measured as "not so good" transcriptions.
- How do we deal with noise?

A good function should address all these questions.

5.3.3 Selection

Genetic Algorithms use three main operations: selection, recombination and mutation. The purpose of the selection operators is to select individuals for breeding (generate new individuals). In order for the algorithm to converge, the selection procedure must not be random. Instead, the selection procedure should be proportional to the rank (fitness value) of each individual. That is: fitter individuals should have higher probability of being selected. On the other hand, and since the algorithm should be able to overcome the local maxima and explore different areas of the search space, one cannot use only the best two individuals to generate the entire offspring. This way, the selection operator should also be stochastic but ensure that the fittest individuals have higher probabilities of being selected.

5.3.4 Recombination

Recombination is the main pillar of Genetic Algorithms and it is the feature that distinguishes GAs from other Evolutionary Algorithms. The main purpose of recombination is to exchange genetic material from two individuals (parents) and, by doing this, generate two new individuals (offspring). By doing so, the two new born individuals inherit the genes of both parents, that is, they inherit both parents characteristics.

Recombination has the role of exploring new areas of the search space, by combining the already achieved feasible solutions. In the case of Multi-Pitch Estimation or Automatic Transcription of Music problems, one must have into account that the number of musical notes or fundamental frequencies might differ from individual to individual. This way, the classic one-point, two-point and uniform crossovers Goldberg (1989) cannot be applied. Instead, they must be adapted in order to cope with individuals with different number of genes.

5.3.5 Mutation

Mutation also has an important role on the robustness of Genetic Algorithms. The role of the mutation operator is to perform minor changes inside the genetic code of each individual. This way, the Genetic Algorithm can explore new areas of the search space that could not be searched by using only recombination. Typically, mutation consists of random bit changes, with very low probability Goldberg (1989).

Regarding the problem of Multi-Pitch Estimation, the GA will need to be able to delete musical notes, create new notes (totally random or based on existing musical notes) or changing the existing musical notes, by modifying its pitch, dynamics or source, if present, and also, for Automatic Transcription of Music, its start time and duration.

5.3.6 Creation of the Initial Population

A Genetic Algorithm starts by creating an initial population, and then applies the selection, recombination and mutation operators, resulting on the next generation. The same process happens over and over again until a stopping criterion is met. This way, the initial population happens to be the first step/stage of the evolutionary process. In most Genetic Algorithm applications, the initial population is randomly generated. This has the advantage of exploring completely different areas of the search space. On the other hand, if one manages to create an initial population that is close to the desired solution, the algorithm will need less generations to find the global optimum.

By taking into account the scope of Multi-Pitch Estimation and Automatic Transcription of Music, it makes sense to perform some kind of spectral analysis on the input signal. This way, by analyzing the spectral peaks, one can estimate which is the set of possible notes inside the signal and, in case of Music Transcription, where they are. By doing this kind of analysis, one can reduce significantly the size of the search space and constrain the possible musical notes or fundamental frequencies to a much smaller set to be used for the random generation of the initial population.

5.3.7 Survivor Selection

The survivor selection consists in choosing which individuals from both current population and offspring will survive, that is, to create the next generation. Survivor selection plays a major role in the convergence of the Genetic Algorithm. The definition of convergence of an Evolutionary Algorithm implies that

- Ideally the Evolutionary Algorithm should find the solution in finite steps with probability 1 (visit the global optimum in finite time);
- If the solution is held forever after, then the algorithm converges to the optimum.

According to Rudolph Rudolph (1998), convergence implies two conditions:

- There is a positive probability to reach a point in the search space from any other point;
- the best found solution is never removed from the population (elitism).

GA theory also says that canonical GAs using mutation, crossover and proportional selection do not converge Oliveto and Witt (2012). On the other hand, elitist variants do converge. This way, and since the goal is to find the global solution (perfect transcription or perfect set of fundamental frequencies) the convergence of the algorithm is mandatory: elitist variants should be taken into account.

5.4 Summary

This chapter presented the problem Automatic Transcription of Music as a combinatorial optimization problem and a perspective about the size of the search space was given. Then, we presented a discussion about the computational complexity of the problem. We have also shown how to address combinatorial optimization problems with Genetic Algorithms and, finally, proposed a way to address the problems of Automatic Transcription of Music and Multi-Pitch Estimation with Genetic Algorithms. Several important issues were identified and discussed.

Chapter 6

Early Genetic Algorithm Approaches to Automatic Transcription of Music: Synthesized Signals and Simple Mathematical Models

During the entire chapter we will describe the first Genetic Algorithm approaches to the problem of Automatic Transcription of Music. We will also discuss how each approach found in the literature addresses each of these topics: genotype, fitness evaluation, selection, recombination, mutation, creation of the initial population and survivor selection.

6.1 First Genetic Algorithm approach to Polyphonic Pitch Detection

The first work in the literature using Genetic Algorithms for polyphonic pitch detection appears in 2001 by Garcia (2001). Garcia claims that polyphonic pitch detection can be considered as a search space problem where the goal is to find the pitches that compose a polyphonic acoustic signal. This way, it makes sense to use genetic algorithms since they perform very well in search space problems (Goldberg, 1989).

According to Garcia, when there is no a-priori knowledge about the number of fundamental frequencies present on a signal, the size of the search space can be extremely large since it consists of all possible combinations of $1, 2, \cdots, \frac{Fs}{2}$ simultaneous frequencies, where $\frac{Fs}{2}$ is half of the sampling rate. For instance, for signal with 44.100Hz sample rate, from 21Hz to 22.050Hz, with an F0 resolution of 10.766 Hz and up to 4 fundamental frequencies, the size of the search space is $1,7549 \times 10^{13}$.

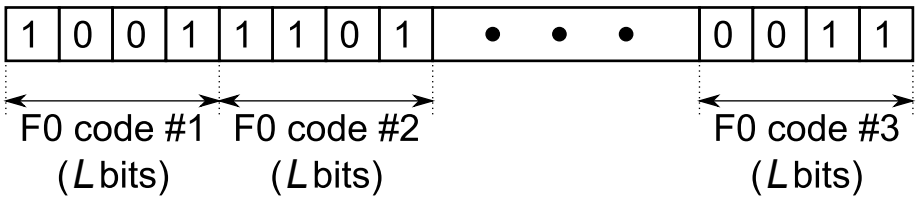

Figure 6.1: Garcia's approach chromosome structure with $L = 4$ bits.

6.1.1 Genotype

Garcia's approach encodes each chromosome as a binary string with variable length (see Figure 6.1). The chromosome's structure is a concatenation of N substrings of L bits each. Each substring encodes one F0 value using binary fixed-point representation. Although the length of the L substrings is fixed since F0 range and resolution is specified as an input parameter, the length of the chromosome is variable because no assumption is made about the number of F0s in the signal. The length of the substrings is defined according to the frequency range, $\Delta F0$, and frequency resolution, $dF0$, as the minimum integer L, where:

$$2^L \geq \frac{\Delta F0}{dF0}. \tag{6.1}$$

6.1.2 Fitness Function

The fitness measure of Garcia's approach (Garcia, 2001), $f(s)$, for a given string or chromosome, s, is based upon a correlation between the input spectrum and a comb spectrum defined in Martin (1981). The partial fitness value $f_p(s, j)$ is computed for each fundamental frequency value, $F0^j$, coded by substring j in string s, as the correlation between the input magnitude spectrum $|X(\omega)|$ and a reference comb spectrum with exponentially decreasing amplitudes $e^{-\alpha h}$, where h is the harmonic index and α a specified input parameter:

$$f_p(s, j) = \sum_h \left|X(2\pi h F0^j)\right| .e^{-\alpha h}. \tag{6.2}$$

After the partial fitness $f_p(s, j)$ is computed for a substring j, the input DFT bins used in the correlation sum are zeroed for the remaining partial fitness evaluations of the string. This way, each spectral bin is constrained to belong to only one harmonic series. This strategy penalizes strings or chromosomes that contain correct F0 values along with spurious multiples or submultiples. For each chromosome a raw fitness value, f_{raw}, is then calculated as the sum of partial fitnesses over all its j substrings:

$$f_{raw}(s) = \sum_{j=1}^{N_S} f_p(s, j). \tag{6.3}$$

The chromosome fitness, $f(s)$, is then computed from the raw fitness as:

$$f(s) = f_{raw}(s) - N_s \overline{f}_p \tag{6.4}$$

where $\overline{f}_p$ is the mean partial fitness over the whole population:

$$\overline{f}_p = \frac{\sum_{s,j} f_p(s,j)}{\sum_s N_s} \tag{6.5}$$

and where N_s is the number of F0s or substrings in the chromosome. The subtraction by $N_s\overline{f}_p$ in Equation 6.4 is a way to penalize strings with too many F0 codes (it is equivalent to subtracting the average partial fitness from each partial fitness) since substrings with partial fitness values smaller than average will become negative and then will penalize the global fitness of the chromosome. Strings with any F0 value outside the allowed range are assigned null fitness.

A final fitness correction step is applied to prevent the premature convergence of the genetic algorithm. This is employed by imposing a fitness floor value F_{min}, such as:

$$F_{min} = \frac{F_{max}}{\beta} \tag{6.6}$$

where F_{max} is the maximum fitness in the current generation, and β is an input positive constant. Individuals whose $f(s) < F_{min}$ have their fitness reset at $f(s) = F_{min}$.

6.1.3 Selection

Each individual is selected for breeding according to the roulette wheel (Goldberg, 1989) selection operator: for each individual in the population a roulette wheel slot is assigned, whose size is proportional to its fitness $f(s)$. Garcia implements the roulette wheel as an array of partial cumulative fitnesses:

$$f_c = \sum_{j=1}^{s} f(j) \tag{6.7}$$

where an uniformly distributed random number r between zero and the total cumulative fitness $f_c(M)$ is drawn and then minimum string index s that satisfies the $f_c(s) > r$ condition is chosen.

6.1.4 Recombination

As recombination operator, Garcia uses the single-point crossover. This operator is designed as follows: two different points of cut are selected - one per individual - since the number of encoded F0s can differ from individuals. This way, two individuals with different chromosome sizes can breed and generate two offspring also, with different chromosome sizes. This operator also ensures that the chromosomes length of the offspring are always multiples of L.

6.1.5 Mutation

The mutation operator consists on flipping single bit in the whole genome of an individual. The probability of mutation (P_n)is given by:

$$P_n = (1 - P_m)^{(N.L)} \tag{6.8}$$

where P_m is the probability of mutation per bit and $(N.L)$ is the chromosome length.

6.1.6 Initialization

The initial population is composed by randomly generated individuals: random number of F0s, each with a random F0 value. Both maximum number of F0s and F0 frequency range are specified inputs.

6.1.7 Survivor Selection

Each new generation consists of individuals selected from the previous generation. This selection is made using the roulette wheel selection operator. Afterwards both recombination (one point crossover) and mutation are applied. Finally, if the current best individual is not as fit as the best individual of the previous generation, the current worst individual is replaced by the best from the previous generation. This strategy is called elitism (Goldberg, 1989).

6.1.8 Experiments and Results

Garcia evaluated his method with synthetic test signals, consisting of N square wave signals plus Gaussian white noise with SNR = 9 dB. N was not known a priori by the Genetic Algorithm, which means that it had two tasks: find the number of signals and their corresponding frequency. The generated test signals had random frequencies, within the allowed frequency range.

For the first test, Garcia used a monophonic input signal and performed 30 runs of the algorithm using $G = 20$ generations and a population size of $M = 100$ strings. The algorithm always found the correct answer: the best individual had only one F0, with the correct frequency.

Next, for the second test, Garcia set an input signal with two fundamental frequencies and a population size of $M = 200$ strings. The algorithm found the best solution in 9 out of 10 runs. The wrong answer actually contained the two correct F0 values, plus a spurious one. There is no information on the manuscript (Garcia, 2001) about the harmonic relation between the spurious F0 and the other two fundamental frequencies.

For the next test, Garcia set and input signal with 4 F0 values and performed two runs consisting of $G = 15$ generations and a population size of $M = 500$ strings. During the first run, the algorithm found an individual with all correct F0 values, plus one spurious value. In the subsequent run, it found a perfect solution.

Afterwards, the algorithm was tested with an input signal with 5 fundamental frequencies. Three different runs were performed with $M = 500$ and $G = 15$. The algorithm constantly found solutions with the five correct fundamental frequencies, plus two spurious ones. Once again, there is no information about the harmonic relation between the spurious frequencies and the correct ones.

One final test was performed as a series of one-run experiments for inputs containing 6, 7, 8, 10 and 15 F0s, with $M = 500$ and $G = 15$. For the input signal with 6 fundamental

frequencies, the algorithm found a solution with 6 correct values and 2 spurious ones. For the input signal with 7 fundamental frequencies it also found a solution containing 6 correct fundamental frequencies and 2 spurious ones. With the input signal with 8 F0s, a solution with 7 correct values and 2 false ones was found. With the input mixture of 10 fundamental frequencies, the solution found by the algorithm consisted in in 5 correct values plus two spurious ones: only 7 F0s. The author then increased the population size to $M = 1000$, and the algorithm increased the number of correct F0s to 8 plus one false. Finally, for the input signal with 15 fundamental frequencies, $M = 1000$ and $G = 15$, the algorithm found 6 correct fundamental frequencies plus two spurious ones.

6.1.9 Additional Constraints

Note that this approach, besides correctly estimating up to 6 fundamental frequencies, does not have in consideration: onset, offset and also dynamics. The algorithm can tell which are the fundamental frequencies present on an audio signal but is unable to detect where those pitches start, where do they end and which are their dynamics. Moreover, by including this three dimensions in the problem, the size of the search space increases exponentially.

6.2 Moving from Polyphonic Pitch Detection to Automatic Transcription of Music

Despite Garcia's approach (Garcia, 2001) being able to work with almost any frequency and resolution, Lu (2007) considers that a polyphonic audio signal is made of by the 128 possible pitches (from the low C, frequency 8.18 Hz to a high G, 12543.88 Hz) defined in MIDI specification (Association, 2008), therefore an audio signal can have up to 128 specific frequencies. As a search space approach, by considering only 128 possible pitches, the size of the search space of a signal with up to 4 fundamental frequencies can be reduced from $1,7549 \times 10^{13}$ to $258.096.640 \simeq 2,58 \times 10^{8}$.

Lu claims that the problem of automatic music transcription is like "reverse-engineering the 'source code' of the music signal" and that it may never be understood. Nevertheless, according to this author, the forward engineering process is known: sound synthesis. Thus, he proposes a method which uses additive sound synthesis, combined with a genetic algorithm, to render each solution or transcription into an audio signal and then compare it with the original audio signal. The result of the comparison is the quality of the individual: how close the transcription is to the desired solution. Lu also states that the main benefit of this kind of approaches is that they are not limited by any particular harmonic model: polyphonic music creates a complex frequency lattice that is computationally infeasible to deconstruct, even for monophonic signals (as reviewed by Gómez et al. (2003)) and by mimicking the process in which the original audio was created, this lattice does not need to be deconstructed but rather be reconstructed.

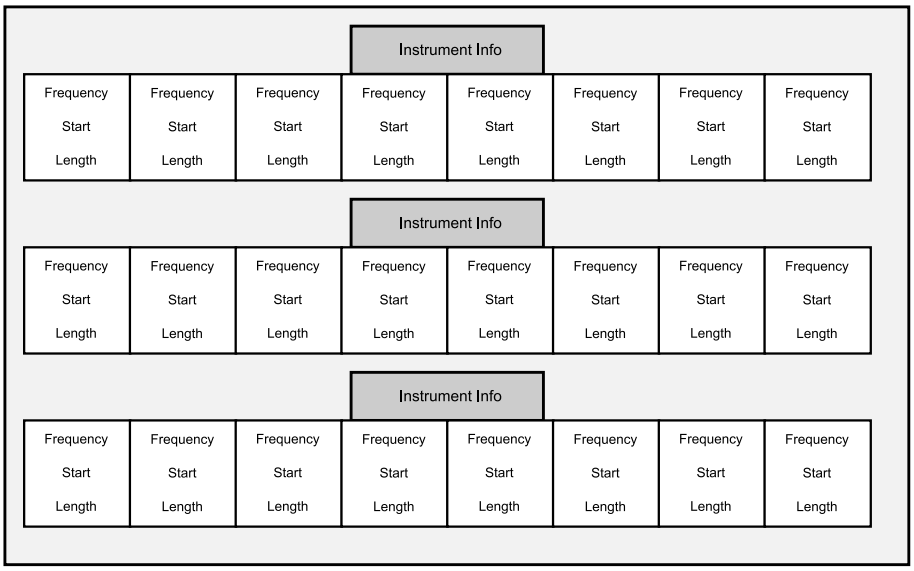

Figure 6.2: Genotype of Lu's approach: notes are separated according to each instrument/track. Each note has frequency, start time and length. Start and length are truncated to time slices.

6.2.1 Genotype

Traditionally, solutions or chromosomes are represented as binary strings of 0s and 1s (such as in Garcia (2001) approach), but other encodings are also possible. Lu encoded each individual as a hierarchical structure, which is not far from the internal representation of a MIDI file (Association, 2008) (see Figure 6.2). Each individual is made of several sequences of notes, organized as tracks, according to each instrument. Each note has a frequency, a start time and length. Both start time and length are truncated to time slices. For instance: for transcribing a set of eight notes, the eighth note should be the time slice.

As mentioned before, Lu (2007) reduces the size of the search space from $1,7549 \times 10^{13}$ to $2,58 \times 10^{8}$ by taking into account just the 128 MIDI pitches. On the other hand, by including three new dimensions to the problem: instrument playing each pitch, start time and duration, the size of the search space becomes much more greater. For instance: for transcribing a signal with duration of 5 time slices and only one instrument playing up to 4 notes, the size of the search space would be: $1,3502 \times 10^{42}$. If the same signal happens to have two instruments, playing up to 4 notes, and same length, the size of the search space is: $1,7961 \times 10^{84}$.

6.2.2 Fitness Function

To evaluate each individual or transcription each transcription is rendered using additive synthesis into an audio signal, which will be compared with the original audio. The result of the comparison (distance) is the fitness value of the corresponding individual. To avoid problems like phase, Lu (2007) proposes that the distance between each individual's transcription and the original audio should be measured in the frequency domain. The

fitness function, similar to euclidean distance, is defined as:

$$Fitness = \frac{1 - \sum_{t=0}^{tmax} \sum_{f=fmin}^{Rochelim} (O(t,f) - X(t,f))^2}{\sigma} \tag{6.9}$$

where $O(t, f)$ is the magnitude of frequency f at time t of the original audio, $X(f, t)$ is the same for the individual's transcription and σ acts as a scaling factor, equivalent to the first worst transcription, putting all fitnesses values between [0,1].

6.2.3 Selection

Although the author of this approach claims that he is addressing music transcription using genetic algorithms, his approach does not use recombination, which is the main pillar of genetic algorithms (Goldberg, 1989). The approach relies exclusively on mutations. This way, individuals are not selected for reproduction.

6.2.4 Recombination

The author of this approach claims that "the genetic material found inside high-fitness individuals is good enough such that most of the material is at least partially correct". According to Lu , the removal of this material for addition into another individual is detrimental for the donating individual. This way, recombination was not included in the approach.

6.2.5 Mutation

Lu applies a roulette selection to determine which mutation will be applied to each individual. The main purpose of this roulette wheel is having some mutations being performed more often that others. The proportions of the roulette wheel also change over time so that mutations that perform small incrementational changes are more often applied during the last generations of the algorithm.

This approach uses the following mutations:

Irradiate Randomly changes one feature (pitch, start time or end time) of a gene.

Nudge Similar to Irradiate, except that changes are on the smallest amount possible: pitch is changed by one semitone, and both start and end time are changed by one time slice.

Lengthen Adds a random musical note to the chromosome.

Split Inserts silence into an encoded musical note. This mutation is capable of deleting a note by inserting a silence with the length of the selected note, shortening a note by inserting the silence on its end or even split the note into two notes by inserting the silence in the middle.

Reclassify Moves a section of the chromosome to a different spot in the chromosome. This mutation allows a set of multiple notes being changed from one instrument to another.

Assimilate Takes a section of the chromosome from one individual and copies it to another individual.

6.2.6 Initialization

The initial population is generated by randomly generated individuals: random number of notes, each with a random start and duration.

6.2.7 Survivor Selection

The top third of the population are copied and then mutations are applied on those copies. These new individuals replace the bottom third (less fit) of the population.

6.2.8 Experiments and Results

In order to be possible to get perfect transcription, Lu rendered several MIDI files using the same synthesizers of the Genetic Algorithm. The resulting audio files composed the test set for the algorithm. Lu also synthesized MIDI files because this way, the ground-truth transcription is known. All the input signals have their difficulty rated by the triplet (x, y, z), where x refers to the number of notes per musical instrument (synthesizer), y is the number of instruments and z is the length (number of time slices).

The system was able to transcribe the monophonic melodies: the monophonic song "Row row row your boat", consisting of 48 notes played by a single instrument - difficulty rating of (1,1,48) -, was successfully transcribed in 233 generations.

For more complete testing, Lu generated several music representations, ranging from (1,1,1) to (3,5,5). For each triplet value in this range, 10 different musical representations were randomly generated. According to the author, *"the results were surprisingly good"*, since for most of the parameter triplets, was able to get at least one exact transcription: only 14 of the 75 triplets were not perfectly transcribed. According to Lu, the biggest set of perfectly transcribed notes consisted in 32 notes, with the corresponding (2,4,4) triplet. When considering only the $(x, y, 1)$ triplets, his system was able to correctly transcribe all trials of the triplet (3,5,1), which means that the algorithm was able to identify 15 overlapping notes, played by a total of 5 instruments.

6.2.9 Additional Constraints

Note that this approach does not have into account the dynamics of each note. Moreover,for the synthesis process, instead of using sample based techniques as Reis et al. (2007), Lu (2007) uses simple and very-well known mathematical models like the sine, square, sawtooth and triangle waves. Therefore, this approach is only able to deal with sounds generated by those mathematical models. Also, the input audio files are MIDI files

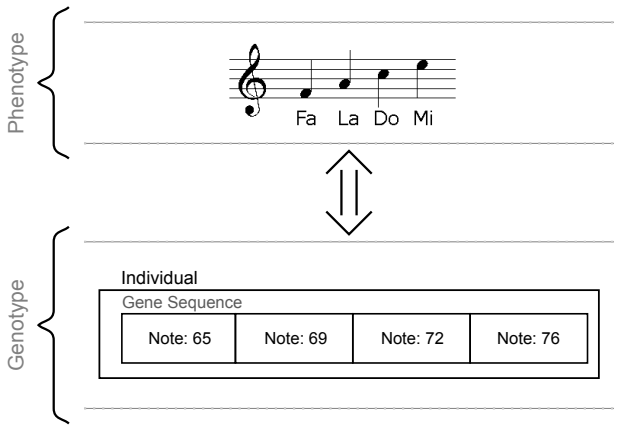

Figure 6.3: Encoding for monophonic transcription. The individual is divided in time frames, where each time frame has can be one of 128 possible MIDI pitches plus the option of silence.

synthesized with the same synthesizers used inside the genetic algorithm, which makes turns the transcriptions much easier to find and without the problem of harmonic overfitting (Reis et al., 2007).

6.3 Automatic Music Transcription using Synthesized Instruments

In 2007 we proposed a new genetic algorithm approach to automatic music transcription, using synthesized instruments (Reis and Fernandez, 2007a). Unlike Lu's method Lu (2007) the synthesized instruments were not simple mathematical models (sine, sawtooth and triangle waves) but, instead, synthesized instruments (piano and vibraphone). Similarly as Garcia Garcia (2001), we proposed a genetic algorithm with recombination, mutation and crossover operators for pitch detection. The latter approach only takes into account the possible 128 MIDI pitches, just as it happens with Lu's Lu (2007) algorithm.

6.3.1 Genotype

We started proposing a system for monophonic pitch detection and later upgraded it to support polyphonic audio signals, just with minor adjustments (Reis and Fernandez, 2007a). The encoding for the monophonic transcription task is based on the assumption that a signal is divided in several time frames, where there can be one of the 128 possible pitches plus the option of having a silence (see Figure 6.3). For polyphonic transcription of music, we extend the previous encoding to support several pitches at the same time, as shown of Figure 6.4.

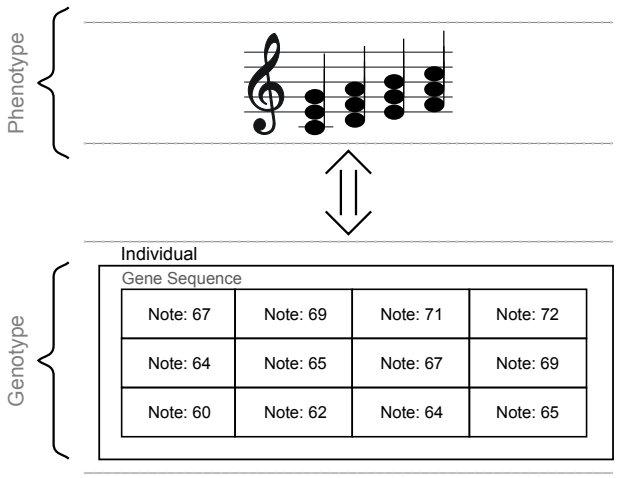

Figure 6.4: Encoding for polyphonic transcription. The individual is divided in time frames, where each time frame has can be one of 128 possible MIDI pitches plus the option of silence.

6.3.2 Fitness Function

Similarly as Lu's proposal (Lu, 2007), the evaluation of each individual is done in the frequency domain, to avoid phase problems, using the STFT. To compare each MIDI-like individual with the target acoustic signal, each individual is rendered into an audio signal using additive sound synthesis. We implemented a synthesizer with the respective oscillator and envelope for the synthesis process (Reis and Fernandez, 2007a). The fitness evaluator renders each MIDI-like individual converting it into an audio signal and then computes it's fitness value by summing the difference between each frequency in each time slice of the song:

$$Fitness = \sum_{t=0}^{tmax} \sum_{f=0}^{fmax} (|O(t,f)| - |X(t,f)|)^2 \tag{6.10}$$

where $O(t,f)$ is the magnitude of frequency f at time slot t in the acoustic audio signal, and $X(t,f)$ is the same for each individual. Fitness is computed from time slot 0 to $tmax$, traversing all time from the beginning to the end, and from $fmin = 0$ Hz to $fmax = 22050$ Hz, which is the nyquist frequency of 44100 Hz sample rate.

6.3.3 Selection

Individuals are selected for breeding with the deterministic tournament (Goldberg, 1989). The size of the tournament is 5.

6.3.4 Recombination

The offspring is generated by applying the classic one-point crossover (Goldberg, 1989) on each pair of parents.

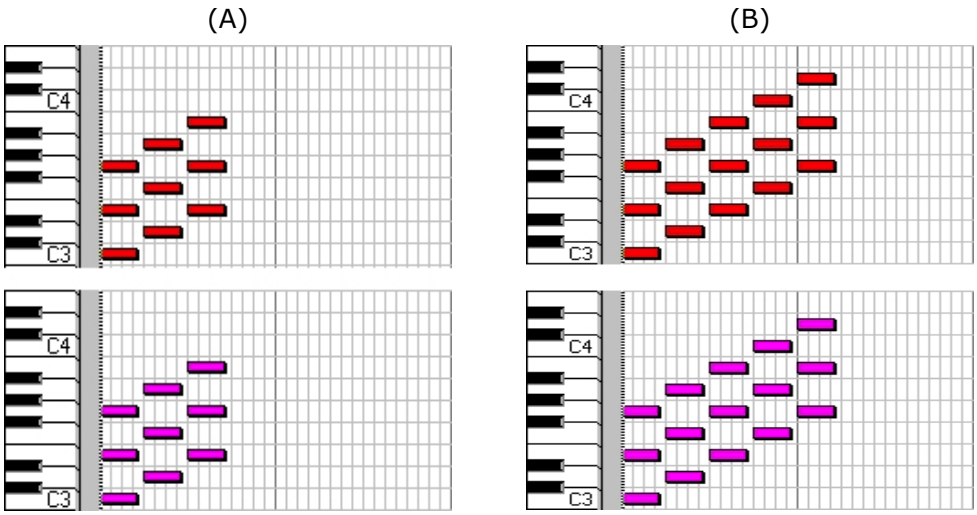

Figure 6.5: Transcription of 3 (A) and 5 (B) consecutive chords. The upper part of the figure corresponds to the original piano-roll and the bottom part is the corresponding generated transcription.

6.3.5 Mutation

We have only implemented a simple mutation that changes the pitch of a random note by -1,1 semitone.

6.3.6 Initialization

As in all the previous works by other authors (Garcia, 2001; Lu, 2007) the initial population consists of random generated individuals.

6.3.7 Survivor Selection

The new individuals generated by recombination and mutation are added to the population. Then, the N fittest individuals (where N is the initial population size) are selected for the next generation.

6.3.8 Experiments and Results

The first experiment consisted in transcribing an audio signal containing for simultaneous notes: the C major triad (C4 + E4 + G4) plus a duplicated root (C5). The algorithm successfully transcribed these notes with a perfect match. Major chords have harmonic related notes, which make them harder to transcribe due to harmonic collisions in the frequency domain. Moreover, C4 + C5 are octave related notes, which means that almost all the harmonics of C5 are also C4 harmonics. This might lead to the transcription error if ignoring C5 and treat its spectral components as components of the C4 spectral envelope.

Two more tests were performed, the first consisted in transcribing three consecutive chords in the C major scale (C3 + E3 + G3; D3 + F3 + A3; E3 + G3 + B3) and another

by adding two more chords to this chord sequence: C3 + E3 + G3; D3 + F3 + A3; E3 + G3 + B3; F3 + A3 + C4; G3 + B3 + D4 (see Figure 6.5). The first chord sequence was transcribed in 6 generations and the second sequence was transcribed in 100. Both sequences were transcribed with 100% accuracy.

6.3.9 Additional Constraints

Although this approach is able to deal with simultaneous notes, it is not able to deal with multiple musical instruments as Lu (2007). This approach can neither work with note dynamics.

Implementation issues

Dealing with audio signals with 44.1 KHz sampling frequency means that each individual, when rendered to audio, will have 44100 samples per second. The API used for implementation, `jMusic` (Brown and Sorensen, 2000), considers each music sample as a single-precision floating-point (32 bits). This way, to deal with 5 seconds of music, each individual, when rendered to audio, will require a memory amount of 44.100 samples $\times$ 5 seconds $\times$ 4 bytes = 1.764.000 bytes. Considering that each generation of the algorithm consists of an initial population of 200 individuals and another 100 individuals (offspring), this means that this implementation requires 529.200.000 bytes $\simeq$ 505 MB of memory. We must bear in mind that we are only talking about 5 seconds of audio: to deal with 30 seconds it would require 3,028 GB per generation.

Another handicap is the fact that `jMusic` (Brown and Sorensen, 2000) deals with each sound sample as a single-precision floating point rather than dealing with it as a `short int`, which has also 32 bits of precision. By dealing with sort int instead of single-precision floating point numbers, the precision would not drop since both have 32 bits and the algorithm would be computationally faster since integer operations require fewer clock cycles than floating point operations.

Given these implementation drawbacks, we rewrote our entire approach from scratch (Reis and Fernandez, 2007b), using the C++ programming language. Moreover, to increase the algorithm performance, our code was compiled using the Intel® C compiler (`icc`) and the signal processing tasks used the Intel® Integrated Performance Primitives (Stewart, 2004; Taylor, 2007). This resulted in significant speedups: the computational required for transcribing both signals in Figure 6.5 was reduced from 6 minutes and 4 seconds to 3,0 seconds (signal A) and from 64 minutes and 43 seconds to 19,523 seconds (signal B).

6.4 Summary

Throughout this chapter we have described the first Genetic Algorithm approaches and presented their main problems. We have presented our first proposal to the problem, by addressing synthesized signals. Our method successfully transcribed with 100% accuracy

the C major triad plus a duplicated root and also a series of major and minor chords along the C major scale.

Synthesized audio files using simple mathematical models are easier to transcribe than real audio recordings, since they do not have inharmonic partials, spurious components, transients and neither have noise. When dealing with real audio recordings, all these problems must be addressed. These aspects will be described during the following chapters.

Chapter 7

Moving to Real Audio Recordings

This chapter presents our first Genetic Algorithm approaches to the problem of Automatic Transcription of Music on real audio recordings. We will also present how each approach addresses the following topics: genotype, fitness evaluation, selection, recombination, mutation, creation of the initial population and survivor selection. Considering that polyphonic real audio recordings have different spectral envelopes for different sources and inharmonic partials, spectral envelope modeling is also introduced.

7.1 First Proposal on Real Audio Recordings

We describe below our first attempt to polyphonic music transcription dealing with real audio data and real instruments (piano recordings) (Reis et al., 2007). Although Genetic Algorithms had already been tried for polyphonic music transcription (see previous Chapter), authors always employed simple simplified audio signals such as mathematical models. To the best of our knowledge, what we present has been the first time Genetic Algorithms face polyphonic music transcription with real audio signals. Since at that time there were only three different genetic algorithm approaches to automatic music transcription (Lu, 2007; Reis and Fernandez, 2007a) and polyphonic pitch estimation (Garcia, 2001), we decided to propose a standard and generic genetic algorithm approach to the problem, which emphasized the considerations discussed in the previous chapter: genotype, fitness evaluation, recombination, mutation, how to generate the initial population and survivor selection.

7.1.1 Genotype

In this new proposal, each individual or chromosome corresponds to a candidate solution (transcription), therefore it is encoded as a sequence of discrete note events. The number of genes (note events) varies from one individual to another. Each gene has all the information needed to represent that note event: note onset, duration, dynamics and

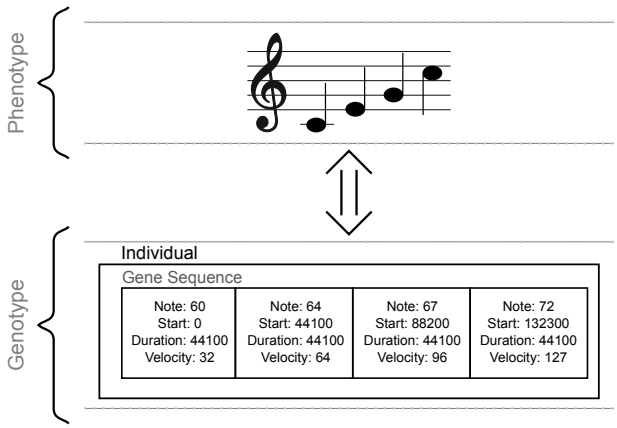

Figure 7.1: Proposed encoding of the individuals. The individual is encoded as a set note events. Each event has a pitch, start time, duration and velocity.

also an instrument/timbre associated with that event, if necessary. Figure 7.1 shows our proposed encoding.

7.1.2 Fitness Function

Since piano synthesis can be easily achieved with sampling methods by playing pre-recorded samples of each piano note, for the evaluation of an individual, each note event passes through an internal synthesizer consisting of previously recorded piano samples. The audio signal generated from each candidate transcription, is then compared with the original audio. The result of the comparison is the fitness value: how close is an individual (transcription) to the input signal.

This way, fitness evaluation consists of two main tasks: synthesis and similarity measure. After rendering each individual into an audio signal, a comparison is made with the original audio. The way audio is compared is the fundamental role on the system, since it measures how close a feasible solution is to the optimal solution. The idea is to have a method that gives importance to aspects that are perceptual and musically relevant and discards other aspects that are not perceptual or musically important. For instance, doing a simple sample comparison (summing the errors between each sample of the original signal and the synthesized one) will most likely have bad results: even two identical signals, but with opposed phase, will be measured as very different.

We propose a fitness evaluation process composed by six different modules (see Figure 7.2):

Pre-Synthesis : note events are processed before synthesis;

Synthesis : the note event sequence is rendered into an audio signal;

Framing : the audio signal is fragment into several audio frames;

Domain : each time frame is converted into a specific domain (e.g. frequency domain);

Post-Processing : post-processing is applied to the obtained values;

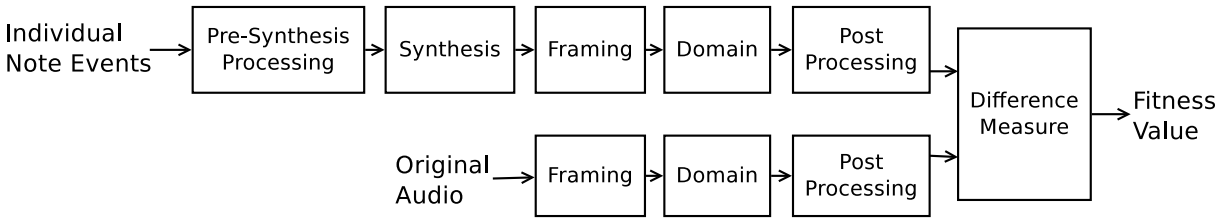

Figure 7.2: Block diagram of the fitness evaluation process.

Difference Measure : applies a mathematical function for measuring the quality of the values (e.g. sum of the squared differences).

Questions

Since this is our first approach on real audio recordings, we faced lot of questions regarding the fitness evaluation like, for instance: "*Which is the best domain to compare two audio signals?*". This way, and in order to address all these questions, all the 6 fitness function modules (pre-synthesis, synthesis, framing, domain, post-processing and difference measure) encompassed several and different features. The idea was to see which are the best options for each module and the best combination of parameters.

The following subsections will describe all the features of each of the fitness function modules. On section 7.1.8 we will describe the best combination of parameters and respective features of each module.

Pre-Synthesis Processing

During pre-synthesis processing, several operations were applied:

- Discard notes whose duration is smaller than n.
- Discard notes whose dynamic values are x dB below the highest dynamic note on the note vicinity;
- Dynamics Average: force all notes to share the same dynamics value by applying the average between all notes;
- Overlapping Removal: prevent two notes with the same pitch to play at the same time;

Synthesis

The internal synthesizer consisted of previously recorded piano 30 seconds samples from a Korg SP100 Piano Keyboard at the made MIDI velocity: 64. The release time of each note was created by applying a fade-out curve to the audio sample of the corresponding

musical note. The decay after releasing the note key was created applying the following equation:

$$R(t) = \max\left(0, \frac{2000.0 - \frac{t}{36}}{2000.0 + t}\right), \tag{7.1}$$

where t varies from $t = 0 \cdots 72000$.

For the dynamic behavior, we considered a dynamic range of 50 dB, that is: one note can vary its dynamic between 1 and 127. In particular, 127 MIDI velocity value correspond to +25 dB gain, 1 MIDI velocity value corresponds to -25 dB gain and 64 MIDI velocity corresponds to 0 dB gain. The gain, according to each note dynamic is given by:

$$gain = 10^{\frac{vel-64}{25}}. \tag{7.2}$$

Framing

The user was able to choose the frame size, overlap and window type: rectangular, Hamming or Hanning (Harris, 1978). The size of the frame is a trade-off between temporal resolution and frequency resolution: smaller frames improve temporal resolution and larger frames improve frequency resolution.

Domain

Each time frame was converted to one of the following specific domains:

- STFT with linear scale;
- STFT with logarithmic scale;
- Filter banks;
- Cepstrum;
- Hybrid (STFT and Cepstrum);
- Autocorrelation (ACF);
- Summary autocorrelation (SACF).

Post-Processing

During post-processing, several rules were applied:

Peaks Only : only data from the local peaks is considered;

Dynamic Range : discards frequency values x dB below the highest frame value;

Normalization : applies a gain factor (up to ±20 dB) to the synthesized frame data to decrease the impact of envelope differences (attack, decay and release) between the original audio and the candidate transcription. This factor could be calculated considering the average frame data, the frame data peak or iteratively, trying several gains with decreasing steps until the best gain was detected to minimize the error;

Data Blur : applies a low-pass filter to blur the frame data. The low-pass filter consists on applying a window (rectangular, triangle or gauss) with a predefined x octave width. The window can either be symmetrical (same frequency width on both sides) or asymmetrical (same octave width on both sides) forms.

Error Measurement

The fitness value is obtained by summing up the frame differences over time t, using frequency f ranging from 27.5 Hz (lowest piano note) to the Nyquist frequency (22050 Hz, which is half of the sampling frequency). The frame error is obtained by using one of the following equations (7.3-7.11). These equations range from the traditional error measurements (equations 7.3-7.5), area interception (equation 7.8), correlation (equation 7.11) and other variations:

$$\sum_{t=0}^{tmax} \sum_{f=27.5}^{22050} \left| |O(t,f)| - |X(t,f)| \right|, \tag{7.3}$$

$$\sum_{t=0}^{tmax} \sum_{f=27.5}^{22050} \left| |O(t,f)|^2 - |X(t,f)|^2 \right|, \tag{7.4}$$

$$\sum_{t=0}^{tmax} \sum_{f=27.5}^{22050} \left| |O(t,f)| - |X(t,f)| \right|^2, \tag{7.5}$$

$$\sum_{t=0}^{tmax} \sum_{f=27.5}^{22050} \frac{\left| |O(t,f)| - |X(t,f)| \right|}{\max\left(|O(t,f)|, |X(t,f)| \right)}, \tag{7.6}$$

$$\sum_{t=0}^{tmax} \sum_{f=27.5}^{22050} \frac{\left| |O(t,f)| - |X(t,f)| \right|^2}{\max\left(|O(t,f)|, |X(t,f)| \right)}, \tag{7.7}$$

$$\sum_{t=0}^{tmax} \sum_{f=27.5}^{22050} \frac{\max\left(|O(t,f)|, |X(t,f)| \right)}{\min\left(|O(t,f)|, |X(t,f)| \right)}, \tag{7.8}$$

$$\sum_{t=0}^{tmax} \sum_{f=27.5}^{22050} \frac{\max\left(|O(t,f)|, |X(t,f)| \right)^2}{\min\left(|O(t,f)|, |X(t,f)| \right)}, \tag{7.9}$$

$$\sum_{t=0}^{tmax} \sum_{f=27.5}^{22050} \log_{10} \left| |O(t,f)| - |X(t,f)| \right|, \tag{7.10}$$

$$\sum_{t=0}^{tmax} \operatorname{corr}\left(|O(t)|, |X(t)| \right). \tag{7.11}$$

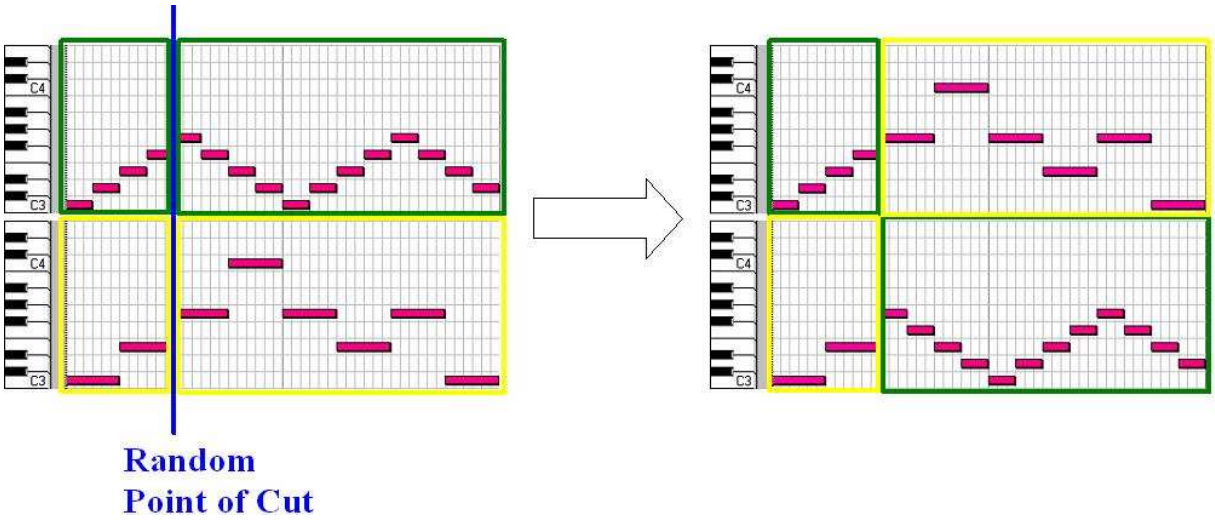

Figure 7.3: One point crossover performed on temporal dimension.

7.1.3 Selection

As in our previous approach (see Chapter 6, Section 6.3.3), the selection for breeding is based on the deterministic tournament, with size of 5.

7.1.4 Recombination

The proposed recombination operator is based on the classic one point crossover (Goldberg, 1989). Instead of choosing a random crossover point in the individuals chromosome, this recombination operator, chooses a random crossover point on temporal dimension. This happens because despite the individuals might differ in the number of genes, they have the same temporal length. The randomly selected crossover point in time will split any note events that cross the chosen time value (see Figure 7.3).

7.1.5 Mutation

We implemented several mutation operators:

- note change (± octave, ± half tone);
- start position (up to ± 0.5 second change);
- duration (from 50% to 150%);
- velocity (up to ± 16 in a scale of 128);
- event split (splits a note event in two events by inserting silence between them);
- event merge (merges two note events having a silence between them);
- event remove;
- new event (random event or duplication with different note).

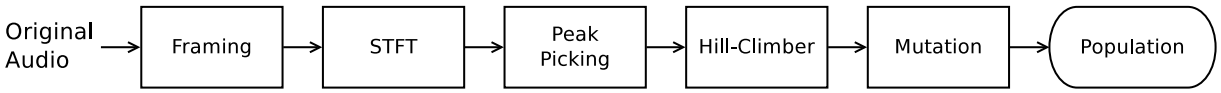

Figure 7.4: Block diagram of the creation of the starting population.

When performing the mutation, one of this mutation operators is randomly chosen and applied.

7.1.6 Initialization

For the starting population, we create a first individual based on the highest peaks of the STFT on the original audio signal. The highest peaks, during each time frame, create (or maintain) a musical note with the corresponding F0s. Afterwards, this individual goes through a hill-climber process that changes all the events equally in terms of velocity, duration and start time to adjust durations and velocity to overcome decay and level differences between the original instrument and the internal synthesizer. Each additional individual in the initial population is created based on the initial individual after 10 forced mutations. Figure 7.4 illustrates this process.

7.1.7 Survivor Selection

After recombination and mutation, some individuals will populate the next generation. The selection of which individuals will pass to the next generation is performed in two steps. The first step consists in selecting the best individual of the current generation and use it directly to populate 5% of the new generation by creating mutated versions of this individual. This extends the robustness of the genetic algorithm, improving the global search by using local search on the vicinity of the best achieved solution (Hart et al., 2004). The second step consists in populate the remainder of the next generation with the fittest individuals of the current generation (elitism).

7.1.8 Initial Experiments and Tuning

Before beginning the experimental step, we ran the proposed system a number of times in order to have an idea about its capabilities. Different combinations of parameters were tried so that we could select an appropriate configuration for each fitness function module (see Section 7.1.2). By assuming that synthesis is not a problem (the internal piano synthesizer or sampler presented a good audio quality), we focused on the analysis of the system ability for searching and measuring similarity.

Pre-synthesis processing

This module revealed to be an important feature of the system, by discarding up many spurious notes. Overlapping notes need to be discarded as well as notes with very small durations (less than 50 milliseconds). Also, note events with small dynamics, when compared to their vicinity notes (below 10 dB), most likely exist due to harmonic overfitting.

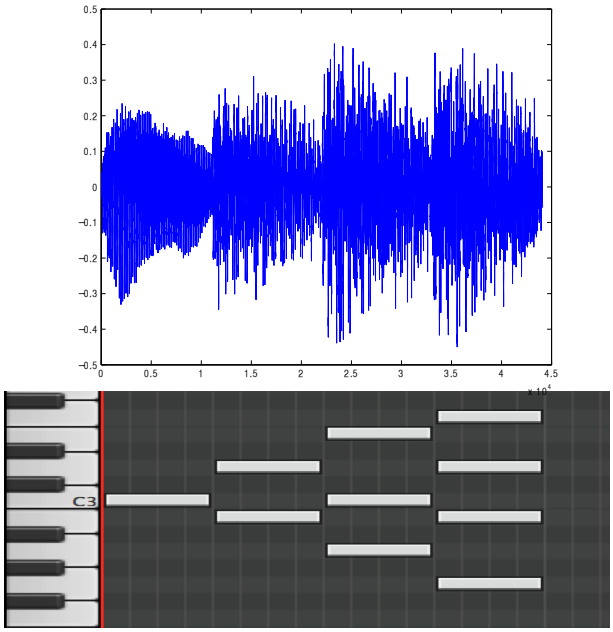

Figure 7.5: Original audio (top) and the piano-roll of the corresponding transcription (bottom)

.

By applying these features on the pre-synthesis module, better results were obtained: precision improved around 15%.

On the other hand, forcing all notes to have the same dynamics values, although it leads to better results when transcribing audio files with a similar behavior (e.g. same MIDI velocity for all musical notes), it is not a realistic feature, and presented worse results on audio recordings with dynamic behavior.

Framing

Tests were done with several parameter values. Frame size ranged from from 512 to 8192 samples and hop size from 0% to 75%. Different windows were also tried: Hamming, Hanning, rectangular, Gaussian, Blackman, etc. The best results were obtained with a frame size of 4096 samples (92 milliseconds at a sampling frequency of 44100 Hz) and a hop size of 1024 samples (23 milliseconds), i.e. 75% overlap. The window function that presented best results was the Hanning window.

Domains

Regarding the best domain for audio comparison, on the contrary to what we were expecting, STFT with linear scale on the magnitude axis presented better results than using a logarithmic scale. We were somehow surprised, considering that the human earing perception has a behavior much more similar to the logarithm scale than the linear one. Probably, the reason why linear scale achieved better results, could be related to the fact that the linear scale increases the importance of higher amplitude frequency components, when compared to the logarithmic scale.

We also had some expectations regarding the SACF domain (Klapuri (2004a), page 27), since it is currently being used in many polyphonic transcription and melody extraction systems (Klapuri, 2008), but, regarding our tests, linear STFT continued to present better results.

Post-processing

The post-processing feature that presented better results was the ability to limit the dynamic range, that is: consider only the frequency components on the top 40 dB range. This way, a better comparison is achieved, discarding spurious spectral components.

Other available features like, peaks only, normalization and data blur did not present better results during our tests.

Error Measurement

The best results were achieved by using the following Equation 7.12:

$$Fitness = \sum_{t=0}^{tmax} \sum_{f=Note21}^{Note108} ||O\,(t,f)\,| - |X\,(t,f)\,||. \tag{7.12}$$

Although the 2D correlation between original audio and the synthesized audio, on a logarithmic STFT domain, presented interesting results, these results were still worse than those obtained with the previous equation.

7.1.9 Experiments and Results

After setting up all the fitness function modules and respective parameters, we focused on the transcription of a single audio file. The tests were based on the polyphonic fragment shown in Figure 7.5, which tries to represent the complexity of a polyphonic audio signal. The base audio signal uses the piano sounds from Microsoft General Midi Synthesizer. This means that although we use piano sounds on the original audio and in the internal synthesizer, they will present differences in their harmonic structure and envelope/release behavior. Table 7.1 shows the parameters of the algorithm.

The quality of the resulting transcriptions was measured using two Information Retrieval measures: Recall and Precision. Precision is the percentage of notes that were correctly transcribed, and is given by the following equation:

$$\text{precision} = \frac{|\{\text{original notes}\} \cap \{\text{transcribed notes}\}|}{|\{\text{transcribed notes}\}|} = \frac{tp}{tp + fp}, \tag{7.13}$$

where tp are the true positives (notes correctly transcribed) and fp are the false positives (notes that were transcribed but are not present in the original audio). Recall is the percentage of correctly transcribed notes present in the original audio:

$$\text{recall} = \frac{|\{\text{original notes}\} \cap \{\text{transcribed notes}\}|}{|\{\text{original notes}\}|} = \frac{tp}{tp + fn}, \tag{7.14}$$

Table 7.1: Algorithm parameters.

Starting Population	200
Number of Generations	1000
Crossover Probability	0.75
Mutation Probability	0.10
Nome Minimal Duration	20 ms
Time Frame	4096 samples
Frame Overlapping	75%
STFT Window	Hanning

where tp are the true positives and fn are the false negatives (number of missed notes).

We have followed the evaluation used by the music information retrieval evaluation exchange (MIREX) (Downie, 2008b; Downie et al., 2010a), that is: a musical note is considered as correctly transcribed (true positive) not only if its frequency is correct, but also if its start time is within ± 50 ms tolerance interval.

Results

The tests shown that, in the end, the system was able to detect all notes (recall 100% - Figure 7.6) although creating some additional wrong notes (precision 81%). Figure 7.6 also shows us that in some runs/generations, the GA was able to find the exact match (achieving 100% in our performance measure), but was unable to keep it.

Although fitness continues to decrease through generations (Figure 7.6 A), after some point, the quality of the results begin to decrease, mainly because of harmonic overfitting. Detected notes continue there, as shown by recall values (Figure 7.6 B), but many additional notes begin to emerge, dropping the precision value (Figure 7.6 C). Since both audio signals use different piano synthesizers (different spectral envelope and decay behaviors), the system begins to create additional notes to compensate those differences in higher harmonics. This means that the fitness function is still needing additional work, since it is not fully working as it should (measuring the right similarity between audio signals).

By comparing the "piano-roll" of the original audio signal (ground-truth) with the best generated transcriptions of each generation, we observe that most errors consist in additional notes that are created in harmonic locations of the original notes, with lower amplitudes and equal or smaller durations. Figure 7.7 shows the piano-roll of the original audio and the generated transcription with the additional low intensity notes on harmonic locations.

We have designated this behavior as *"harmonic overfitting"*, because these additional notes were inserted by the algorithm has a way to decrease the error (fitting) on harmonic locations (harmonic), but using wrong notes (overfitting) since it could not decrease the error using only the right notes.

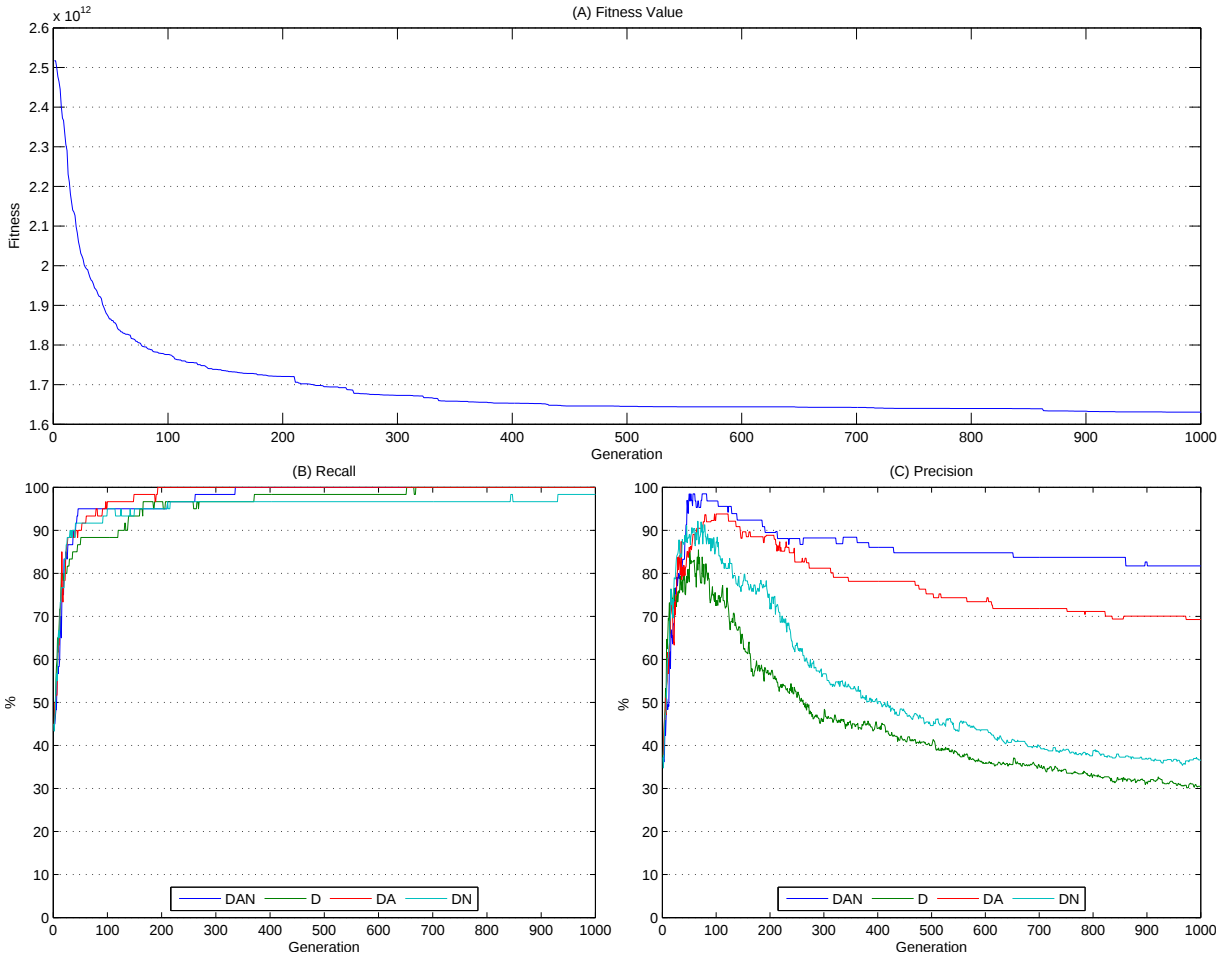

Figure 7.6: Evolution of fitness (A), recall (B) and precision (C) values over 1000 generation, with different configurations.

7.1.10 Additional Constraints

If we make a closer and deeper analysis of this approach, specially on the error measurement module of the fitness function given by equation 7.12, there are two main problems in the fitness function that contribute to harmonic overfitting:

- The fitness function relies on the linear scale instead of relying on the logarithmic scale of the frequency spectrum;
- The fitness function considers that each frequency bin has the same weight during the error measurement.

This way, by considering the linear scale, the system increases the importance of higher amplitude frequency components when measuring the error differences, leading the algorithm to create additional notes in these frequency locations to overcome those differences. Figure 7.8 shows the magnitude spectrum of a middle C or C4 (MIDI note 60) played by a piano in an input audio file (left part of the figure) and the magnitude spectrum of the internal piano sampler of the same musical note. As you can see, there is a big difference in the first harmonic (around frequency bin 50). This means, that the same

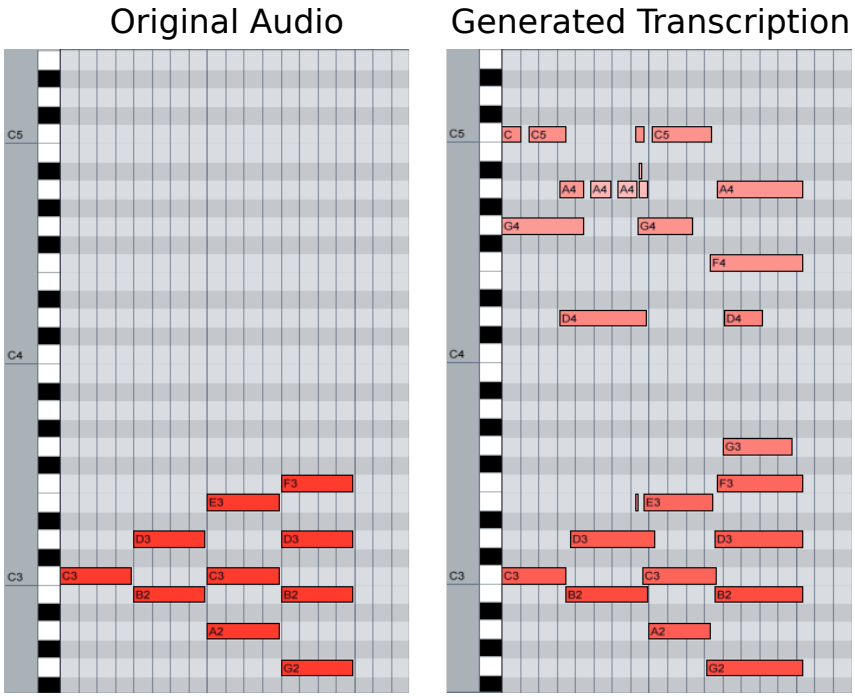

Figure 7.7: Original audio and corresponding transcription.

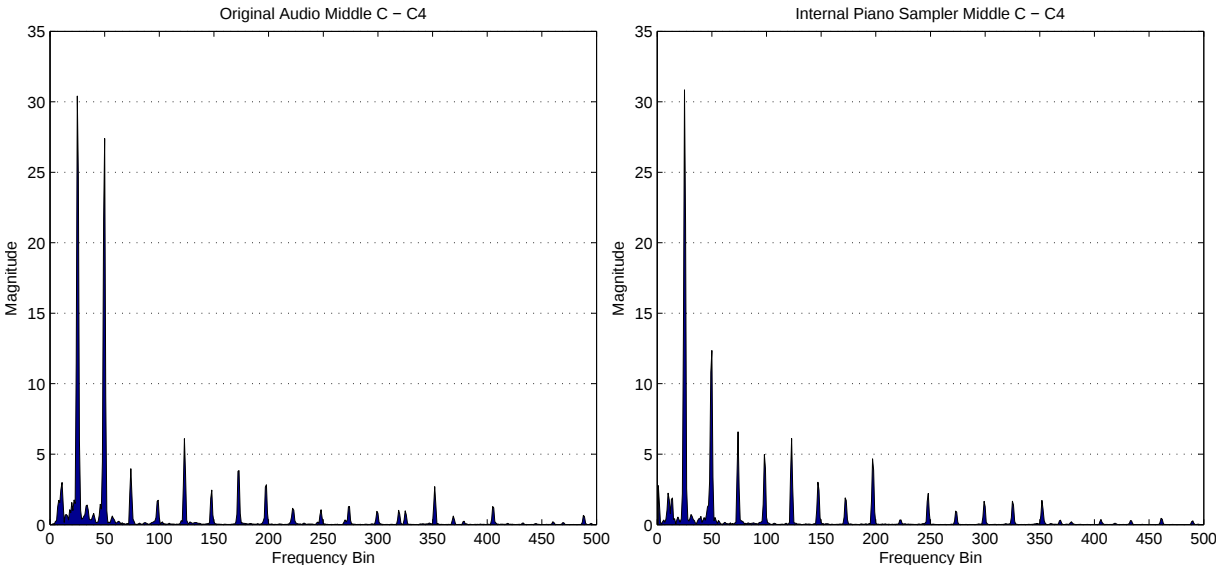

Figure 7.8: Magnitude spectrum from piano played in the original audio (on the left) and from the internal piano sampler, both on the Middle C note (C4).

musical note played by the internal piano sampler will have a big difference on the first harmonic, which will lead the algorithm to create an additional note on that frequency location (musical note C5) to compensate this difference. Thus, the system achieves, as final result, a sound that from the perception point of view is very near the original audio, but since the focus of the problem is to discover only the notes that are played in the original song, the removal of all those additional notes is mandatory.

Moreover, and according to equation 7.12, each frequency bin has exactly the same weight for the error measurement. Considering that the highest octave occupies half of the frequency bins, it means that the higher octave weights the same as the sum of all other octaves. Which means that higher octaves (frequencies) have much higher impact in the error measurement than lower octaves or frequencies. In other words: the current fitness function gives much more importance to higher frequencies, that is: it favors the harmonic overfitting, since it occurs due to differences in higher frequency locations. This stresses the need of a mechanism to perform some kind of frequency normalization so that each octave has the same contribution/weight to the error measurement module.

7.2 Reducing the Harmonic Overfitting

During the approach proposed in the previous Section, we have noticed that the genetic algorithm tends to create additional notes (with lower amplitudes) in harmonic locations of the original notes to overcome the timbre differences between the internal samples and original piano sounds: despite the fitness values continues to decrease through generations, the quality of their results started to decrease after some point, mainly because of harmonic overfitting. The original notes continued there (as shown by recall values) but many additional notes begin to emerge, dropping the precision value.

The fact of these additional notes have low amplitudes and are placed in harmonic locations, most of the times with similar onsets, strongly decreases their impact from the perception point of view. Nevertheless, for the metrics or in situations where the dynamic information is discarded (for instance: creating music sheets), these errors are very undesirable.

To study more accurately the harmonic overfitting phenomena, our system went through several changes. The first change was to include 3 different pianos samplers (Steinway, Bosendorfer and Bechstein piano samplers, from Native Instruments) into the internal synthesizer. This way, we could rely on the spectral difference between different pianos and perform a more accurate study on harmonic overfitting removal. On a second phase of the work, the architecture of the system suffered more architecture changes in order to adapt the internal piano samplers to the piano played on the input audio files. This way, the goal was to be able to evolve, not only the note sequence, but also the synthesizer characteristics for better modeling the original piano sound.

7.2.1 Evolving Timbre

There are several aspects that can differ in two piano sounds. For instance, figure 7.9 shows how the spectral envelope of a piano note varies over time, on two different pianos. This way, it is important to identify what sound features can two piano sounds have, so that the synthesized piano sound could be adapted to the original, if not completely, at least partially, and thus avoiding harmonic overfitting. We consider the following six domains:

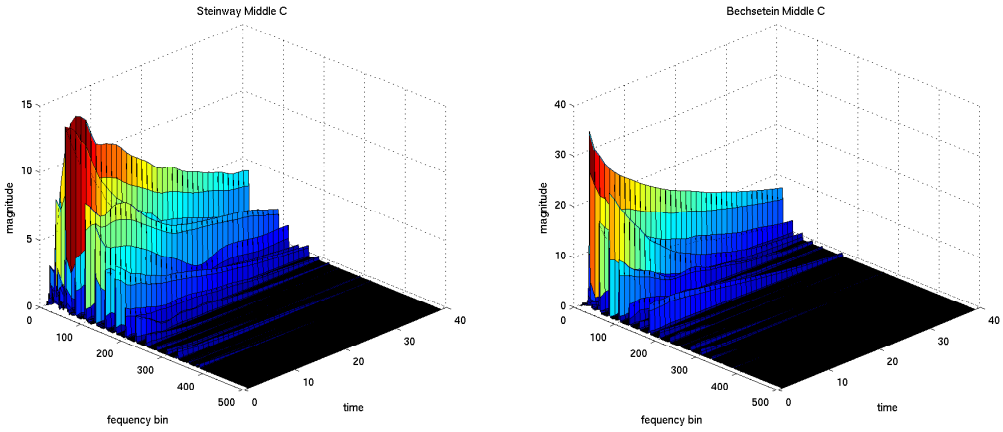

Figure 7.9: Magnitude spectrum of two different pianos (Steinway and Bechstein) playing the middle C (note C4) during one second.

Waveform Envelope : the behavior of the waveform amplitude over time is an important factor. The way the sound attack occurs, how long it takes to decay and how it behaves on note release;

Spectral Envelope : what is the amplitude relation between partials;

Inharmonicy : how much each partial deviates from the closest multiple of the F0;

Dynamics : how both waveform and spectral envelopes change by playing the same note at different velocity values;

Static Ressonances : how different is the behavior of the piano internal resonances;

Recording Frequency Response : how different is the frequency response of the recording process: microphone locations, frequency responses and directivity patterns.

Each one of these six domains represents a dimension of the same sonic space and cannot be handled separately. Waveform envelope, for instance, depends on the spectral envelope, since each sinusoidal component has its own waveform envelope. But, once again, the waveform envelope of each component will also depend on the note dynamics, not only by a constant gain form, but because "*piano*" musical notes have a different timbres than "*forte*" musical notes.

7.2.2 Genotype

Our initial approach begins by considering only the spectral envelope and inharmonicity domains. This way, along with the note event sequence that encodes a candidate transcription, the harmonic structure information is included inside the individual genome (see figure 7.10). This harmonic structure information consists of 19 gains and 19 shifts: the gains and shifts indicate to the internal synthesizer how much gain and inharmonicity deviation should be applied to each one of the first 19 partials above the fundamental

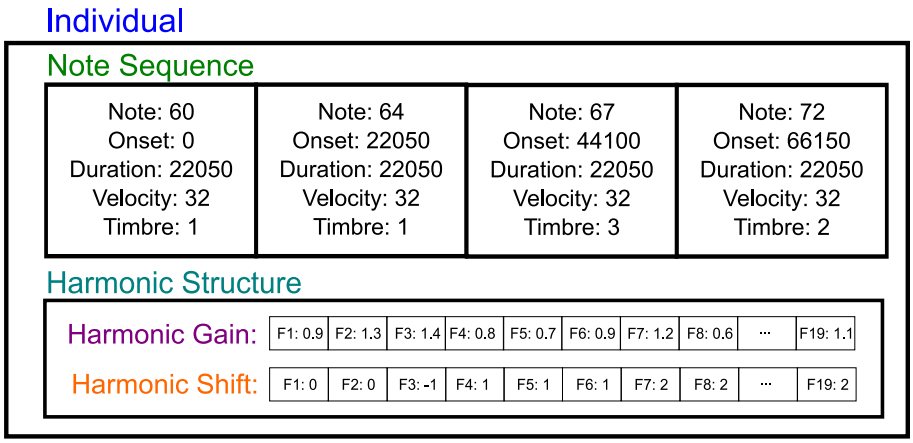

Figure 7.10: Encoding of the individual with the Harmonic Structure.

frequency of the playing musical note. The gain represents the gain (in dB) that should be applied to that partial and the shift represents the amount of STFT bins that that gain should be shifted.

Since the evaluation of each individual is done in the frequency domain (see Section 7.1.8) we apply the gains and shifts directly to the spectrum of each musical note frame, rather than applying filters in each note waveform. In practice, this done by applying different weights on the STFT bins belonging to the note harmonic series. This way, instead of applying filters on each note sample, adding the resulting samples to generate the final audio signal and then process the STFT information based in the polyphonic audio, we use a different approach: for each time frame, the frames of the sound notes were converted to their spectral representation, harmonic gains and shifts are applied and then the spectrum of each note is combined to create the final signal spectra to be compared to the original STFT frame data. This enables each individual to have its own synthesizer, with a complete evolving harmonic structure: the harmonic structure of the internal synthesizer evolves until it matches the synthesizer played on the original audio, and the note events evolve towards the original song's notes.

To combine the spectral data of several notes, some additional tests were made with and without phase information, so we could verify if just the magnitudes were sufficient or if phase values should be taken into consideration, since they have an impact in the final magnitude: two signals with opposing phase result on zero amplitude. As we expected, considering phase values did not bring significant improvements ($\simeq 3\%$).

7.2.3 Fitness Function

Before the error measurement (fitness value) we include two new pre-synthesis features: note discard and dynamic range.

Note Discard

Another feature that we proposed as a means to avoid the harmonic overfitting is note discard. Note discard is based on the assumption that most notes have similar dynamics.

By considering that each note has dynamic scale between 1 and 128 (MIDI velocity range), this feature discards all notes present that have a dynamic difference of 20 between the their dynamics and the dynamics of the other notes existing during the note duration.

Dynamic Range

Harmonic overfitting can also happen due to noise, weak harmonics or even frequency neighborhood. Dynamic range feature uses the highest value of the STFT bins of the current frame as a reference, and sets all bins of the same frame with values 40 dB below this reference to 0.

Error Measurement

Due to the natural logarithmic scale of musical notes, STFT bins are not equally distributed by all octaves (e.g. the highest octave occupies the highest half of frequency bins). To reduce the higher impact of higher notes and, thus, reduce the harmonic overfitting, we perform a division by f for frequency normalization:

$$Fitness = \sum_{t=0}^{tmax} \sum_{f=27.5Hz}^{\frac{fs}{2}} \frac{||O(t,f)| - |X(t,f)||}{f}. \tag{7.15}$$

7.2.4 Recombination

By extending the individual genotype in order to include the harmonic gains and shifts, the recombination operator had also to be extended to support these additional chromosomes: the note events are still recombined using the one point crossover on the temporal dimension (see Section 7.1.4) and both harmonic series and harmonic shifting are recombined using the classic one point crossover (Goldberg, 1989).

7.2.5 Mutation

Besides the mutations already defined in Section 7.1.5, two new mutations were included to support the additional chromosomes:

- harmonic change (up to ± 0.50 gain);
- inharmonicity deviation (up to ± 3 frequency bins).

7.2.6 Experiments and Results

Tests were made with 30 seconds audio extracted from "*Mozart's Pano Sonata No. 17 in B flat (K570)*", played by a Bechstein piano. To test the proposed approach two banks of tests were made using Bosendorfer piano samples or Steinway piano samples inside the internal piano sampler. Since the base audio signal uses different piano sounds from our internal piano samplers, they will present differences in their harmonic structure and envelope/release behavior.

Table 7.2: Results of the proposed approach.

	Recall	Precision	F-Measure	Overlap Ratio
Bosendorfer "onset only"	0.673	0.726	0.699	0.631
Steinway "onset only"	0.659	0.745	0.699	0.631
Mean "onset only"	**0.666**	**0.736**	**0.699**	**0.631**
Bosendorfer "onset/offset"	0.369	0.398	0.382	0.798
Steinway "onset/offset"	0.368	0.398	0.383	0.798
Mean "onset/offset"	**0.369**	**0.398**	**0.383**	**0.798**

Our current approach used the following parameters: population size of 200, in each generation 100 more individuals are breed, only the best 200 pass to the next generation (elitism), the maximum number of generations is 1500, the probability of crossover is 75%, mutation 4% and the note minimal duration is 10 ms.

For better performance, the original audio fragment was split into 5 second fragments and the algorithm was run on each fragment. Each run consisted on 1500 generations. Then, the respective results were merged to create the transcribed 30 seconds sequence. The CPU time needed for transcribing a 5 seconds fragment with 1500 generations is around 7 hours, using one core of a Dual Core 2.0 GHz processor, which still needs improvement.

The used metrics were based on MIREX 2007 (mir, 2007), and consists of "onset only" and "onset/offset" analysis. In "onset only" a note is considered correctly transcribed if pitch is $\pm\frac{1}{2}$ semitone apart and with onset inside $\pm 50ms$ tolerance. In "onset/offset" a note is considered correct if, besides "onset only" requirements, offset is within $\pm 50ms$ or 20% of the note duration (which is the bigger value).

Results are presented using recall (percentage of original notes that were transcribed), precision (percentage of transcribed notes that were present on the original signal), F-measure and mean overlap ratio. F-measure is the harmonic mean (Nostrand, 1962) between precision and recall:

$$F - measure = \frac{2 \times recall \times precision}{recall + precision} \tag{7.16}$$

and mean overlap ratio is the average of notes overlap ratio, that in each corrected transcribed note, measures the overlap between original and transcribed note:

$$OverlapRatio = \frac{\min(offsets) - \max(onsets)}{\max(offsets) - \min(onsets)}. \tag{7.17}$$

We also consider that the same note cannot overlap (e.g. two C4 notes paying at the same time) and that notes with duration smaller than 10 ms are discarded.

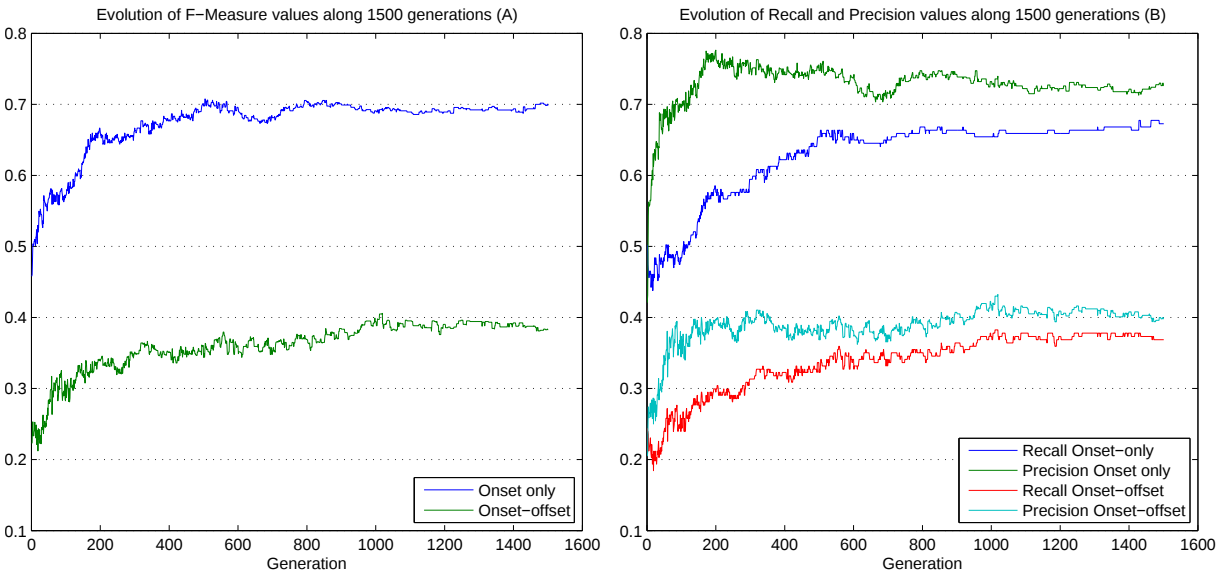

Figure 7.11: Evolution of F-Measure (A) and Reacall and Precision (B) values along 1500 generations.

Results

Table 7.2 shows that both Bosendorfer piano and Bechstein piano present similar results. This means that, despite of algorithm started with different timbres (different pianos) both ended with the same results. This happens due to the harmonic structure evolution: both instruments evolve until they match the original piano. Therefore, the harmonic structure evolution presented very similar results with different pianos, showing that different pianos were able to adjust to minimize timbre differences, making an approximation to the original piano sound.

7.2.7 Additional Constraints

Note that at some point, the F-Measure values decrease (see Figure 7.11 (A)). As a matter of fact, the highest F-measure value during the evolutionary process was 0.7076 for "Onset-only" metric and 0.4059 for "Onset-offset" and the algorithm ended with 0.6286 and 0.3828 F-measure values. This means that the harmonic overfitting still exists, but in a much smaller scale, when compared with our previous approach. The left plot of figure 7.11 shows that, when compared to precision, the recall values on both "Onset only" and "Onset-offset" metrics is improving with few oscillations. This means that detected musical notes continue present in the transcription. On the other hand, Precision has big oscillations. As a matter of fact, Precision in the "Onset only" metric starts decreasing around generation 200. This means that rate of false positives starts increasing, lowering the quality of the results. Those false positive notes are added by the algorithm as a means to decrease the spectral differences between the piano played in the original audio and the internal piano sampler: harmonic overfitting. Some additional notes are present in the original, thus correctly transcribed, and contribute to the recall improvement during the evolutionary process.

In order to avoid the precision values to start decreasing, spectral envelope modeling should be improved. For instance, it should also take into account the note and time: the spectral envelope changes according to the notes played (each note has its spectral envelope) and that envelope changes through time.

This approach has also a performance handicap. Recall that, for better performance, the original audio fragment was divided into 5 second fragments and a different run of the algorithm was processed for each fragment, consisting of 1500 generations. After those 1500 generations the results were merged to create the transcribed 30 seconds sequence. The CPU time needed for transcribing a 5 seconds fragment with 1500 generations is around 7 hours, using one core of a Dual Core 2.0 GHz processor, which is a lot of computational time. This is one of the main issues addressed in the following proposals.

7.3 Automatic Music Transcription of Multi-Timbral Music

All our previous approaches were proposed for transcription of polyphonic piano music. To demonstrate that these approaches were general enough to work with other musical instruments besides piano, and to show the applicability of our system to other musical instruments, we extended our approach to deal with other kinds of pitched instruments, such as: trumpet, saxophone, clarinet and trombone. Basically, both our previous approach and this new one are essentially the same, except that the individuals chromosome now includes the spectral envelope and its inharmonicity deviation for each different timbre.

To extend our previous GA approach to Automatic Music Transcription to support different instruments or voices at the same time, there are several important considerations that need to be addressed:

- How to encode the individual or how the additional information related with timbre should be encoded in the individual's genome?
- How to avoid the harmonic overfitting when different instruments are present in the audio?
- What are the implications of the multi-timbral support for the harmonic overfitting mitigation?
- What new recombination and mutation operators should be created and which ones should be adapted?
- Which samples should be used for each instrument?

7.3.1 Genotype

For the encoding of the individual, we decided to extend the encoding proposed during our previous approach. Several internal samplers and synthesizers (e.g. piano, bass, acoustic guitar, flute, violin, cello, etc.), with samples extracted from the Musical Instrument

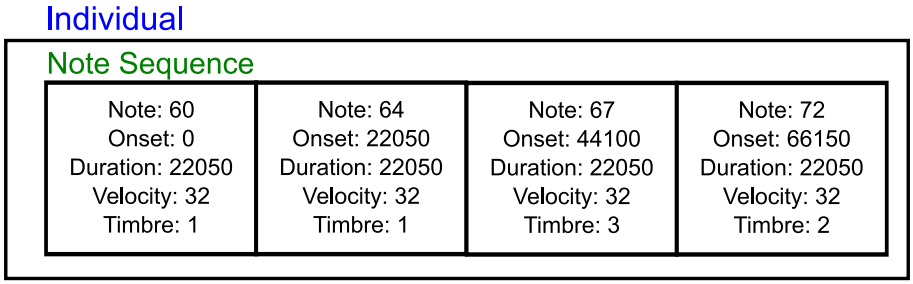

Figure 7.12: Encoding of the individuals with timbre information.

Sound Database - RWC Music Database (Goto and Nishimura, 2003), were included in our system. The information about which instrument plays each note was included in the genotype, on the note event, as timbre information. Figure 7.12 illustrates the current encoding of each individual.

Avoiding Harmonic Overfitting

By including several instrument or voices, the harmonic complexity of the acoustic signal increases exponentially: since several instruments are playing the same music piece, they are most likely playing on the same music scale and also on several octave related notes, therefore there will be several harmonic collisions: a Fundamental Frequency (note) can easily be masked by or even misunderstood as an harmonic of another note, which may lead to greater harmonic overfitting.

Harmonic Series Evolution : Our previous solution to the problem of harmonic overfitting (see Section 7.2.2), was to create harmonic gains, that boost or cut the value of the first 19 partial peaks: almost like an equalizer, but instead of operating on fixed frequency bands, it operates on each note partial. From the implementation point of view, this was not done with real filters, but by changing the values of the STFT bins that correspond to the note partial locations.

This means that each individual, besides having a sequence of note events as their candidate solution to the problem, also includes additional parameters to help the synthesizer to get a timbre more similar with the original instrument. The gain of the Fundamental Frequency of the note - F0 - is always 1.

Inharmonicity Evolution : Sometimes the partials are not located in integer multiples of the Fundamental Frequency. Those partials are often shifted some bins to the left or to right of the real multiple corresponding frequency bin. To solve this problem, the amount of shifting for each harmonic in the harmonic structure was also encoded within the individual's genotype, together with the harmonic structure. This way each individual had his own piano sampler with a complete evolving harmonic structure towards the original piano and also with evolving notes towards the original song's notes. The shift of the Fundamental Frequency of the note - F0 - is always 0.

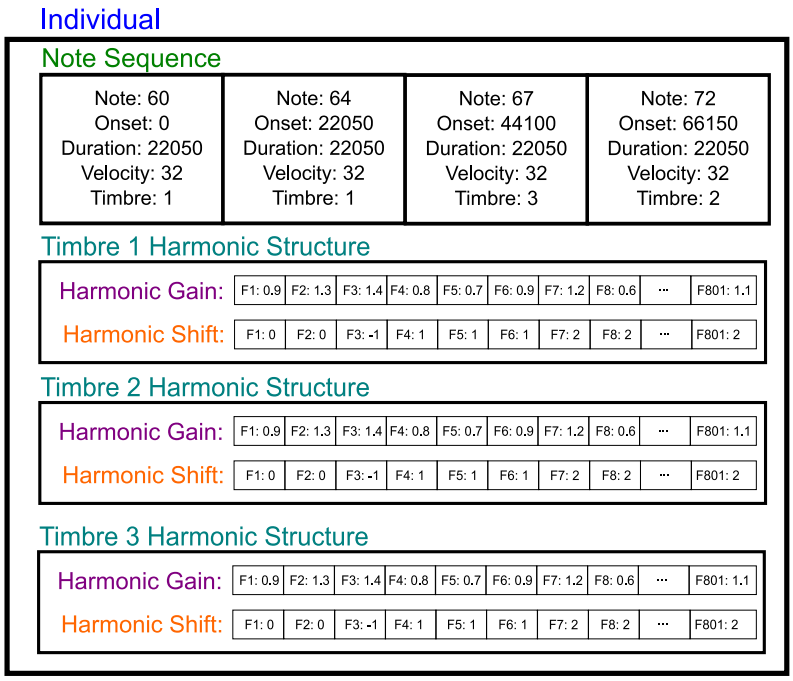

Figure 7.13: Internal structure of an individual.

Multi-Timbre Support

In order to fully support Multi-Timbre and still continue to have both harmonic series and inharmonicy evolution as a mean to mitigate the harmonic overfitting, the harmonic structure of each internal synthesizer was encoded inside the individuals genome (see Figure 7.13). This way, it is possible to avoid the harmonic overfitting in each instrument or voice.

7.3.2 Fitness Function - Individual Evaluation

The error measurement or fitness function is the same defined in Section 7.2.3, Equation 7.15.

7.3.3 Recombination

For recombining the additional data (spectral envelope of each instrument), besides the recombination operator presented in Section 7.2.4, two more random points of cut are used for each harmonic structure: one for splitting the harmonic series and another for splitting the harmonic shifting.

7.3.4 Mutation

Regarding mutation operator, besides the mutations defined in Section 7.2.5, one additional mutation operator was created: change the instrument that plays a musical note. When performing the mutation, one of the mutation operators is randomly chosen and applied.

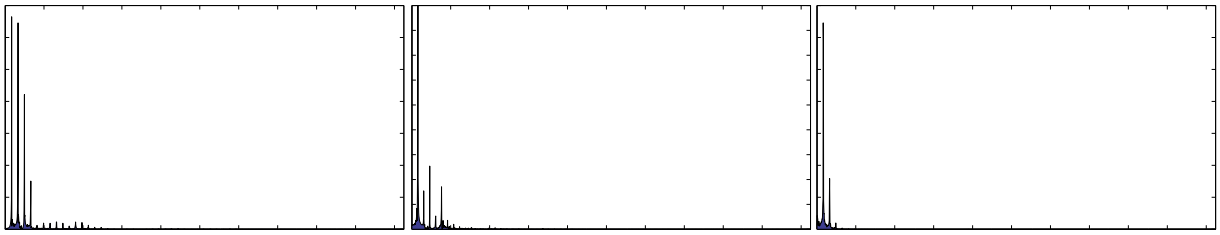

Figure 7.14: Spectrum of the Alto Saxophone, Clarinet and Trombone respectively for the note F4 $\simeq$ 349.23Hz.

7.3.5 Instrument Identification

As for our previous genetic algorithm approaches, our algorithm uses an internal synthesizer so that it can evaluate each individual by comparing its synthesized sound with the original acoustical signal. For the internal synthesizer instead of just using bosendorfer, bechstein and steinway pianos, we used several woodwind, string and piano instruments, extracted from the RWC Musical Instrument Database (Goto and Nishimura, 2003). The timbre information encoded in each gene (note) tells the algorithm which internal synthesizer should be used to render that note. By including the timbre information in each gene, the algorithm is able to choose the correct instrument for each note by changing its timbre property, thus identifying which is the instrument that plays that note in the original signal the one who makes less spectral error.

7.3.6 Experiments and Results

To test the proposed approach and also to have some kind of comparison with a state-of-the-art approach, all the tests were made using the same audio data published on Bay and Beauchamp (2006)[1].

We used the same parameters as our previous approach: population size of 200, in each generation 100 more individuals are breed, only the best 200 pass to the next generation (elitism), the maximum number of generations is 1500, the probability of crossover is 75%, mutation 4% and the note minimal duration is 10 ms.

Transcription of Solo Instruments

Since the test data set has three different instruments with three different spectra (see figure 7.14) the first test consisted in trying to transcribe the solo performances of each instrument. The algorithm was running with the three different internal synthesizers to see if it was able to choose the appropriate synthesizer for each solo performance (Instrument Recognition). Although Instrument Recognition is not the main scope of this dissertation, we wanted to see if the proposed system was able to recognize the instrument played on each note. Figures 7.15, 7.16 and 7.17 show the spectral from both internal synthesizers (left) and original instruments (right) used to transcribe the Alto Saxophone, the Clarinet

[1]The audio data is available on: `http://ems.music.uiuc.edu/beaucham/sounds/sep/sep.bay/`

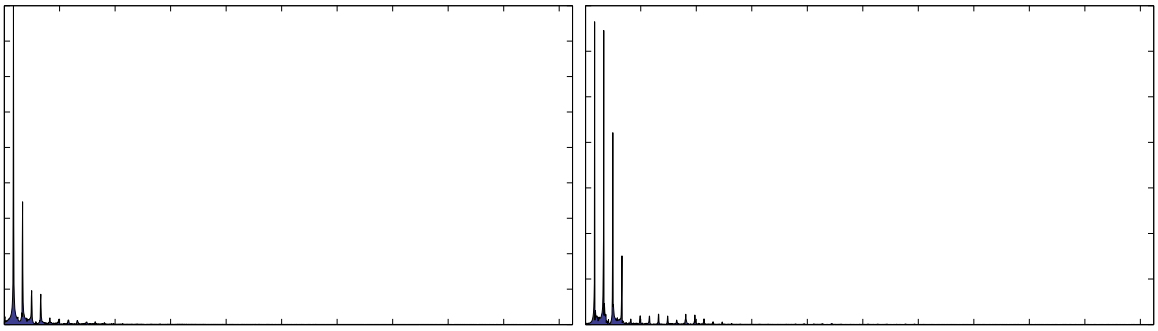

Figure 7.15: Spectrum of the internal synthesizer for alto saxophone (left) and of the original alto saxophone for the note F4 $\simeq$ 349.23Hz.

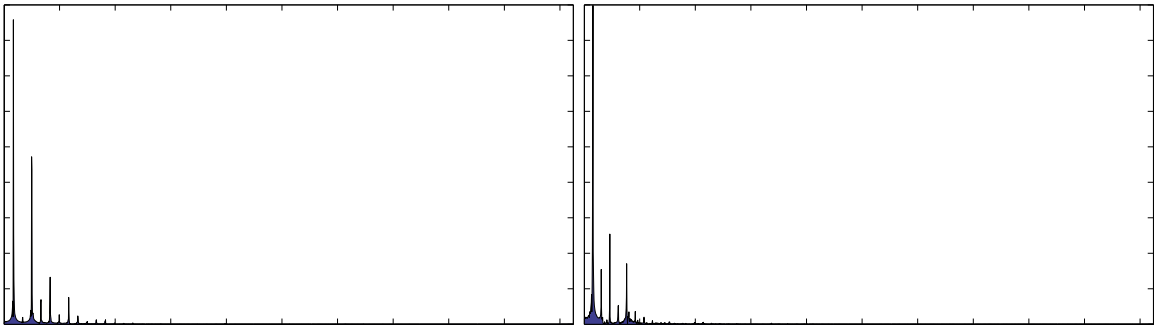

Figure 7.16: Spectrum of the internal synthesizer for clarinet (left) and of the original clarinet for the note F4 $\simeq$ 349.23Hz.

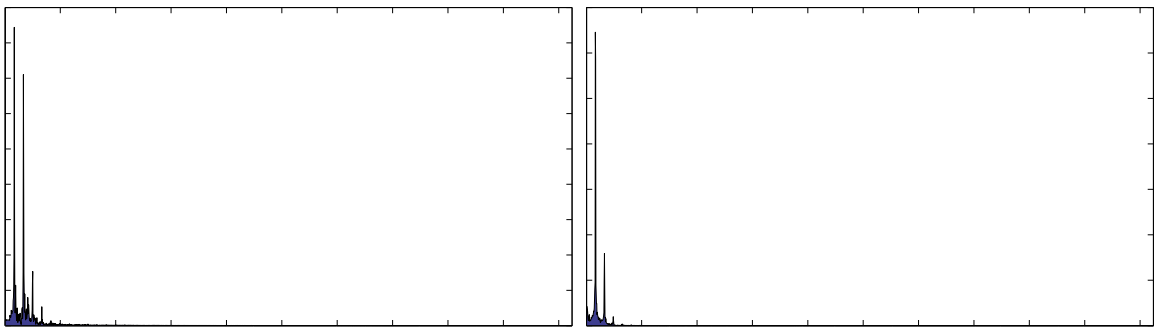

Figure 7.17: Spectrum of the internal synthesizer for trombone (left) and of the original trombone for the note F4 $\simeq$ 349.23Hz.

and the Trombone mixture. You can see that the internal synthesizer for the alto saxophone is somehow similar with the internal synthesizer used for the trombone. This could lead to wrong instrument identification, for instance: the internal synthesizer for the alto saxophone, with the propper harmonic structure (harmonic gain and harmonic shifting) could generate less error for a trombone transcription than using the right instrument with a different harmonic structure or different velocity (note amplitude).

The system correctly transcribed, with 100% of accuracy all the three solo performances and recognized the instrument played in each note with 100% of accuracy. This test was also made to see the performance of algorithm using the RWC Music Database samples as internal synthesizers since those samples are not normalized and some of them

Table 7.3: Initial results of the proposed approach for the whole mixture and for each instrument inside the whole mixture.

	F-Measure	Recall	Precision
Whole Mixture	0.23	0.54	0.93
Alto Saxophone	0.09	0.07	0.13
Clarinet	0.32	0.24	0.47
Trombone	0.21	0.2	0.23

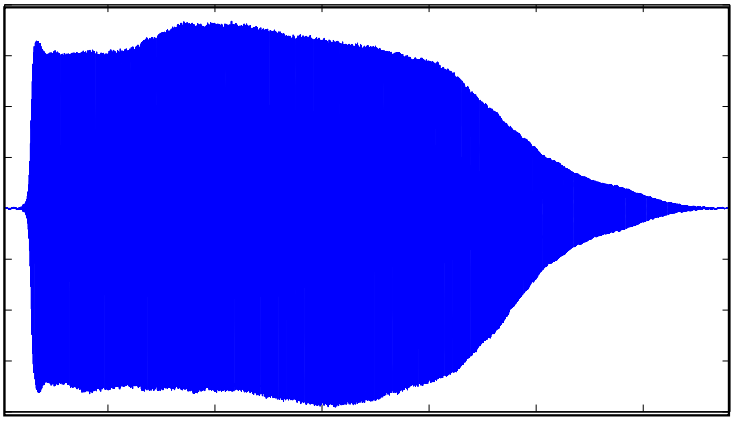

Figure 7.18: Sample used as internal clarinet synthesizer for the F4 note.

have noise, which may lead to a bad synthesis of correct individuals and may lead also to incorrect transcriptions. The fact of the RWC Music Database samples are not normalized is enough for the algorithm to identify the wrong instrument of a note: it may choose an instrument with the closest amplitude (sound volume) to the original note rather than the one with the closest timbre, since the one which has same amplitude generates less error.

Transcription of the Whole Mixture

After successfully transcribing the solo performances, the files corresponding to the clarinet, saxophone and trombone solo performances, were mixed in a single monaural file, which was used as the input file of our system. Table 7.3 presents the results of the transcription of the mixture file. Results are also presented as Recall Precision and F-Measure, which are standard measuring in information retrieval.

Due to the presence of noise and also latency in several samples of the RWC Music Instrument Database, the transcription achieved by the algorithm was shifted to the left to compensate the delay of the instruments samples. Figure 7.18 shows that between the start of the sample and the earing of the sound we have a latency of 125 milisseconds. Since the notes from the transcription are only considered correct if their onsets are within $\pm$ 50 milisseconds interval from the original music score, almost all notes were considered as incorrect. The algorithm shifted the transcription around 125 milisseconds earlier to compensate the delay of the internal synthesizers.

To test if the algorithm was working properly discarding the latency of the inter-

Table 7.4: Results of the proposed approach for the whole mixture and for each instrument inside the whole mixture.

	F-Measure	Recall	Precision
Whole Mixture	0.68	0.54	0.93
Alto Saxophone	0.61	0.46	0.9
Clarinet	0.77	0.64	0.96
Trombone	0.67	0.54	0.88

Table 7.5: Comparison of the algorithms.

		Proposed Approach	Previous Approach
Whole Mixture	F-measure	0.68	0.51
	Precision	0.93	0.66
	Recall	0.54	0.42
Alto Saxophone	F-measure	0.61	0.65
	Precision	0.9	0.71
	Recall	0.46	0.60
Clarinet	F-measure	0.77	0.44
	Precision	0.96	0.69
	Recall	0.64	0.32
Trombone	F-measure	0.67	0.45
	Precision	0.88	0.52
	Recall	0.54	0.4

nal samples, we evaluated the transcription generated by the genetic algorithm with 175 milliseconds of tolerance (125 for the internal samples and 50 milisseconds for the transcription interval). The algorithm obtained a final score of 68% of F-Measure (see Table 7.4), which appear to be very promising results, when compared to the results obtained in the 2008 MIREX contest (Downie, 2008b), where the winning automatic music transcription algorithm (Multiple F0 Estimation and Tracking Task, subtask no. 2) had an F-measure of 61.4%.

One final test was made, running our previous approach (see Section 7.2 on the same data set, as a mean of comparison. Table 7.5 shows the comparison of both approaches.

Despite the proposed method performing better than the previous method when transcribing the whole mixture (it has 68% of F-Measure against 51%), we can see that the previous method performed better on the notes played by the alto saxophone. This happens because the internal synthesizer of the piano has a more close spectrum to the alto saxophone played on the mixture, rather than the spectrum of the internal saxophone synthesizer.

7.3.7 Additional Constraints

If we make a closer and deeper analysis of this approach, specially on the error measurement module of the fitness function, there is still one main problem that contributes to harmonic overfitting: the fitness function relies on the linear scale instead of relying on the logarithmic scale of the frequency spectrum. This way, by considering the linear scale, the system gives more importance to frequency components with higher amplitude when measuring the error differences, leading the algorithm to create additional notes in these frequency locations to overcome those differences.

Also, the spatial complexity of the search space is enormous. There is an almost infinite combination of musical notes, given that they can occur at any time and have any duration. This way, and in order to work properly, the system lacks some kind of mechanism to detect where the musical notes are present (e.g. onset detector) and focus only on those spots of the audio signal to perform the transcription.

7.4 Summary

As previously discussed in Sections 7.2.7 and 7.3.7, the main drawback of the approaches presented during this chapter relies on the spectral envelope modeling, i.e. harmonic overfitting still exists. In order to avoid the precision values to start decreasing at some point during evolution, spectral envelope modeling should be improved by taking into consideration both note and time: different musical notes of the same musical source may have different spectral envelopes, according to their frequency region, and the spectral envelope should be able to evolve through time.

If we look closer to the error measurement module of the fitness functions, there is also another aspect that contributes to the harmonic overfitting: the fitness functions rely on the linear scale instead of the relying on the logarithmic scale of the frequency spectrum. By considering the linear scale, the proposed systems give more importance to frequency components with higher amplitude when measuring the error differences, leading the corresponding algorithm to create additional notes in those frequency locations to overcome those differences. Also, the noise is not being estimated, leading to errors in the transcription.

There is also a performance handicap. During Section 7.2.6 it was described that, for better performance, the original audio was split into smaller fragments and a different run of the algorithm was processed along each fragment. The results of each run were then merged to generate the whole transcription. This was done because the spatial complexity of the search space is enormous: there is an almost infinite combination of musical notes, given that they can occur at any time and have any duration. This way, the described approaches lack some kind of mechanism to detect where the musical notes are present (onset detector) so that they could perform the transcription only on those areas of the signal. Also, the CPU time needed for transcribing a 5 seconds fragment with 1500 generations is around 7 hours, using one core of a Dual Core 2.0 GHz processor, which is a lot of computational time.

Chapter 8

Gene Fragment Competition: Improving the Performance of the Algorithm for Real Audio Transcription

During our research, specially when proposing the approaches presented in the previous chapter, we have noticed that the individuals evaluation demanded a lot of digital signal processing based on an high number of STFTs, which resulted on heavy computational cost. We also realized that, during the evaluation of an individual, it was possible to know which were the better and worst audio fragments of the corresponding individual. In order to solve this problem related to the computational cost, we used the information obtained during evaluation and created a new and intelligent recombination operator. This resulted in a Hybrid Genetic Algorithm, which we baptized as **Gene Fragment Competition**. This new algorithm is general enough to be used for solving other kind of decomposable problems, specially in signal and image processing.

This chapter presents the Gene Fragment Competition algorithm and shows its applicability not only in the Automatic Transcription of Music problem, but also in other kinds of decomposable problems, such as the Royal Roads functions. The presented method is also compared with other classical evolutionary approaches on the Royal Road functions.

8.1 Introduction

Although Genetic Algorithms are very good at rapidly identifying good areas of the search space (exploration), they are often less good at refining near-optimal solutions (exploitation). For instance: when a Genetic Algorithm is applied to the "OneMax" problem[1], near-optimal solutions are quickly found but convergence to the optimal solution is slow. Therefore, hybrid GAs using local search can search more efficiently by incorporating a

[1]The OneMax problem is a binary maximization problem, where the fitness function is simply the count of the number of genes set to "1"

more systematic search in the vicinity of "good" solutions (Hart et al., 2004). For instance: a bit-flipping hill-climber could be quickly applied within each generation for the OneMax to ensure fast convergence. Memetic Algorithms (MAs) are a class of stochastic global search heuristics in which Evolutionary Algorithms-based approaches are combined with local search techniques to improve the quality of the solutions created by evolution (Hart et al., 2004). This means that Memetic Algorithms go a further step by combining the robustness of GAs on identifying good areas of the search space with local search for refining near-optimal solutions. Recent studies on MAs have revealed their successes on a wide variety of real world problems (Hart et al., 2004). Particularly, they not only converge to high quality solutions, but also search more efficiently than their conventional counterparts. In diverse contexts, MAs are also commonly known as hybrid EAs, Baldwinian EAs, Lamarkian EAs, cultural algorithms and genetic local search.

However, a new problem with traditional local search operators arises due to the cost of evaluation. If the calculation of the fitness function is heavy, having local search operators changing each individual several times means lots of individual evaluations and, thus, lots of computational cost. Therefore, as in our problem, where the evaluation of a single individual takes a lot of computational effort, this might be a prohibitive solution.

Fitness evaluation allows us to measure the whole "quality" of each individual, and in many cases, it is obtained by simply combining the values of the evaluations of each gene or gene fragment. In these cases where the evaluation of gene fragments is possible, those fragment evaluation values are not taken on consideration during recombination.

Gene Fragment Competition is a different approach to recombination that takes advantage of gene/fragment evaluation and gene/fragment selection as a way to speed up the process, especially when evaluation of individuals is a demanding computational task. In our specific problem, this is useful given that by recombining the best fragments of each individual, when generating the offspring, the algorithm has a speedup.

8.2 Gene Fragment Competition

In the traditional GA approach (see Figure 8.1a), genetic algorithms are based on a cycle made of evaluation, selection, recombination and mutation: individuals are evaluated, based on their evaluation parents are selected for recombination, creating new individuals that are subject to mutation. On the other hand classical MAs apply a new local search operator in each individual just after the mutation (or even after recombination), looking for better solutions in the vicinity of already found good solutions. Gene Fragment Competition uses a different kind of global/local search approach: instead of using separate operations for global and local search, like the Memetic Algorithms, a different type of recombination is proposed which is responsible for a semi-global/semi-local search.

We can consider that traditional recombination operators are made of two operations: fragmentation (parent's genes are divided on two or more fragments), and merging (these gene fragments are merged to create new individuals). The main idea of the proposed method is to add two additional steps inside recombination: gene fragment evaluation and gene fragment selection. Parent genes are split on n fragments, each fragment is evaluated

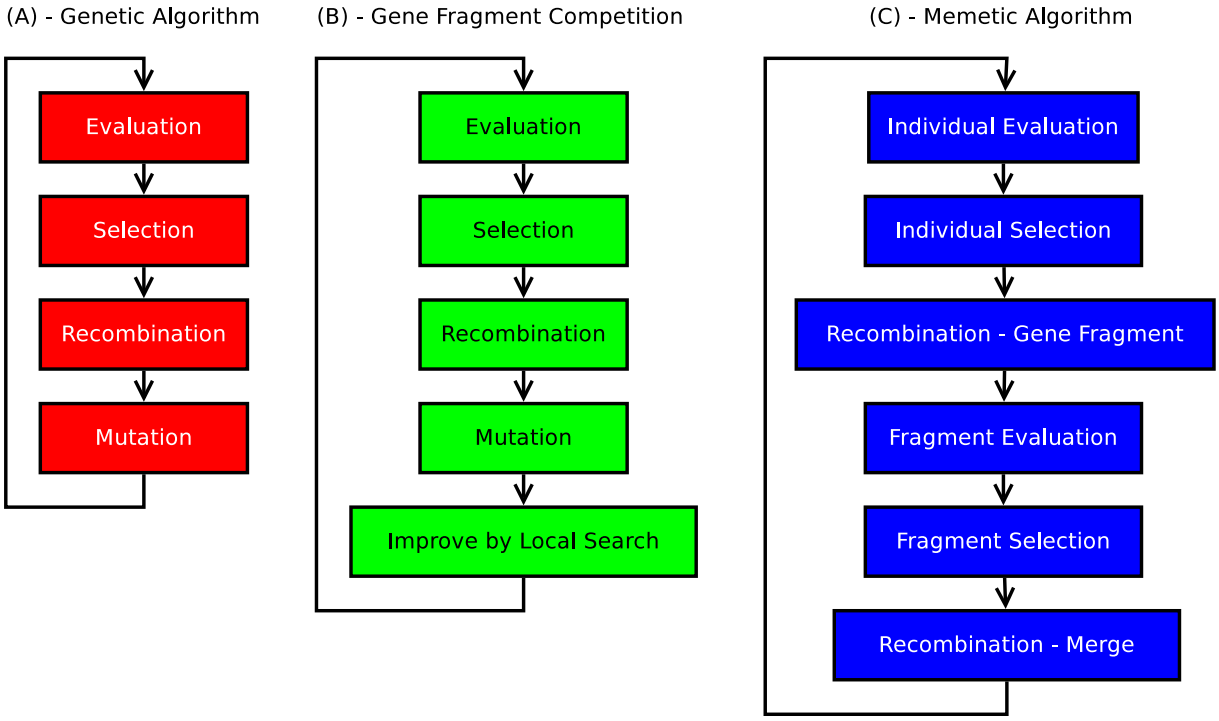

Figure 8.1: Classic Genetic Algorithm approach (A) vs Traditional Memetic Algorithm approach (B) vs Gene Fragment Competition (C).

and then a selection method is applied to choose the best gene fragments, which will be merged to create a new born individual. To split the parents in n fragments two methods can be applied: static fragment size, on which equally sized fragments are created or dynamic fragment size where fragments with random sizes are created. For selecting gene fragments classic selection methods also apply (roulette, tournament, etc.). Although standard recombination operators breed two individuals from two parents, our method breeds only one individual from two or more parents.

As can be seen, a very important requirement must be fulfilled: it must be possible to evaluate gene fragments. If this is not possible, the method cannot be applied. This does not mean that the system must be able to evaluate individual genes, or every kind of group of genes. In some applications it simply means that some special type of fragmentation must be applied to make possible the evaluation of its fragment (see section 8.3). For instance, in signal processing applications or image processing applications we can fragment the individual in time fragments or spatial fragments. In the cases when evaluation is a complex operation a cache feature is highly desirable, for quick evaluation of gene fragments, since its only a matter of adding partial fitness values.

Many of the canonical aspects of evolutionary algorithms are retained under the Gene Fragment Competition (see Figure 8.2), allowing a great flexibility in its deployment. However, each implementation is necessarily application dependent, where the design of a suitable fragment evaluation is a determinant factor. Thus, the following implementation issues have been identified:

Fragmentation The genetic representation of an individual is decomposed in several

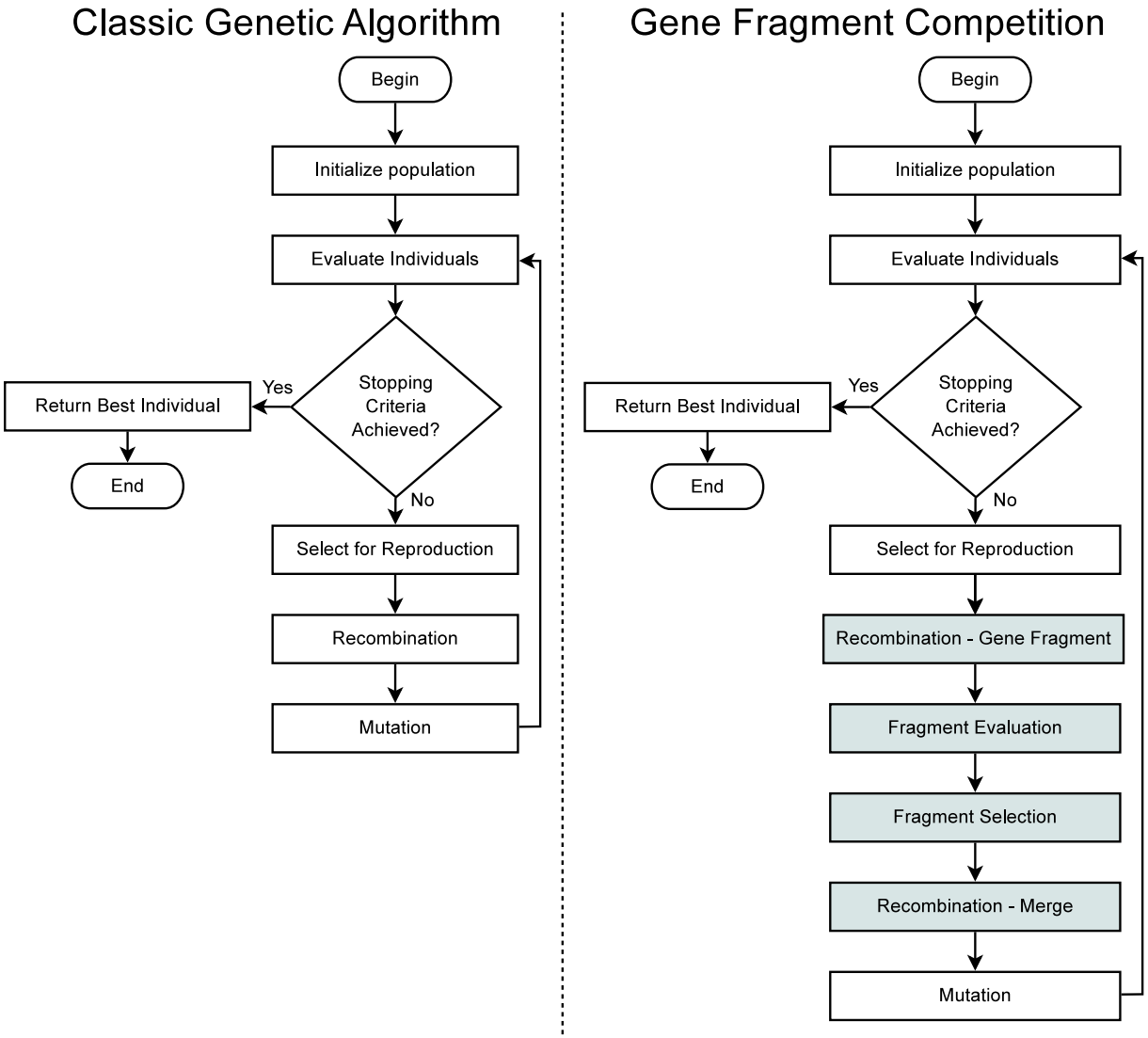

Figure 8.2: Outline of the Gene Fragment Competition, compared with the classical GA. During the recombination each parent is fragmented, the resulting fragments are evaluated and the best fragments of each parent are selected and merged to generate the offspring.

fragments so that they can be evaluated independently. The Fragmentation process can use either static fragment sizes, where equally sized fragments are created, or dynamic, which creates fragments with random sizes.

Fragment Evaluation A meaningfull merit function must be designed for each fragment. In this way, the worthiness of a single fragment can be evaluated in order to estimate the potential contribution to the global solution.

Fragment Selection - Recombination Engine The breeding of the offspring should promote the emergence of better aggregate solutions. The evolutionary engine requires a scheme for aggregating the whole fitness value from partial ones.

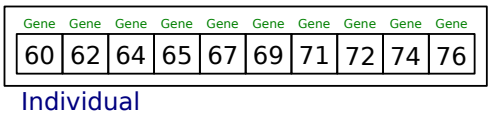

Figure 8.3: Individual's encoding on the "Find the sequence problem".

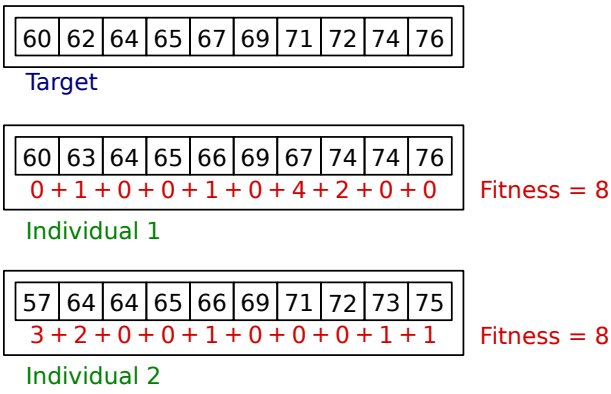

Figure 8.4: Fitness values of Individual1 and Individual2. The fitness value of each individual is calculated by the sum of the absolute difference between the values of their genes and the target individual's genes.

8.2.1 Simple example

Imagine that our goal is to create an individual which is an exact copy of a target sequence of integer numbers. Therefore the individual's encoding could be an array of integers, where each gene would correspond to each sequence number (see Figure 8.3).

Let's consider that the fitness value is obtained by the sum of the absolute differences (Equation 8.1) of each individual's gene ($X(i)$) and the our target individual ($O(i)$) (see Figure 8.4).

$$Fitness = \sum_{i=1}^{genes} |O(i) - X(i)| \tag{8.1}$$

Important note: since the best individuals are those who are closer to our target and the fitness function is measuring that distance, the best individuals are those who have lower fitness values.

If we want to breed a new Individual from the parents Individual1 and Individual2 with 2 random crossover points (dynamic fragment size) our intelligent recombination operator will calculate the fitness value of each fragment and then choose for the new born individual the fragments with the best fitnesses (see Figure 8.5).

8.3 Applying Gene Fragment Competition to Music Transcription

To apply the proposed operator to the music transcription problem, for instance the approach presented in Chapter 7, Section 7.1, there are some remarks. The requirement

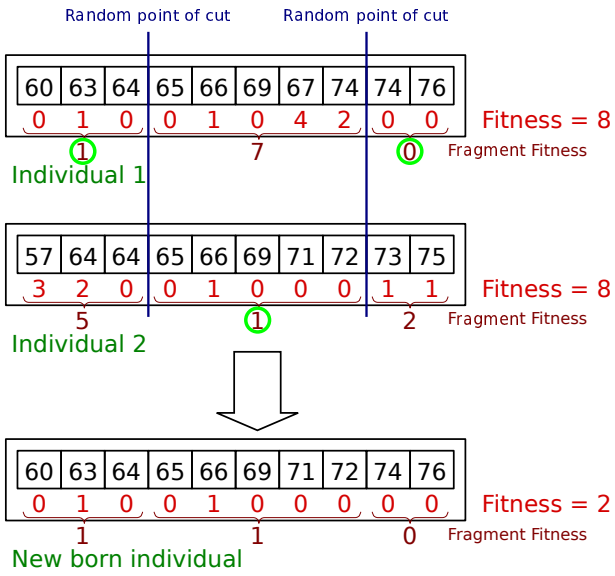

Figure 8.5: Breeding of a new born individual. The best fragments of each father are inherited to the new born individual.

needed for gene fragment competition is that it must be possible to evaluate genes or gene fragments. In the music transcription approach presented before, each gene represents a musical note, and it is not possible to evaluate each note, especially in polyphonic parts. Nevertheless, there is a solution for that. The overall fitness of each individual is obtained by adding the STFT differences over the different time frames, which means that although it is not possible to evaluate note by note, it is possible to evaluate time frames (for instance, it is possible to evaluate the behavior of an individual on a time fragment between time=2.0s and time=4.0s). This way, it is possible to map that time interval to the genes (notes) acting on that time fragment. If some notes on that fragment began before or end after the time frontiers, the note is split, and only the inside part are considered. Later, during the recombination merge phase, if a note ends on the exact same time that a similar one begins, notes are merged as one, since the algorithm considers that a previous split happened.

For global selection, the "deterministic tournament" method was used, but in fragment selection a "non-deterministic tournament" method was used as a means to preserve some biodiversity. Regarding fragment size, that in this case is measured in seconds since we consider time fragments, the value of 5 seconds was used. Increasing the fragment size should decrease the impact of the operator, and decreasing fragment size has the side effect of splitting too much the notes.

For each individual, a cache feature was implemented, that stored the fitness values for each time frame, which means that evaluating a time fragment is simply done by adding fitness values of its internal time frames.

Table 8.1: Algorithms parameters.

Common Parameters	
Population	100
Survivor Selection	Best 100 individuals (population size)
Crossover Probability	75%
Mutation Probability	1%
Note Minimal Duration	20 ms
Classic GA	
Selection	Deterministic Tournament (5 individuals)
Recombination	1-point time crossover (with eventual note split)
Gene Fragment	
Individual Selection	Deterministic Tournament (5 individuals)
Fragment Selection	Tournament (5 individuals)
Fragment size	5 seconds[3]

8.3.1 Experiments and Results

To analyze the impact of the presented method, several tests were made. The proposed method was applied on our music transcription approach on an audio file with the first 30 seconds of the piano performance of the Schubert's Impromptu No.2 in E Minor. Each set of tests consists in 1000 generations, and at least 4 different runs[2]. The values presented correspond to the average values.

In the first set of tests, there were 3 different scenarios: classic GA approach, static fragment size and dynamic fragment size of the proposed method (see Table 8.1). In these tests (and on the following ones), the presented values are the fitness values. Since our goal is not to evaluate our music transcription approach, presenting other results (% of transcribed notes, etc) would remove the focus from the Gene Fragment Competition.

Table 8.2 and left part of the Figure 8.6 show the fitness evolution (average fitness values over several different runs) over 1000 generations. Once again, it is important to recall that in our implementation, since fitness measure the error between STFT's, the lower values of fitness, means better individuals.

The first test, shows us that the proposed method achieves a better performance in this scenario. Nevertheless, there are other situations that needed to be tested in order to discard other hypothesis. One question that could rise is regarding the "Tournament" vs. "Deterministic Tournament" selection. A new set of tests was run using the classic GA, but changing the selection mode from "Deterministic Tournament" to "tournament". The obtained results were identical within a range $\simeq 0.1\%$. The other question that could rise is related to mutation probabilities. Since the proposed method fragments the genes, could it be that changing the mutation probabilities of the classic GA could result in much better results? A new set of tests was made with classic GA approach with different mutation probabilities (5%, 1% and 0.05%). Figure 8.6(right) and Table 8.2 show the results. Using different mutation probabilities above (5%) or bellow (0.5%) did

[2]The value differences between runs are very small ($< 3\%$)

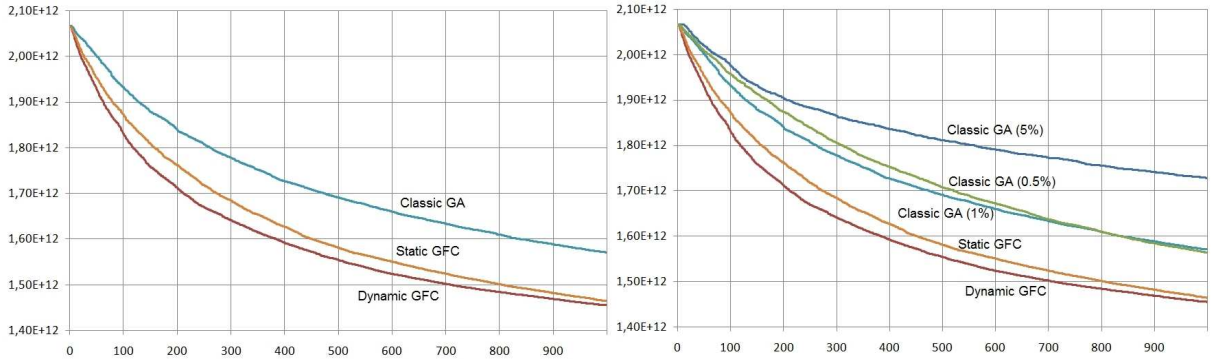

Figure 8.6: Fitness values over 1000 generations using classic GA, using static fragment size gene competition and dynamic fragment size gene competition (left) and with different values of mutation probabilities in generic GA approach (right).

Table 8.2: Average best fitness values over 1000 generations using classic GA, using static GFC and dynamic GFC.

Generation	GA (1%)	Dyn GFC	Static GFC	GA (0,5)%	GA (5%)
250	1,808 E12	1,670 E12	1,718 E12	1,839 E12	1,882 E12
500	1,690 E12	1,554 E12	1,580 E12	1,708 E12	1,811 E12
750	1,621 E12	1,492 E12	1,512 E12	1,624 E12	1,765 E12
1000	1,571 E12	1,455 E12	1,464 E12	1,563 E12	1,728 E12

not present significant improvements.

Tests were also made with another audio file to confirm the earlier results. A 30 seconds audio file of Mozart's Piano Sonata n. 17 in B flat K570[4] was used and is presented in Figure 8.7. Once again the proposed method presents an increase of performance. In this test the performance difference between static and dynamic fragment size also increases comparatively with the initial test.

It is important to say that in both tests by applying our operator with dynamic size, we have achieved in only 500 generations the same results the classical GA achieves in 1000 generations, which is a very significant gain in performance. Moreover, this new method employ less computational cost than the standard GA since each recombination operator only generates one offspring, which in turn results in less computing time.

Although the Gene Fragment Competition presents some requirements that are not fulfilled on several GA applications (capability of evaluating fragment of genes), it is shown that at least in some scenarios can achieve an important performance increase.

8.4 Gene Fragment Competition: a Deeper Analysis

Throughout the rest of this chapter, we apply two decomposable and cooperative approaches: *Gene Fragment Competition* and *Parisian* approach ("Individual Evolution") which exploit the modularity and hierarchical structure of the Royal Road functions (see

[4]Audio files are available at: `http://www.native-instruments.com/index.php?id=apsounds`.

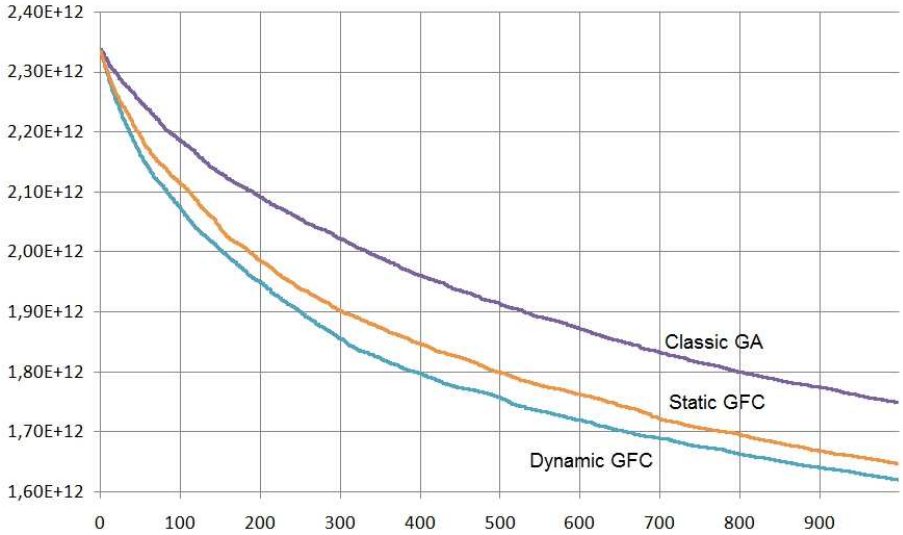

Figure 8.7: Average best fitness values over 1000 generations using classic GA, using static fragment size gene competition and dynamic fragment size gene competition for the Mozart's Piano Sonata.

Section 8.4.3). We will empirically show that both approaches overcome standard genetic algorithms, previous studied multiple-population co-evolutionary approaches, and most importantly, the random mutation hill-climber on Royal Road functions. The optimal number of separation components for each approach is found, as well as the ideal population size for each algorithm. An explanation for each of these numbers is also proposed.

The Parisian Approach is included in this study because, besides being a very well known decomposable approach, it also has some characteristics in common with the Gene Fragment Competition like, for instance, local and global evaluation.

8.4.1 Comparing Gene Fragment Competition and Parisian Approach

Traditionally, evolutionary algorithms (EAs) encode each individual as a possible solution to the whole problem. In other words: each individual represents the entire problem. As a natural extension to standard EAs, problem decomposition emerged for addressing complex problems. Although many problem decomposition methods rely on dividing the main problem in several less complex sub-problems, launching independent populations (species) to solve each of them, there are also other problem decomposition approaches that require a single population. Besides Gene Fragment Competition (GFC), Parisian approach (Collet et al., 2000)(often called Individual Evolution) is also a single-population problem decomposition approach, where the problem can be decomposed in smaller sub-problems, so that they can be evaluated individually reducing the size of the search space.

In Individual Evolution only part of the problem is encoded on each individual, having all individuals in the same population, interacting with each other. This approach is similar to the Michigan approach (Holland, 1975) developed for classifier systems, where each individual represents a single rule and the solution is a rule base made of some of the best individuals of the final population. Each solution (aggregate solution or global

solution) is a set of partial solutions (individuals). Parisian approach can then be considered as a single population approach to cooperative co-evolution: despite being on the same population, the individuals interact between themselves in order to evolve a better aggregate or global solution. This stresses the need of two fitness measures (local and global) so that the individuals can be evaluated individually (local fitness) and also how they interact with each-other as an aggregate solution (global fitness).

As stated above, Gene Fragment Competition is a more recent concept and differs mainly from the Individual Evolution on the fact that each individual encodes the whole solution. Despite those differences, GFC employs a recombination mechanism where the parents's genes are fragmented and evaluated at fragment level. Thus, similarly with Parisian approach, GFC also employs two kinds of fitness evaluation: local evaluation or partial solution evaluation (fragment fitness) and global evaluation (individual fitness).

While Parisian approach is restricted to problems where the solution can be decomposed into homogenous elements or components, this is not mandatory in GFC as long as individuals can be evaluated in fragments.

We take advantage of the modular and hierarchical structure of the Royal Road test functions to adapt them to both Individual Evolution and Gene Fragment Competition. It is our claim that these functions may serve in the theoretical studies of single-population problem decomposable approaches, such as the Parisian and Gene Fragment Competition, since the landscape can be varied in a number of ways, and the global optimum and all possible fitness values are known in advance. Besides presenting a comparison between a standard genetic algorithm and both Individual Evolution and Gene Fragment Competition on several instances of the Royal Road functions, it is also presented a comparison between these single-population problem decomposition approaches and the results of a previous study made by Ochoa et al. (2007) about multi-population co-evolutionary approaches to the same Royal Road functions. One final comparison was made against the Random Mutation Hill Climber. Both Individual Evolution and GFC explore all the possible problem decompositions with modular Royal Road functions. Our results show that Parisian approach and Gene Fragment Competition overcome not only the multi-population co-evolutionary approach but also the Random Mutation Hill Climber in all functions.

8.4.2 Parisian Approach

The main difference from traditional approaches to evolutionary computing relies on a single individual in the population representing only a part of the problem solution. As previously mentioned Parisian approach is similar with the Michigan approach (Holland, 1992a), which was developed for classifier systems since it requires an aggregation of multiple individuals in order to obtain a solution to the problem being studied. This favors the evolution of the whole population instead of the emergence of only a single dominant solution. The main motivation behind this approach is to make an efficient use of the genetic search process. First, the algorithm discards less computational effort at the end of execution, while considering more than a single best individual as output. Second, the computational expense of the fitness function evaluation is considerably reduced for

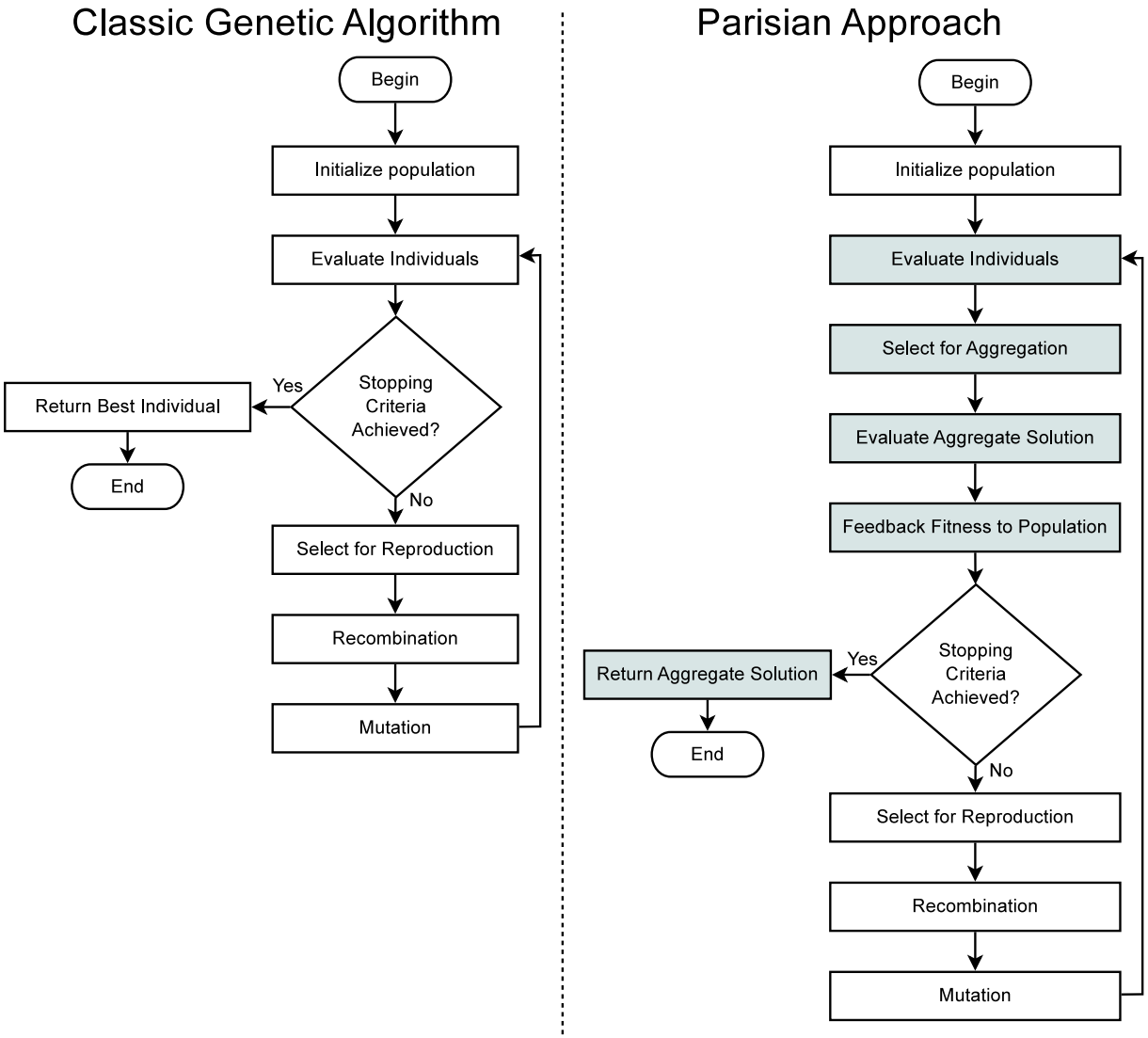

Figure 8.8: Outline of the Parisian approach, compared with the classical GA, done by Dunn et al.[2]. Fitness evaluation takes into account the local and global contribution of an individual.

a single individual. Successful applications of the Parisian approach can be found in the image analysis, signal processing and also in data retrieval applications (Collet et al., 2000; Louchet et al., 2002; Dunn et al., 2005; Landrin-Schweitzer et al., 2006; Dunn et al., 2006; Olague and Puente, 2006).

Many of the canonical aspects of evolutionary algorithms are retained under the Parisian approach (see Figure 8.8), allowing a great flexibility in its deployment. However, the applicability of this algorithm is restricted to problems where the solution can be decomposed into homogeneous elements or components, whose individual contribution to the complete solution can be evaluated. Therefore, each implementation is necessarily application dependent, where the design of a suitable problem decomposition is a determinant factor. Thus, the following implementation issues have been identified:

Partial Encoding The genetic representation used for a single individual encodes a par-

tial solution.

Order Dependency Although in previous implementations of Parisian approach the aggregation process was not order dependent, the order of the individuals (partial solutions) was not relevant for the global solution, there are some specific problems where order plays a very important role. For example: An aggregate solution of $\{Ind_1, Ind_2, Ind_3\}$ might not be the same as $\{Ind_3, Ind_1, Ind_2\}$.

Local Fitness A meaningfull merit function must be designed for each partial solution. In this way, the worthiness of a single individual can be evaluated in order to estimate the potential contribution to an aggregate solution.

Global Fitness A method for the aggregation of multiple partial solutions must be determined. In turn, a problem defined fitness function can be evaluated from this complete solution. However, the worthiness of this composite solution should be reflected on each partial solution.

Evolutionary Engine The evolution of the complete population should promote the emergence of better aggregate solutions. The evolutionary engine requires a scheme for combining local and global fitness values. It may also require a diversity preserving mechanism in order to mantain a set of complementary partial solutions.

8.4.3 Royal Road Functions and the Hitchhiking Phenomena

The building-block hypothesis (Holland, 1975) states that a genetic algorithm performs well when short, low-order, highly-fit schemata (building blocks) recombine to form even more highly-fit, higher-order schemata. The ability to produce fitter and fitter partial solutions by combining building blocks is believed to be the primary source of the GA's search power. Although there have been recent criticism, empirical evidence, and theoretical arguments against the building-blocks hypothesis (Reeves and Rowe, 2003), the study of schemata has been fundamental in our understanding of genetic algorithms. The first counter-evidence against the building-block hypothesis came from Holland himself, in collaboration with Mitchell et al. (1992); Forrest and Mitchell (1993) when they proposed the Royal Road functions which emphasize one feature of the fitness landscapes: hierarchy of schemata.

The Royal Road functions were created to demonstrate the superiority of the genetic algorithms over local search methods such as hill-climbing and then prove the usefulness of the recombination. Surprisingly, the results from their investigation demonstrated the opposite: the random mutation hill-climber significantly outperformed the genetic algorithm on those functions. In the random mutation hill-climber (RMHC) a string is initially generated and then its fitness function is evaluated. Afterwards that string is mutated (bit flip mutation) at a random locus, and a new fitness is calculated. If the new generated string has an equal or higher fitness than the previous one it replaces the old string. This process is iterated until the string reaches the optimal solution or until a

s_1 =11111111**********; $c_1 = 8$
s_2 =********11111111**********; $c_2 = 8$
s_3 =****************11111111**********; $c_3 = 8$
s_4 =************************11111111**********; $c_4 = 8$
s_5 =********************************11111111**********; $c_5 = 8$
s_6 =**11111111**********; $c_6 = 8$
s_7 =**11111111********; $c_7 = 8$
s_8 =**11111111; $c_8 = 8$
s_{opt} =11111111111111111111111111111111111111111111111111111111111111111

Figure 8.9: Royal Road function $R1$. An optimal string is broken up into eight building blocks. The function $R1(x)$ (where x is a bit string) is computed by summing the coefficients c_s corresponding to each of the given schemas of which x is an instance. For example, $R1(11111111\ldots0) = 8$, and $R1(1111111100\ldots011111111) = 16$. Here $c_s = \text{order}(s)$.

maximum number of generations is achieved. This hill-climber algorithm revealed to be the ideal for the Royal Road functions since it traverses the "plateaus" and achieves the successive fitness levels. However, RMHC, like any other hill-climber algorithm, will have problems with any function with many local optima.

In Mitchell et al. (1992); Forrest and Mitchell (1993) the authors also found that despite the contribution of the crossover to the genetic algorithm performance on the Royal Road functions, there was a detrimental role of the intermidate stepping stones (intermediate-order schemata obtained by recombining fit low-order schemata). This is caused by the hitchhiking phenomenon (also called spurious correlation): unfavorable alleles are spread over the population due to an early association with an instance of an highly fit schema. In Mitchell et al. (1994) the same authors described the hitchhiking phenomenon as follows: "once an instance of a higher-order schema is discovered, its high fitness allows the schema so spread quickly in the population, with 0s in other positions in the string hitchhiking along with the 1s in the schema's defined positions. Hitchhiking can in general be a serious bottleneck for the GA."

Royal Road Function $R1$

The function $R1$ (Figure 8.9) is computed very simply: a bit string x gets 8 points added to its fitness for each of the given order-8 schemata ($s_i = 1\ldots8$) of which it is an instance. For example, if x contains exactly two of the order-8 building blocks, then $R1(x) = 16$. Likewise, $R1(111\ldots1) = 64$. More generally, the value $R1(x)$ is the sum of the coefficients c_s corresponding to each given schema of which x is an instance. Here c_s is equal to order (s). The fitness contribution from an intermediate stepping stone (such as the combination of s_1 and s_3 in Figure 8.9) is thus a linear combination of the fitness contribution of the lower-level components. $R1$ is similar to the plateau problem described by Schaffer and Eshelman (1991).

s_1 =11111111**********; $c_1 = 8$
s_2 =********11111111**********; $c_2 = 8$
s_3 =****************11111111**********; $c_3 = 8$
s_4 =************************11111111**********; $c_4 = 8$
s_5 =********************************11111111**********; $c_5 = 8$
s_6 =**11111111**********; $c_6 = 8$
s_7 =**11111111********; $c_7 = 8$
s_8 =**11111111; $c_8 = 8$
s_9 =1111111111111111**********; $c_1 = 16$
s_{10} =****************1111111111111111**********; $c_2 = 16$
s_{11} =********************************1111111111111111**********; $c_3 = 16$
s_{12} =**1111111111111111; $c_4 = 16$
s_{13} =11111111111111111111111111111111**********; $c_5 = 32$
s_{14} =********************************11111111111111111111111111111111; $c_6 = 32$
s_{opt} =11111111111111111111111111111111111111111111111111111111111111111

Figure 8.10: Royal Road function $R2$. Some intermediate schemata are added to the those in $R1$. Namely, $s_9 \dots s_{14}$. $R2$ is computed in the same way as $R1$: by summing the coefficients c_s corresponding to each of the given schemas of which x is an instance. For example, $R2(1111111100 \dots 011111111) = 16$, but $R2(11111111111111100 \dots 0) = 32$. $R2(11111111 \dots 1) = 192$.

Royal Road Function $R2$

In $R2$, the fitness contribution of some intermediate stepping stones is much higher (Figure 8.10). $R2$ is calculated in the same way as $R1$: the fitness of a bit string x is the sum of the coefficients corresponding to each schema ($s_1 \dots s_{14}$) of which a string is an instance. For example, $R2(1111111100 \dots 011111111) = 16$, since the string is an instance of both s_1 and s_8, but $R2(11111111111111100 \dots 0) = 32$, since the string is an instance of s_1, s_2 and s_9. Therefore, a string's fitness depends not only on the number of 8-bit schemas to which it belongs, but also on their positions in the string. The optimum string $x = 11111111 \dots 1$ has fitness 192, since the string is an instance of each schema on the list.

Given the increased number of stepping stones, in Mitchell et al. (1992) the authors were expecting the genetic algorithm to perform better on $R2$ rather than on $R1$, since in $R2$ there is a clear path via crossover from pairs of the initial order-8 schemata ($s_1 \dots s_8$), to the four order-16 schemata ($s_{19} \dots s_{12}$), and from the two order-32 schemata (s_{13} and s_{14}), and finally to the optimum (s_{opt}). They believed that the presence of this stronger path would speed up the genetic algorithm's discovery of the optimum, but their experiments showed the opposite: the GA performed significantly better on $R1$ that on $R2$. In Forrest and Mitchell (1993) they noticed that this bottleneck was caused by hitchhicking.

8.4.4 Methodology

As cooperative and decomposable approaches to the Royal Road functions we used both Parisian approach and Gene Fragment Competition. Besides comparing those two approaches with each-other it is also our aim to make a further comparison with a previous

study made by Ochoa et al. (2007) on the same Royal Road functions since it uses multiple-population co-evolutionary approach and also with the Random Mutation Hill Climber, which is very well known for being the best algorithm performing on the Royal Road functions (Forrest and Mitchell, 1993).

In order to adapt the Royal Road functions to the decomposable setting, a solution string is broken into equally sized sub-strings that contain one (or more) of the original function low-order schemata. Each of these substring represents a problem subcomponent or fragment and it is assigned to an individual in the Individual Evolution and assigned to a fragment in Gene Fragment Competition. To maintain resemblance with the originally proposed Royal Road functions, we used functions (both R1 and R2) with low order schemata (building blocks - BBs) of length 8. Both Parisian and GFC were evaluated by comparing its performance with a standard genetic algorithm on the Royal Road functions ($R1$ and $R2$).

For a realistic comparison with the previous works made by Ochoa et al. (2007) on applying multi-population decomposition on both Royal Road ($R1$ and $R2$), we considered functions with string length L=64, 128 and 256 (functions containing 8, 16 and 32 of those BBs). In other words: we considered several numbers of components/fragments, starting from the minimum of two components/fragments and doubling this number up to the maximum given by the number of BBs in the function. This corresponds to having the Royal Road functions $R1$ and $R2$ decomposed in components/fragments having, respectively, a string length L equal to half of total Royal Road function length (half the number of BBs), down to fragments/components having a string length of 8 (i.e. a single BB or schemata). The tested population size values were also 64, 128, 256 and 512, respectively. The remaining algorithm parameters are the same used in Ochoa et al. (2007), so we can have a true comparison, and were applied to both Parisian approach and Gene Fragment Competition and held constant over the experiments: binary tournament selection, 2-point crossover (rate = 0.8) and bit-flip mutation (rate = $\frac{1}{L}$, L = chromosome length), and 50 runs for experiments.

8.4.5 Parisian approach

The Individual Evolution approach to both Royal Road functions $R1$ and $R2$ consists in breaking the solution string into equally sized-substrings, which contain one (or more) of the original function low-order schemata. Each of those substrings will be encoded in the individuals genome. Since in this problem the optimal string is $x = 11111111\ldots1$, this means that all the individuals composing this aggregate solution will be equal ($11\ldots11$ substrings). Thus, we decided not to use any niche strategy mechanism. Otherwise the fitness of similar individuals would be dramatically reduced and the optimal string would never be found. Regarding the problem specific implementation issues identified in Section 2, we did the following:

Partial Encoding The genetic representation used for a single individual encodes a partial solution. Thus it will encode from order-8 schemata (1 BB) doubling this value up to half of the optimal string size, L.

Order Dependency Since the optimal solution s_{opt} is $x = 11111111\ldots1$, the order of the individuals was not considered relevant when making up the whole solution: all partial solutions (substrings with maximum fitness) will be exactly the same.

Local Fitness The individuals were evaluated according to the Royal Road functions.

Global Fitness The best individuals are selected from the population and then aggregated to compose a single aggregate solution. This solution is then evaluated according to the Royal Road functions $R1$ and $R2$.

Evolutionary Engine As previously mentioned there was no niche strategy mechanism since the optimal string s_{opt} is composed by equal individuals (all of them will be substrings of $(11111111\ldots1)$.

8.4.6 Gene Fragment Competition

The Gene Fragment Competition approach to the Royal Road functions $R1$ and $R2$ consists in assigning each individual to a possible solution. During the recombination process the individuals are broken in several fragments, breaking the solution string into equally sized-substrings, which contain one (or more) of the original function low-order schemata. Regarding the problem specific implementation issues identified in Section 3, we did the following:

Fragmentation The genetic representation of an individual is decomposed in several fragments so that they can be evaluated independently. The Fragmentation process uses static fragment sizes, where equally sized fragments are created. The size of those fragments varies from 8 bit strings (one order-8 schemata) doubling its size up to half of the optimal string size, L.

Fragment Evaluation The worthiness of a single gene fragment is evaluated according to both Royal Road functions $R1$ and $R2$..

Fragment Selection - Recombination Engine The selection of each fragment to generate the offspring is the classical binary tournament.

8.4.7 Results and Analysis

The first set of tests consisted in comparing how each algorithm performed for each different population size. Although it is not fair to compare a classical genetic algorithm with different population sizes, we are comparing three different algorithms: standard genetic algorithm, Parisian approach and Gene Fragment Competition. The population values that gave the best results for each set of experiments are present in Table 8.3. Those values were also used in all other experiments. Notice that whenever the number of components or fragments is the same as the number of building blocks (BBs), the best population size is always 512, except for Individual Evolution with 32 components which could not be

Table 8.3: Optimal population sizes for Standard Genetic Algorithm, Parisian approach and Gene Fragment Competition. The tested population values were: 64, 128, 256 and 512. For Parisian and GFC the best performance over all the numbers of fragments (from 2 up to the block size) was considered. L stands for the Royal Road function's string length. The number of evaluations for the Parisian approach with 32 components (Parisian_{32}) was too big.

	$R1$			$R2$		
	$L = 64$ (8 BBs)	$L = 128$ (16 BBs)	$L = 256$ (32 BBs)	$L = 64$ (8 BBs)	$L = 128$ (16 BBs)	$L = 256$ (32 BBs)
SGA	64	512	64	64	256	128
GFC_2	64	64	64	64	64	64
GFC_4	256	64	64	128	128	64
GFC_8	512	128	64	512	64	64
GFC_{16}		512	64		512	64
GFC_{32}			512			INF
Parisian_2	64	64	512	64	64	256
Parisian_4	64	64	64	64	64	64
Parisian_8	512	64	64	512	64	64
Parisian_{16}		512	128		512	128
Parisian_{32}			INF			INF

determined. In the Parisian approach, having the problem decomposed in several components is the same as having several individuals to form the possible solution. Since the optimal solution is s_{opt} is $x = (11111111 \ldots 11111111)$, all the individuals composing it must be optimal and completely equal. If the population size increases, the probability of having completely equal and optimal individuals also increases. This is why when the number of components is the maximum possible (number of components = number of BBs) the algorithm performs better with the population size also set to the maximum (512). Bigger populations also means more recombinations per generation and Gene Fragment Competition exploits recombination to generate fitter individuals by gathering the best fragments of each parent. Thus, when the number of fragments is maximized, similarly as it happens with the Parisian approach, GFC algorithm performs better with the maximum population size. Almost all the other instances performed better with population size = 64. This happens because with greater population the algorithm tends to have slower convergence, therefore it needs more generations and thus, more evaluations to discover the optimal string.

Although Parisian approach performs better than Gene Fragment Competition in both Royal Road functions (see Table 8.4), its performance, after some point, drops completely when the number of components continues to increase. As previously mentioned, having the global solution decomposed in several components, stresses the need to have that same number of individuals completely equal in order to find the global solution. Therefore if the number of components starts to increase, it also increases the number of the individuals needed to be completely equal and optimal, decreasing exponentially the probability of having that number of optimal individuals among the population: any of them can suffer

mutations, be discarded between generations or even replaced by a less fit offspring. On the other hand GFC performs better when the number of fragments is the same as the number of building blocks.

Table 8.4: Average number of evaluations and standard deviations ($\times 10^2$) to find the optimal string in both $R1$ and $R2$ functions. The results of a previous study in the literature about multi-population co-evolutionary approaches[7] are also included (CCEA), as well as the Random Mutation Hill Climber (RHMC). Both Parisian and GFC methods significantly overcome both CCEAs and RHMC.

		$R1$	
Algorithm	L=64 (8 BBS)	L=128 (16 BBs)	L=256 (32 BBs)
SGA	**211.66 (128.06)**	**654.5 (291.05)**	**2633.54 (896.85)**
$CCEA_2$	165.4 (66.22)	571.4 (270.62)	2161.2 (768.20)
$CCEA_4$	142.2 (71.56)	402.2 (143.18)	1473.9 (577.42)
$CCEA_8$	**114.1 (43.30)**	**327.1 (130.71)**	1094.8 (464.03)
$CCEA_{16}$		365.2 (128.84)	**851.5 (319.73)**
$CCEA_{32}$			1140.9 (352.26)
GFC_2	201.8 (90.58)	624.9 (206.79)	2367.84 (927.23)
GFC_4	149.98 (63.54)	546.04 (314.03)	1843.86 (654.87)
GFC_8	**56.22 (24.07)**	463.84 (207.96)	1760.08 (651.49)
GFC_{16}		**172.2 (121.18)**	1477.16 (423.91)
GFC_{32}			**476.9 (312.23)**
$Parisian_2$	87.62 (67.52)	227.66 (137.21)	689.58 (336.35)
$Parisian_4$	**54.42 (45.39)**	**99.04 (65.88)**	240.34 (136.35)
$Parisian_8$	99.58 (76.88)	195.14 (191)	**136.6 (77.26)**
$Parisian_{16}$		38 543.24 (33 701.63)	115 197.18 (111 628.6)
$Parisian_{32}$			INF
		$R2$	
Algorithm	L=64 (8 BBS)	L=128 (16 BBs)	L=256 (32 BBs)
SGA	**299.1 (158.03)**	**1025.94 (442.72)**	**3643.92 (1902.96)**
$CCEA_2$	173.5 (74.76)	640.8 (301.91)	2480.1 (1141.24)
$CCEA_4$	**115.6 (51.22)**	412.2 (214.98)	1432.1 (472.53)
$CCEA_8$	127.0 (56.36)	**305.7 (92.60)**	1094.5.52 (430.02)
$CCEA_{16}$		399.9 (142.02)	**876.2 (330.46)**
$CCEA_{32}$			1389.2 (365.93)
GFC_2	203.6 (98.74)	702.54 (281.63)	2100.58 (605.92)
GFC_4	149.12 (95.93)	542 (199.61)	1857.46 (684.64)
GFC_8	**59.82 (37.75)**	459.26 (194.48)	1660.52 (469.93)
GFC_{16}		**170.94 (171.87)**	1471.9 (608.44)
GFC_{32}			**576.68 (518.77)**
$Parisian_2$	80.38 (50.44)	319.24 (187.72)	1092.1 (502.79)
$Parisian_4$	**54.42 (45.39)**	**99.04 (65.88)**	330.12 (195.81)
$Parisian_8$	99.58 (76.88)	195.14 (191)	**142 (88.39)**
$Parisian_{16}$		38 543.24 (33 701.63)	115 197.18 (111 628.6)
$Parisian_{32}$			INF

		$R1$ and $R2$	
Algorithm	L=64 (8 BBS)	L=128 (16 BBs)	L=256 (32 BBs)
RHMC	**66.74 (36.57)**	**176.44 (79.55)**	**386.34 (104.4)**

Table 8.4 also shows that GFC and Parisian approach perform very similar in both Royal Road functions, thus both approaches can cope very well against the hitchhiking

phenomenon. This happens because the individuals are split in fragments according to the hierarchical structure of the Royal Road functions. Thus, when recombining individual fragments or aggregating partial solutions, we are only dealing with the fragments themselves and not with the fragments's neighbors genes, which are the most likely positions for hitchhikers. Another interesting observation is that the behavior on Parisian approach in $R2$ only differs from $R1$ when the number of components is two or when the number of components is four and the size of the string L is 256. This happens because, by only evaluating the partial solutions, the fitness of the intermediate stepping stones of $R2$ is discarded. This does not happens in GFC because the individuals are evaluated as a global solution, and so the intermidate stepping stones are taken into account.

The results obtained in Ochoa et al. (2007) using a multiple-population and co-evolutionary approach to the same problem are also included in Table 8.4 (CCEAs). We can see that both Parisian approach and Gene Fragment Competition overcome the multi-population co-evolutionary approach to the problem. The superiority of single-population approaches is due to focusing the global search space inside only one population, which leads to a more effective search. In the multiple-population approach there was no interaction between the individuals of different populations, which means that despite one specie or sub-population might have several sub-optimal individuals or near sub-optimal, the individuals in other sub-population or specie could be far way of the desired solution. By focusing the search space one population all the individuals can interact with each other, recombine and exchange genetic material, which leads to a faster and better search.

Finally, the last experiment consisted in running the Random Mutation Hill-Climber (Forrest and Mitchell, 1993) and then compare its performance with both approaches. This hill-climber algorithm is very well known because it significantly outperforms the standard genetic algorithm approaches on those Royal Road functions. The RHMC, as it happened with the other algorithms, was run 50 times for each test. Results are shown in Table 8.4 (RMHC). By looking at those results we can see that Individual Evolution outperforms the RMHC in all instances, as well as Gene Fragment Competition overcame the Random Mutation Hill-Climber, except the instances with string length $L = 256$ (32 BBs).

8.5 Summary

During our research, we noticed that the individuals evaluation demanded a lot of computation cost due to the high number of STFTs. We also realized that, during the evaluation of an individual, it was possible to know which were the better and worst audio fragments of the corresponding individual. By exploiting this knowledge, we have designed a new recombination operator for solving the problem related to the computational cost and created the Gene Fragment Competition.

We have presented an analysis of how decomposable approaches are suitable to decomposable problems. Moreover, we have taken advantage of the modular and hierarchical structure of the Royal Road functions in order to use them as test functions and see

how single-population decomposable approaches can overcome the spurious correlation or hitchhiking.

Our empirical results have shown that both Parisian approach and Gene Fragment Competition clearly outperform not only the standard genetic algorithm and the multiple-population co-evolutionary approach but also the random mutation hill-climber, except the GFC on the instances with string length $L = 256$.

Hitchhiking is known to be, in general, one of the major bottlenecks of the genetic algorithms performance, thus avoiding hitchhiking boosts the performance of the algorithm. Applying problem decomposition in building blocks appears to be a very advantageous optimization technique.

Despite the random mutation hill-climber algorithm has been revealed to be the ideal for the Royal Road functions in the past, since it traverses the plateaus and achieves the successive fitness levels, we have shown that single population decomposable approaches can explore more efficiently the search space on Royal Road functions.

Chapter 9

Multiple-F0 Estimation on Piano Recordings using Spectral Envelope Modeling and Dynamic Noise Level Estimation

The main drawback of our proposed approaches (see Chapter 7, Sections 7.2.7 and 7.3.7) relies on the spectral envelope modeling, i.e. harmonic overfitting still exists. The spectral envelope modeling should be improved having in consideration that different musical notes of the same musical source may have different spectral envelopes, according to their frequency region. Also, the spectral envelope should be able to evolve through time.

Those same approaches presented also a performance handicap. This way, and for better performance, the original audio was split into smaller fragments and a different run of the algorithm was processed along each fragment. The results of each run were then merged to generate the whole transcription. Due to the enormous complexity of the search, our methods lacked some kind of mechanism to detect where the musical notes are present (onset detector) so that they could perform the transcription only on those areas of the signal.

If we perform a deeper analysis on the error measurement module of the fitness functions, there is also another aspect that contributes to the harmonic overfitting: the fitness functions rely on the linear scale instead of relying on the logarithmic scale of the frequency spectrum. Thus, by considering the linear scale, the proposed systems give more importance to frequency components with higher amplitudes when measuring the error differences, leading the corresponding algorithm to create additional notes in those frequency locations to overcome those differences.

Finally, the noise is not being estimated, leading to errors in the transcription.

This chapter presents a new method for multiple-F0 estimation on piano recordings that aims at solving all the presented problems. We propose a framework based on a genetic algorithm in order to analyze the overlapping overtones and search for the most

likely F0 combination. The search process is aided by adaptive spectral envelope modeling and dynamic noise level estimation: while the noise is dynamically estimated, the spectral envelope of previously recorded piano samples (internal database) is adapted in order to best match the piano played on the input signals and aid the search process for the most likely combination of F0s. For comparison, several state-of-the-art algorithms were run across various musical pieces played by different pianos and then compared using three different metrics.

9.1 System Overview

By analyzing Equation 2.27 or even the equation of a short-time monodic signal:

$$x[n] = \sum_{h=1}^{H} A_h cos(h\omega_0 + \phi_h) + z[n] \tag{9.1}$$

for both $y[n]$ and $x[n]$ modulation, there are three properties that must be considered: (1) the fundamental frequency (ω_0); (2) the spectral envelope (A_h); and (3) the residual noise ($z[n]$). The method we introduce now, works on all these properties: partial tracking, adaptive spectral envelope modeling and dynamic noise level estimation. Figure 9.1 shows the proposed system, which consists of three phases: (1) audio segmentation; (2) transcription process; (3) note duration adjustment.

During the Audio segmentation an onset detector is applied on the input signal to extract onset information. Afterwards, the audio signal is divided into several audio segments according to the detected onsets. Each interval between two consecutive onsets is considered a segment. Then, for each segment, a *thread* is launched running a 50-generation genetic algorithm to perform the corresponding transcription. The search for the most-likely combination of F0s to model $y[n]$ is aided by an internal database consisting of previously recorded piano samples. The genetic algorithm also adapts the spectral envelope of the used piano samples in order to best match the power spectrum of the corresponding audio segment. During this process, residual noise's spectral envelope is also dynamically estimated to favor the search process towards the desired solution (see Figure 9.1). The results obtained on each audio segment are then merged into one whole transcription. Finally, a hill-climber algorithm (Russell and Norvig, 2003) is applied on the global transcription to adjust note duration or to merge notes that transverse several segments. The output of the system is the hill-climber's final result.

Since the main focus of our work is the actual transcription problem and not the onset detection (and also because the lack of accuracy of the onset detector could compromise the performance of the algorithm) the user is able to choose other onset information as an input to the algorithm. The audio will then be segmented in accordance to the supplied information. This way, other onset detection systems that might be more robust can be used and, when dealing with labeled data, usage of the real onsets as input is also possible.

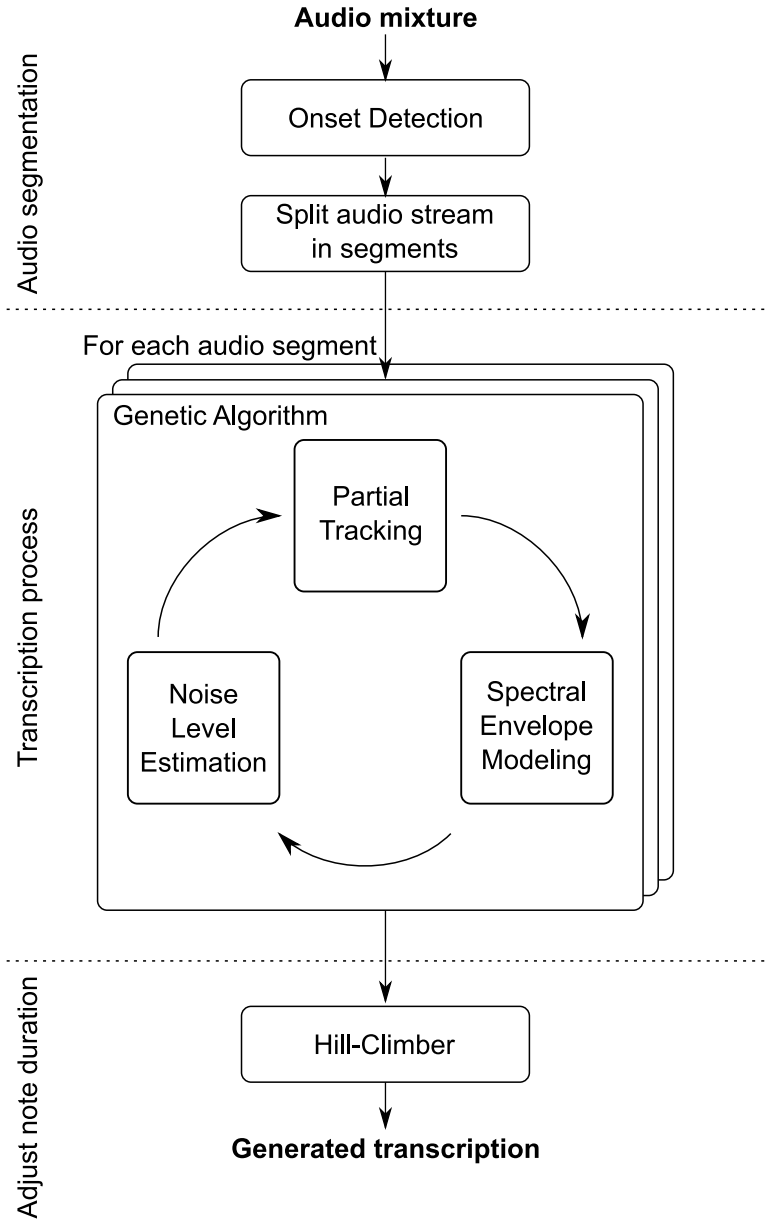

Figure 9.1: Block diagram of the transcription algorithm.

9.2 Proposed Genetic Algorithm

Since the problem being solved is the automatic transcription of an audio segment, a candidate solution must be a candidate transcription of the corresponding segment. We consider a transcription as a set of musical notes where each note has four attributes: start time, duration, MIDI note and also MIDI velocity. Therefore, an individual is encoded as a chromosome with a set of genes where each gene corresponds to a musical note (see Figure 9.2 (A)).

Despite the onset being fixed to its segment boundaries, the onset information needs to be included into the chromosome so the Hill-Climber can operate properly: after the transcription of each onset-synchronous segment, those transcriptions are merged into one whole transcription (new individual). At this stage, the onset time differs from each gene.

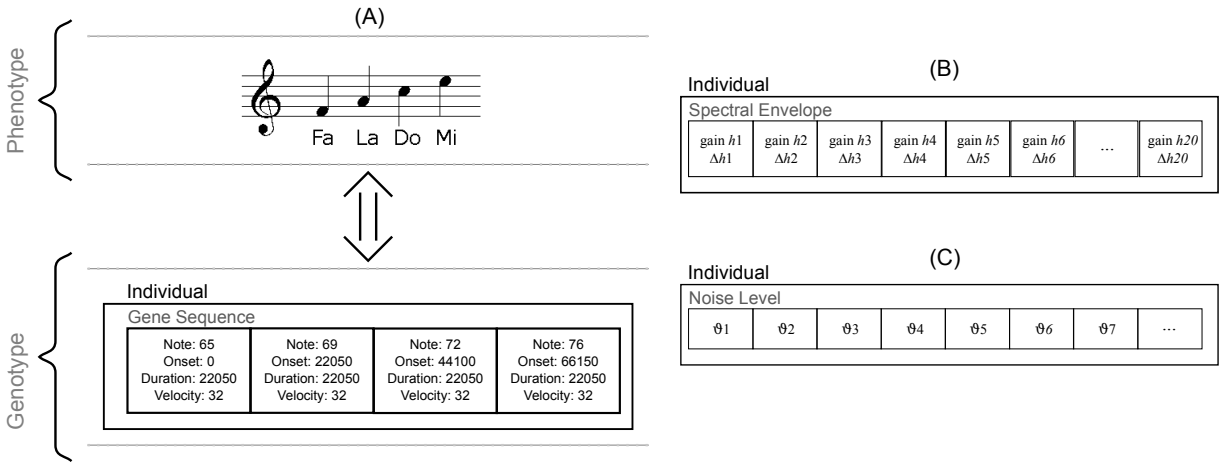

Figure 9.2: (A): Genotype of an individual and corresponding phenotype. (B): Spectral envelope encoding. Each gene corresponds to a pair of values: the gain of the respective harmonic (expressed in dB) and its inharmonicity deviation. (C): The noise is encoded as an adaptive threshold below a maximum peak of the current time frame. Each gene corresponds to the noise threshold (expressed in dB) of the corresponding frequency bin.

Moreover, the purpose of the Hill-Climber algorithm is to adjust the duration of notes and merge notes that overlap (transverse several audio segments). This process requires both duration and onset information on each gene.

In order to deal with dynamic noise level estimation and also spectral envelope modeling, additional chromosomes were included inside the genotype: one additional chromosome to encode the adaptive threshold used for the dynamic noise level estimation and one additional chromosome for each internal piano to adapt its spectral envelope to best match the piano played in the original audio.

Spectral Envelope Modeling

As in our previous approaches (see Chapter 7, Sections 7.2 and 7.3), the spectral envelope is encoded on the individual as a new chromosome (see Figure 9.2 (B)), where each gene corresponds to a pair of values: the gain for its harmonic - gain h_i - expressed in dB and its inharmonicity deviation - Δh_i - for each partial. Although the frequency of each partial could be calculated using the equation proposed by Fletcher et al. (1962), such as in the works of Emiya et al. (2008a, 2010): $f_h = h f_0 \sqrt{1 + \beta h^2}$ where β is the inharmonicity coefficient of the piano tone (Fletcher and Rossing, 1998), the encoding of the harmonic deviation of each partial pertaining to the genome of each individual genome was adopted so the system could be general enough to work with other kinds of pitched instruments.

Noise Level Estimation

During the transcription process, the algorithm compares the magnitude spectrum of each generated transcription with the original audio. This comparison should rely only on the spectral peaks of both sounds. Otherwise, spectral differences on spurious locations might lead the algorithm to an erroneous transcription. Thus, spectral data which does

not belong to the spectral peaks should be discarded. This requires a way to somehow ignore spectral differences of spurious components.

As in Yeh et al. (2010), the noise is understood as generated from white noise filtered by a frequency-dependent spectral envelope. This way, the noise level is defined as the expected magnitude level of noise peaks and encoded in an additional chromosome. This chromosome has the noise level for each frequency bin (see Figure 9.2 (C)). The noise level is encoded as an adaptive threshold below the maximum peak of the current time frame n, such that:

$$z[k] = \max(X(n,k)) + ind.noise[k] \tag{9.2}$$

where $ind.noise[k]$ corresponds to the noise value encoded in the kth gene (see Figure 9.2 (C)). Therefore, synthesized peaks below the $z[k]$ threshold are considered as noise peaks:

$$\widehat{X}(n,k) = \begin{cases} \min(z[k], X(n,k)) & \text{if } \widehat{X}(n,k) < z[k] \\ \widehat{X}(n,k), & \text{if } \widehat{X}(n,k) \geq z[k] \end{cases} \tag{9.3}$$

where $X(n,k)$ is the magnitude of the kth bin from the nth frame of the original spectrum, $\widehat{X}(n,k)$ represents the magnitude of the kth bin of the nth frame of the model spectrum (individual). This way, spectral data below the noise threshold will be considered as $\min(z[k], X(n,k))$. Thus, below the threshold, the spectrum of each generated transcription will be equal to the spectrum of the original audio: their difference below the noise threshold will always be zero (see top right plot of Figure 9.3). Moreover, the spectral peaks (or at least most of them) will be above the threshold and, thus, have impact on the comparison. If it happens, for some reason, to have a spectral peak below the threshold there are two hypotheses: (1) if the corresponding spectral peak on the original audio is also below the threshold: they will be equal (see top right plot of Figure 9.3, below 3500 Hz); (2) if the corresponding spectral peak on the original audio is above the threshold: the spectral peak of the candidate transcription will be equal to the threshold (see top right plot of Figure 9.3, below 3500 Hz) - this way, it will still have impact on the comparison, but since its difference between the corresponding spectral peak on the original audio is diminished, it is easier for the adaptive spectral envelope modeling to compensate their differences and to do the rest.

Figure 9.3 illustrates how both Spectral Envelope Modeling and Dynamic Noise Level estimation work, by showing an example of a transcription of a C major chord (C4 - 261.6256 Hz; E4 - 329.6276 Hz; and G4 - 391.9954 Hz) played by a Bosendorfer piano (original audio) and its generated transcription (Bechstein). The bottom left plot on this Figure shows the corresponding input chord played the Bosendorfer piano (the original audio) and its generated transcription (Bechstein). The spectrum of the generated transcription consists of the sum of each estimated component (top 3 plots on the left of the same figure). The top right plot shows how the algorithm sees the generated mixture played by the internal synthesizer (Bechstein) after applying the noise level estimation, and the bottom right plot shows how the algorithms sees the generated mixture after applying both noise model estimation and spectral envelope. If we compare the bottom

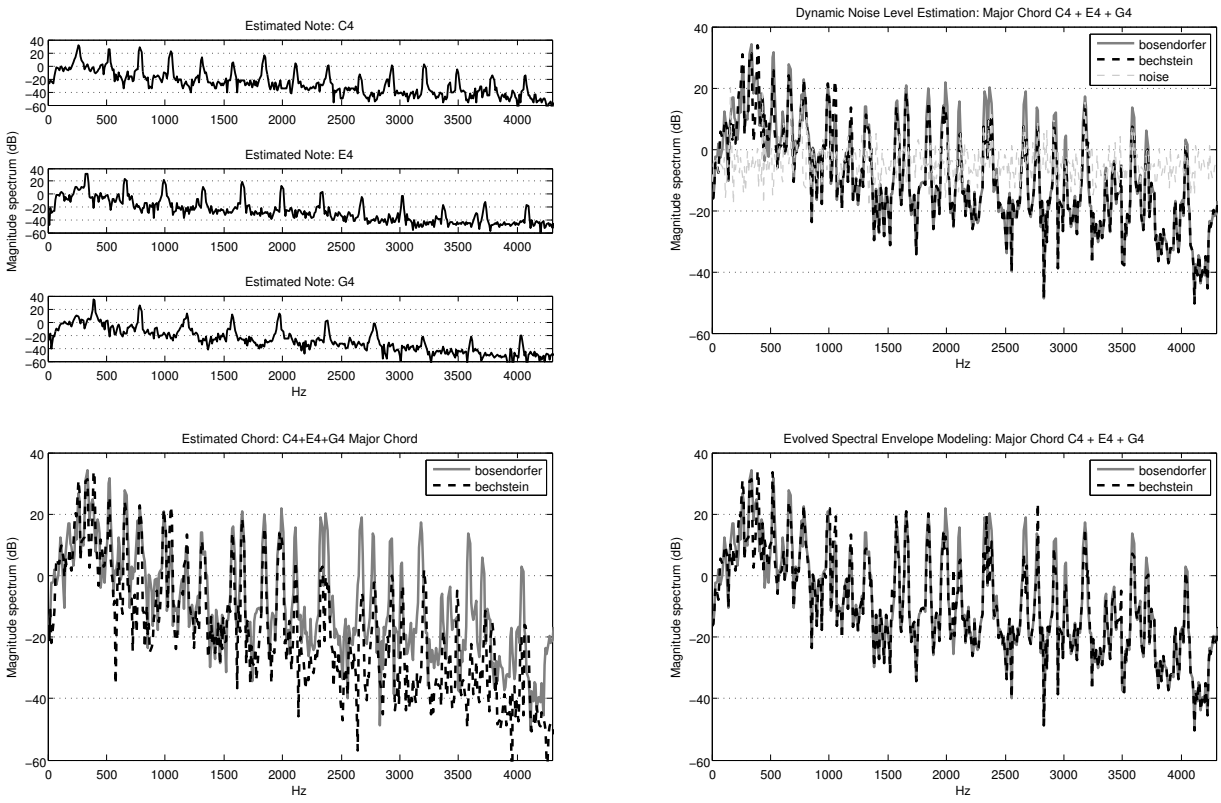

Figure 9.3: The bottom left plot represents a major chord (C4 - 261.6256 Hz; E4 - 329.6276 Hz; and G4 - 391.9954 Hz) played by a Bosendorfer piano (original audio) and its generated transcription (Bechstein). The spectrum of the generated transcription consists of the sum of each estimated component (top 3 plots on the left of the same figure). The top right plot represents the same spectrum after applying the noise level estimation (light gray) and, finally, the bottom right plot represents the latter spectrum with the evolved spectral envelope modeling.

left plot, which represents the original audio versus the generated mixture with the bottom right plot, where both dynamic noise level estimation and adaptive spectral envelope modeling were applied, we can see that the two initially different spectra became almost identical. This way, the algorithm will consider the generated transcription as correct despite their spectral differences (bottom left plot).

9.2.1 Fitness Evaluation

A good evaluation of each candidate's quality leads to a better selection of candidate solutions to form the next generation, speeding up the convergence of the algorithm towards a possible maximum. On the other hand, a less efficient quality (fitness) evaluation of each candidate (individual) can drastically reduce the evolution of the genetic algorithm. The fitness function is the key in the evolution/convergence of the genetic algorithms when solving different kinds of problems.

To evaluate candidate transcriptions, we need first to render them to an audio signal and then compare the corresponding audio signals to the input audio segment. Transcrip-

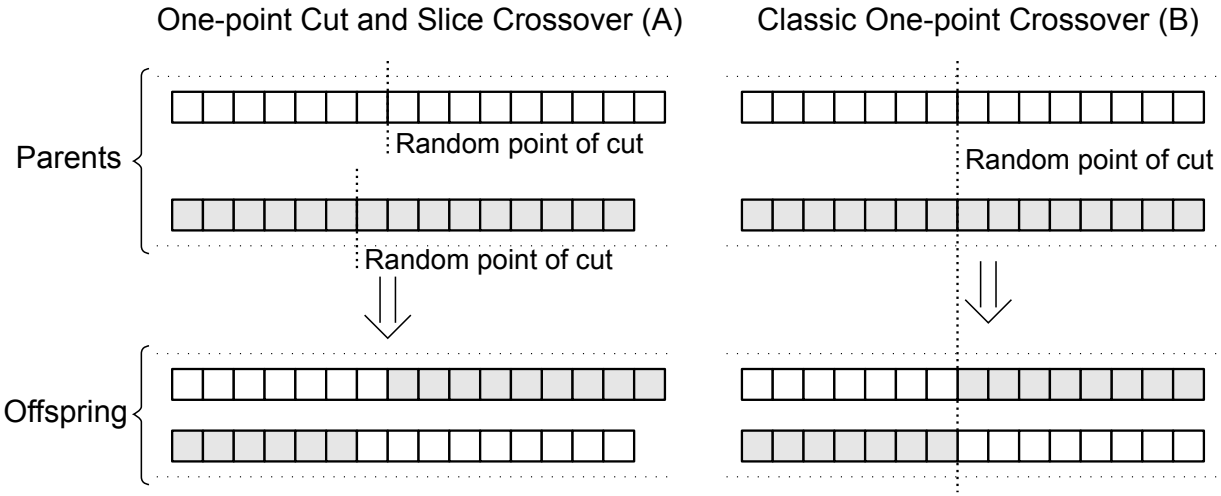

Figure 9.4: Recombination operators: one-point *cut and slice* crossover (A) and classic one-point crossover (B).

tions whose audio is similar to the audio input are closer to the desired solution and, thus, have fewer errors. The comparison between the candidate transcriptions and the input audio segment is done in the frequency domain.

For the rendering process, we considered a dynamic range of 16dB, that is: a note can vary its dynamics between 1 and 127. In particular, 127 MIDI velocity value corresponds to +8dB gain and 1 MIDI velocity corresponds to −8dB gain and 64 MIDI velocity corresponds to 0 dB gain. The gain, according to each note dynamic, is given by $10^{\frac{vel-64}{80}}$. After each note offset, the following release equation is applied: $release(t) = \frac{2000-\frac{t}{36}}{2000+t}$, where t varies from $t = 0 \ldots 72000$.

The current fitness function is based on the Log Spectral Distance or Log Spectral Distortion and was chosen empirically among several other spectral distances (Wei and Gibson, 2000). The fitness function is defined by the equation:

$$f(i) = \sum_{n=1}^{nMax} \sqrt{\sum_{k=2}^{\frac{N}{2}} \left(\left[10 \log_{10} \frac{|X(n,k)|}{\left|\widehat{X}(n,k)\right|} \right]^2 \times \log_2 \left(1 + \frac{1}{k} \right) \right)} \tag{9.4}$$

where N is the size of the Hamming window, which is 93 ms (i.e. $N = 4096$ with 44100 Hz sampling rate). k starts in 2 because it is the bin corresponding to the frequency of the first piano note (A_0 = 27,5 Hz). The multiplication by $\log_2(1 + \frac{1}{k})$ normalizes the weight of the bins of each octave so that, when summed, all the octaves have the same weighted sum equal to 1. As in Eq. 9.3, $X(n,k)$ is the magnitude of the kth bin from the nth frame of the original spectrum, $\widehat{X}(n,k)$ represents the magnitude of the kth bin of the nth frame of the model spectrum (candidate solution being evaluated).

9.2.2 Recombination

In this specific problem, individuals might differ in the number of genes (detected F0s) so the classic one-point crossover (Goldberg, 1989) had to be adopted to recombine individ-

uals with different number of genes by choosing different points-of-cut on each parent to generate the offspring. This one-point *cut-and-splice* recombination operator is described in Figure 9.4 (A).

The chromosomes encoding the spectral envelope of each instrument or the noise level estimation have always the same size: the number of genes is fixed. Thus, the classic one-point crossover operator (Goldberg, 1989) (see Figure 9.4 (B)) can be easily applied without any restriction.

9.2.3 Mutation

The mutation operator consists on the following steps described in Algorithm 9.1, where P is the mutation probability.

Algorithm 9.1: Mutation

1: **for** each gene **do**
2: $r \leftarrow$ random()
3: **if** $r \leq P_m$ **then**
4: Choose one of the following mutations:

- note change
- duration change by $[-50\text{ms}, 50\text{ms}]$
- harmonic change
- add a new note
- velocity change by $[-8, 8]$
- timbre change
- remove the current gene

5: **end if**
6: **end for**

During the *note change* and *add a new note* mutations, the note value is chosen from a list containing all allowed notes. This list is previously calculated according to the most prominent spectral peaks in each frame (see Appendix D). The *harmonic change* mutation changes the current note to a harmonic location: minus 12 semitones (half of the frequency of the note - one octave below); minus 19 semitones; minus 24 semitones (one fourth of the frequency of the note - two octaves below); minus 28 semitones; minus 31; minus 34 and, minus 36. This mutation has the purpose of solving harmonic errors that may occur in the detection of the possible notes, since the detection only selects notes from the most prominent spectral peaks. *Timbre change* mutation happens to change the instrument that plays a given note. This mutation exists to improve the support of other kind of pitched instruments.

Spectral Envelope Modeling

The chosen mutations for this chromosome are changing the gain of a harmonic by a random value in the range $[-12, 12]$ dB and changing the inharmonicity deviation using a bin value in the range $[-3, 3]$. The gain for F0 and its inharmonicity deviation are always 0 since they are not coded in the chromosome.

Noise Level Estimation

The mutation may occur in each gene and changes the power magnitude of the corresponding bin by a value in the range $[-3, 3]$ dB.

9.2.4 Initialization

According to our previous approach (see Chapter 7, Section 7.1.6), the initial population has two major contributions to the end result. First, if the initial population is created nearer (or even half way) to the final result than a randomly generated initial population, the genetic algorithm will need a much smaller number of generations to achieve the target result. Nevertheless, it is also important to have a very heterogeneous initial population to allow a better exploration of different areas of our search space.

The first step to get good results is to find a way to create an initial population somehow based on the original audio signal. Although we are aware that the initial population could have been based on cepstrum (Bogert et al., 1963), we choose to base it on the major peaks of the spectrogram. This happens to ensure that the genetic algorithm has enough biodiversity (genetic material) to perform the search process. The main power behind genetic algorithms relies on the capability of selecting the best parts of each candidate solution and recombining them into fitter and fitter solutions, something which requires a more heterogeneous population (Holland, 1992b). Thus, for the starting population, each individual is created with a random number of notes selected from the previously generated list of possible notes (see Appendix D). After its creation, each individual suffers 10 forced mutations.

Although the initialization stage depends on the spectral peak picking to determine the list of possible notes, the lack of frequency resolution does not hinder the accurate peak picking at the low frequencies, and thus does not affect the estimation of those notes. During the selection of possible notes stage, the α most prominent peaks are selected from the original spectrum and then, for each peak, the corresponding musical notes are added to the list of possible notes (see Appendix D). This process takes into account that each bin (specially on lower frequencies) might correspond to several musical notes and, if it is the case, all those notes are added to the possible notes list. This way, even with low frequency resolution, we assure that at least the correct note is added to the list of possible notes. Therefore, the algorithm is able to choose the correct note from the set of candidates. Moreover, since we are selecting the α most prominent peaks, there is a high probability of selecting harmonically related notes of the correct ones. If this is the case, the harmonic change mutation will fix those notes.

Tests were also performed using the cepstrum for this process, but the experiments have shown that despite having several individuals that are harmonically related, the genetic algorithm performs better when the initial population is based on the spectrogram.

9.2.5 Survivor Selection

During recombination, each pair of individuals generates two offsprings. This leads to an overcrowded population. The survivor selection operator chooses which individuals should or should not pass to the next generation. As in our previous approaches (see Chapter 7), the chosen selection method consists in determining the best individuals for survival. Also, 5% of the new generation consists on copies of the best individual of the previous generation, each with one forced mutation. This extends the robustness of the genetic algorithm, improving the global search by using local search on the vicinity of the best achieved solution (Hart et al., 2004).

9.3 Hill-Climber

Algorithm 9.2: Hill-Climber

```
1:  best ← individual returned by the genetic algorithm
2:  bestFitness ← best.evaluateFitness()
3:  i ← 1
4:  while i ≤ best.numberOfNotes do
5:     ind ← best
6:     change ind.notes[i] duration by +50ms
7:     if ind.notes[i] now overlaps with another note then
8:        merge both notes
9:     end if
10:    fit ← ind.valuateFitness()
11:    if fit ≥ bestFitness then
12:       best ← ind
13:       bestFitness ← fit
14:    else
15:       i ← i + 1
16:    end if
17: end while
18: return MIDI file
```

The Hill-Climber algorithm (see Algorithm 9.2) is fed with the final transcription generated by the algorithm and consists in applying the following steps: traversing all musical notes and, for each note, increasing its duration by 50ms. If this note now overlaps with another note both notes are merged. Also, if the quality of the individual improves, this process is then repeated on the same musical note, otherwise the last change is discarded and the algorithm goes for the next musical note.

Table 9.1: Genetic Algorithm parameters

Genetic Algorithm parameters	
population size	200
maximum number of generations	50
probability of recombination	0.8
parent Selection	tournament (tournament size = 2)
probability of mutation	0.2 (transcription) 0.01 (spectral envelope) 0.02 (noise)

The output of the system is the result achieved by the hill-climber. The impact of the Hill-Climber on the overall results is studied in Section 9.4.4.

9.4 Experiments and Results

9.4.1 Implementation and Tuning

The proposed approach was implemented using the C programming language. The transcription of each audio segment was run in parallel (one thread per segment). The computational time of the approach is 60 × real time. Several tests were carried out across a selection of audio files, including a development database of 2700 mixtures from 7 different pianos, with polyphony levels from 2 to 7 (more details about this database are provided in Section9.4.2). Those same tests employed various frame lengths, window functions and hop sizes. A 93-ms frame length with Hamming window function and 75% hop size were chosen empirically. $\alpha = 10$ is used for generating the possible notes list (see Appendix D) and the number of harmonics for the spectral envelope modeling was also empirically set to 20. The algorithm was also set to a 5 limit polyphony since polyphony levels in musical recordings have a 4.5 average polyphony and a 3.1 standard deviation reported for a number of classical music pieces (Emiya, 2008, p.114). The probabilities and other parameters specific to the Genetic Algorithm are shown in Table 9.1.

Although the resolution of 10.76 Hz might not appear enough for discriminating notes starting at MIDI note A0 (25.500 Hz), it suits their spectrum comparison. Recall that the algorithm searches for the best combination of notes by performing an evaluation based on their spectrum comparison with the original audio. Thus, same musical notes have similar harmonic locations, even if played by different pianos. Therefore, and that is also the case for lower notes, the algorithm tends to choose the correct piano samples because they correlate best with those played in the original audio. In fact, the algorithm is capable of identifying and discriminating notes from the 21 MIDI note (A0 - 27.500 Hz) to the 108 MIDI note (C8 - 4186.0 Hz). Also, the comparison between the original audio

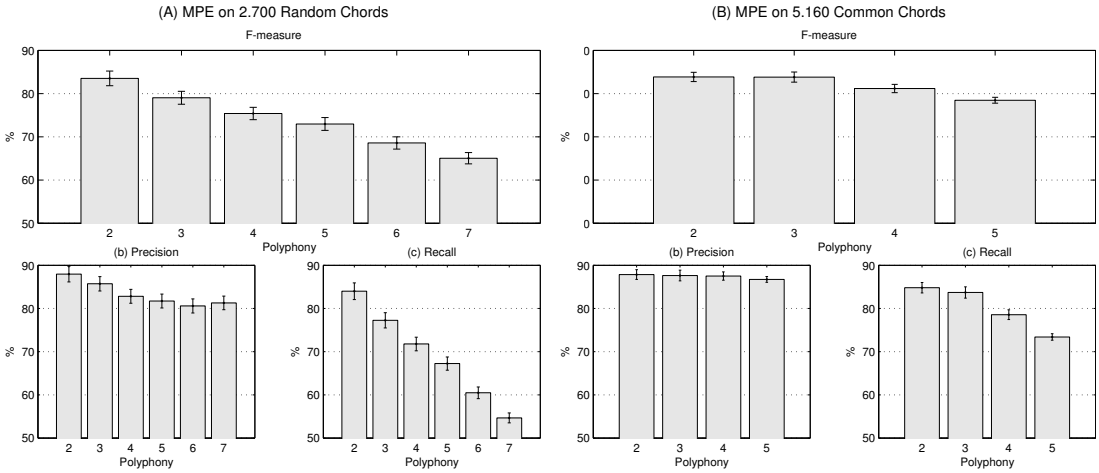

Figure 9.5: (A) Multi-pitch estimation results for each polyphony on 2.700 random-pitched chords; (B) Multi-pitch estimation results for each polyphony on 5.160 common chords.

and the individual's audio is made on the frequency domain, frame by frame. A frame overlap ratio of 25% means less STFT frames and, therefore, fewer comparisons, which in turn means having fewer mathematical calculations that consequently computationally accelerate the algorithm. Moreover, less STFT frame comparisons mean fewer errors in the signal comparison that might occur due to spurious components or harmonic cancellation, which ultimately leads to a faster conversion of the Genetic Algorithm, hence better results. Even though it might occur an amplitude modulation during overlap-add, it will happen on both signals being compared therefore, so it will not affect the comparison.

9.4.2 Evaluation

The proposed algorithm has been tested on a database called MAPS (Emiya, 2008; Emiya et al., 2010) consisting of around 10,000 piano sounds either recorded by using an upright Disklavier piano or generated by several virtual piano software products based on sample sounds. The development set and the test set are disjoint. A set of 2,700 random chords between A0 (25.500 Hz) and C8 (4186Hz) with polyphony levels ranging from 2 to 7 were used in the former while the latter comprises 2,700 random chords and 5,160 common chords from western music (major, minor, etc.) from C2 (65.406 Hz) to B6 (1975.5 Hz). For each sound, a single 93ms-frame located after the last onset time is extracted and analyzed. In total, 10,560 audio files were used from two upright pianos and five grand pianos. The authors considered only the frame after the last onset for both training and test tasks because both tasks consisted in transcribing simple chords and also due to time restrictions since there were used a total of 10,560 different chords. During all other tests, the algorithm performs a frame by frame analysis.

General results are presented in Figure 9.5. Relevant items are defined correct notes after rounding each F0 to the nearest half-tone. Typical metrics are used: the recall is the ratio between the number of relevant items and of original items; the precision is the ratio between the number of relevant items and detected items; and the F-measure is the harmonic mean (Nostrand, 1962) between the precision and the recall. In this context,

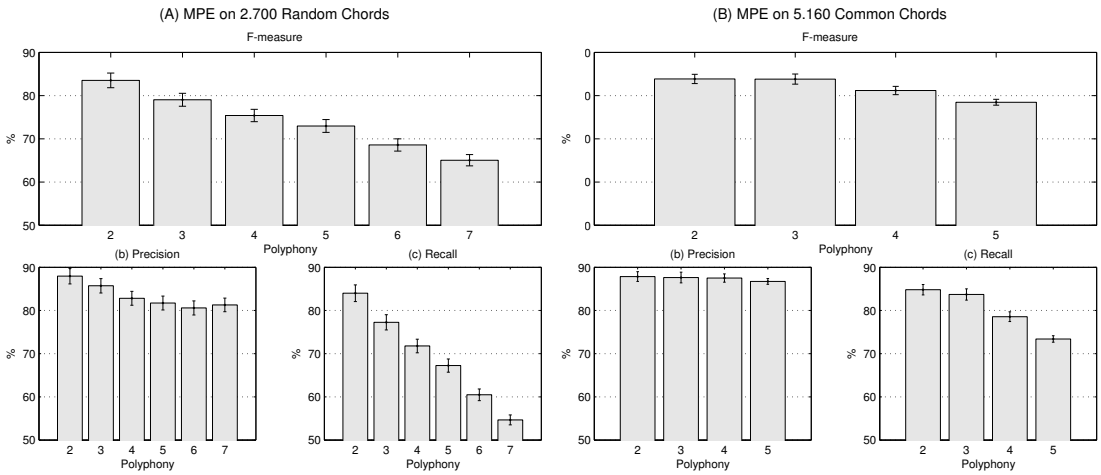

Figure 9.6: The distribution of estimated polyphony for the polyphony from 2 to 7 on (A) random-pitched chords and from 2 to 5 on (B) common chords. The title of each subfigure indicates the correct polyphony; the x-axis represents the estimated polyphony; the y-axis represents the percentage of the estimated polyphony among all instances. The peaking at the correct polyphony is observed for polyphony below six, except for four.

our system returns F-measures of 84%, 79%, 74%, 73%, 69% and 65% for polyphony 2, 3, 4, 5, 6 and 7 on random chords and 84%, 84%, 81% and 78% for polyphony 2, 3, 4 and 5 on common chords. Moreover, the precision is high for all polyphony levels whereas the recall is decreasing whenever polyphony increases.

The ability of the system to infer the polyphony levels (independently of the pitches) is presented on Figure 9.6. For polyphony levels from 2 to 5 the system only fails on detecting the correct polyphony level on polyphony 4 on the random chords data set. Polyphony levels of 6 and 7 obviously fail because the system is set to a maximum 5 polyphony. On the other hand, the system successfully detects the correct polyphony for all polyphonies in the common chords data set. Also, F-measure values are between 6% and 10% better for common chords than for random-pitched chords. This suggests that, while the algorithm faces more harmonically related notes in common chords (spectral overlap), widely-spread F0s in random chords are a bigger difficulty. By limiting the algorithm polyphony to 5, the proposed system underestimates the polyphony level since the parameter tuning consists in optimizing the F-measure on the development set. This objective function could have been changed to take the polyphony level balance into account. This would result in reducing the polyphony underestimation trend. However, the overall F-measure would decrease. Moreover, from a different perspective, it has been shown that a missing note is generally less annoying than an added note when listening to a re-synthesized transcription (Cemgil et al., 2006b). Thus, underestimating the polyphony may be preferable to overestimating it. Still, this trend turns out to be the main shortcoming of the proposed method, and should be fixed in the future so it can efficiently address sounds with polyphony higher than 5 notes.

While the test database has 7 different pianos, the internal synthesizer of the algorithm consists only of three pianos, which means that the spectral envelope modeling plays a major role in the achieved results by adapting the internal piano samples to the 7 pianos

on the database. Moreover, the results are comparable from one piano to another, with only a small % deviation. This means that the results do not significantly depend on the upright/grand piano differences.

9.4.3 Comparison with other State-of-the-art algorithms

For a deeper analysis, we decided to extend a previous study performed by Emiya et al. (2008a). This study compares several state-of-the-art music transcription algorithms: Emiya et al. (2008a); Vincent et al. (2008); Bertin et al. (2007); Marolt (2004a). We included four new algorithms: Reis et al. (2009); Vincent et al. (2010); Ryynänen and Klapuri (2005)[1] and the proposed method. In total, our algorithm is compared to seven other state-of-the-art transcription algorithms:

- Vincent'10 (Vincent et al., 2010);
- Reis'09 (Chapter 7, Section 7.3);
- Emiya EUSIPCO'08 (Emiya et al., 2008a);
- Vincent B'07 - The baseline method presented in (Vincent et al., 2008);
- Vincent H'07 - The harmonic method presented in (Vincent et al., 2008);
- Bertin'07 (Bertin et al., 2007);
- Marolt'04 (Marolt, 2004a);
- Ryynänen'05 (Ryynänen and Klapuri, 2005) - The authors thought that the inclusion of this algorithm was very important for the study performed since this algorithm won the last MIREX (Downie, 2008a; Downie et al., 2010b) competitions.

Our previous method (see Chapter 7, Section 7.3) was also included for comparison.

These algorithms were run on 9 randomly chosen pieces of music used on Emiya benchmark (Emiya et al., 2008a)[2]. All the results that will be presented on this section are available at the author's website[3]. Along with these results we also provide the audio files and their MIDI representation in visual form.

Figure 9.7 (A) shows the results obtained by our algorithm in comparison to the other seven state-of-the-art algorithms and (B) shows the Friedman Mean Ranks with regard to F-measure on individual files. Results in Figure 9.7 (A) are presented using the Onset-only metric. In this metric a correct note implies a correct onset with a deviation up to 50 ms. Results are presented on the Onset-only metric as Recall, Precision, F-measure and Mean Overlap Ratio (MOR) (Ryynänen and Klapuri, 2005). Mean Overlap Ratio is an averaged ratio between the length of the intersection of the temporal supports of an

[1]The authors asked to several researchers in the field for their algorithms so that they could also be included in this study. These were the algorithms provided.

[2]see: `http://www.irisa.fr/metiss/vemiya/EUSIPCO08/bench0.html`

[3]`http://www.estg.ipleiria.pt/~gustavo.reis/benchmark`

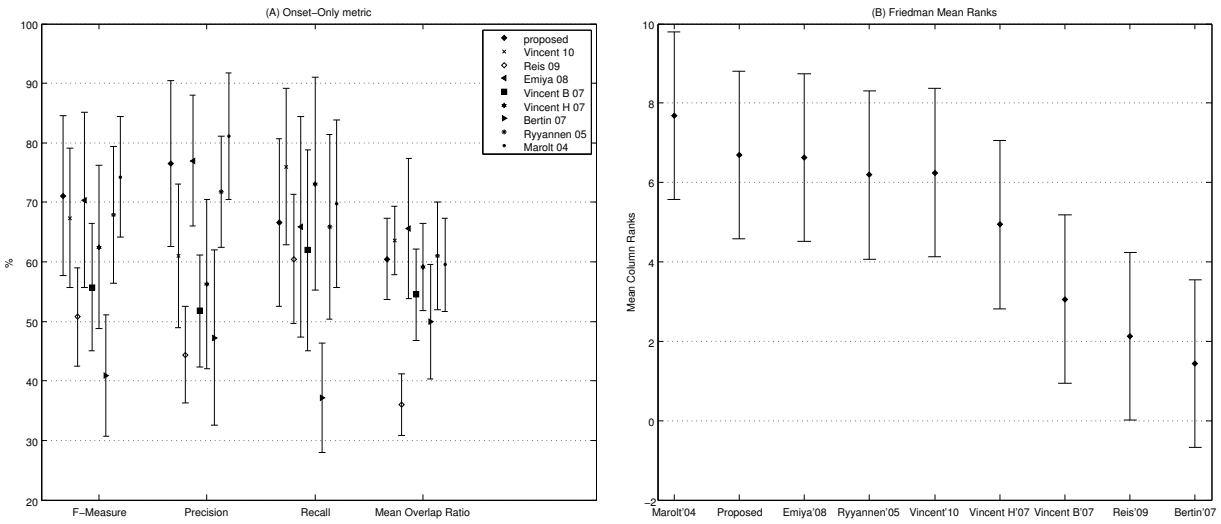

Figure 9.7: (A) Onset-only F-measure, Precision, Recall and Mean Overlap Ratio, respectively; (B) Friedman Mean Ranks with regard to F-measure on individual files.

original note and its transcription, and of the length of their union. This measure acts more like a guideline for researchers to know how the correctly transcribed notes intersect with the original notes in terms of note duration. It is also used to measure the phrasing similarity with the original piece.

Performance rates on Figure 9.7 (A) have been averaged with respect to all tested musical pieces, and the respective standard deviation is also represented. While we have achieved low F-measure deviation on the evaluation of audio chords (multi pitch estimation), on the automatic transcription benchmark we have a deviation around 15%, no matter the method. This happens because in the chord evaluation we used isolated frames composed of one chord, while here the evaluation implies several other difficulties, like having asynchronous notes overlapping in time, detecting onsets, estimating the end of damping notes, dealing with reverberation queues and so on. Thus, large standard deviation values are due to the dependency of musical excerpts. For instance: F-measure greater than 85% is reached on musical pieces with slow tempo or low polyphony, while fast pieces are generally difficult to transcribe. Since the results dramatically depend on the database, this was the main reason why we decided to extend a previous study. This way we can have a more realistic comparison among all tested algorithms.

In this context (Figure 9.7 (A)) our system is comparable to the state-of-the-art: it ranked the 2nd place, below Marolt'04 and above Emiya'08. It should be noted that Emiya'08 is the algorithm with higher Mean Overlap Ratio, which suggests that the used HMM framework for note tracking is efficient in both selecting pitches among candidates, and also in detecting their possible endings.

Regarding the proposed approach and Reis'09, both algorithms have large differences in F-Measure (20%), Precision (32%) and Mean Overlap Ratio (25%) and a small difference on Recall (6%). The large F-Measure difference (20%) shows that the proposed system has much better performance than the previous genetic algorithm approaches. It also features a smaller percentage of both false positive rate and false negative rate (Preci-

Table 9.2: Tukey-Kramer HSD (Honestly Significant Difference) Multi-Comparison Onset-only metric

Algorithm	Algorithm	Lower Bound	Mean	Upper Bound	Significance
Marolt'04	Proposed	-3.2317	1.0000	5.2317	FALSE
Marolt'04	Emiya'08	-3.1692	1.0625	5.2942	FALSE
Marolt'04	Ryyannen'05	-2.7317	1.5000	5.7317	FALSE
Marolt'04	Vincent'10	-2.7942	1.4375	5.6692	FALSE
Marolt'04	Vincent H'07	-1.4817	2.7500	6.9817	FALSE
Marolt'04	Vincent B'07	0.3933	4.6250	8.8567	TRUE
Marolt'04	Reis'09	1.3308	5.5625	9.7942	TRUE
Marolt'04	Bertin'07	2.0183	6.2500	10.4817	TRUE
Proposed	Emiya'08	-4.1692	0.0625	4.2942	FALSE
Proposed	Ryyannen'05	-3.7317	0.5000	4.7317	FALSE
Proposed	Vincent'10	-3.7942	0.4375	4.6692	FALSE
Proposed	Vincent H'07	-2.4817	1.7500	5.9817	FALSE
Proposed	Vincent B'07	-0.6067	3.6250	7.8567	FALSE
Proposed	Reis'09	0.3308	4.5625	8.7942	TRUE
Proposed	Bertin'07	1.0183	5.2500	9.4817	TRUE
Emiya'08	Ryyannen'05	-3.7942	0.4375	4.6692	FALSE
Emiya'08	Vincent'10	-3.8567	0.3750	4.6067	FALSE
Emiya'08	Vincent H'07	-2.5442	1.6875	5.9192	FALSE
Emiya'08	Vincent B'07	-0.6692	3.5625	7.7942	FALSE
Emiya'08	Reis'09	0.2683	4.5000	8.7317	TRUE
Emiya'08	Bertin'07	0.9558	5.1875	9.4192	TRUE
Ryyannen'05	Vincent'10	-4.2942	-0.0625	4.1692	FALSE
Ryyannen'05	Vincent H'07	-2.9817	1.2500	5.4817	FALSE
Ryyannen'05	Vincent B'07	-1.1067	3.1250	7.3567	FALSE
Ryyannen'05	Reis'09	-0.1692	4.0625	8.2942	FALSE
Ryyannen'05	Bertin'07	0.5183	4.7500	8.9817	TRUE
Vincent'10	Vincent H'07	-2.9192	1.3125	5.5442	FALSE
Vincent'10	Vincent B'07	-1.0442	3.1875	7.4192	FALSE
Vincent'10	Reis'09	-0.1067	4.1250	8.3567	FALSE
Vincent'10	Bertin'07	0.5808	4.8125	9.0442	TRUE
Vincent H'07	Vincent B'07	-2.3567	1.8750	6.1067	FALSE
Vincent H'07	Reis'09	-1.4192	2.8125	7.0442	FALSE
Vincent H'07	Bertin'07	-0.7317	3.5000	7.7317	FALSE
Vincent B'07	Reis'09	-3.2942	0.9375	5.1692	FALSE
Vincent B'07	Bertin'07	-2.6067	1.6250	5.8567	FALSE
Reis'09	Bertin'07	-3.5442	0.6875	4.9192	FALSE

sion and Recall). Note that there is also a considerable difference on Mean Overlap Ratio (25%). This means that our system results in a more efficient transcription, enhancing the phrasing similarity with the original pieces and thus improving the subjective quality when hearing the correctly transcribed notes. The computation time of Reis'09 algorithm is 540 times real-time which is much higher than the proposed method: 60 times real-time. This also represents a significant improvement. Also, the most significant difference between both algorithms relies on Precision (32%). This shows that the adaptive spectral envelope modeling, along with the dynamic noise level estimation, play a major role in reducing the false positive rate: precision is the percentage of the transcribed notes that are correctly transcribed. Thus, the proposed algorithm is much more effective on reducing the harmonic overfitting. On the other hand, a lower difference on Recall tells us that both systems have a similar false negative rate, which again emphasizes that the main difference on both systems is on how the dynamic noise level estimation, along with the adaptive spectral envelope modeling, have a significant impact on reducing the false

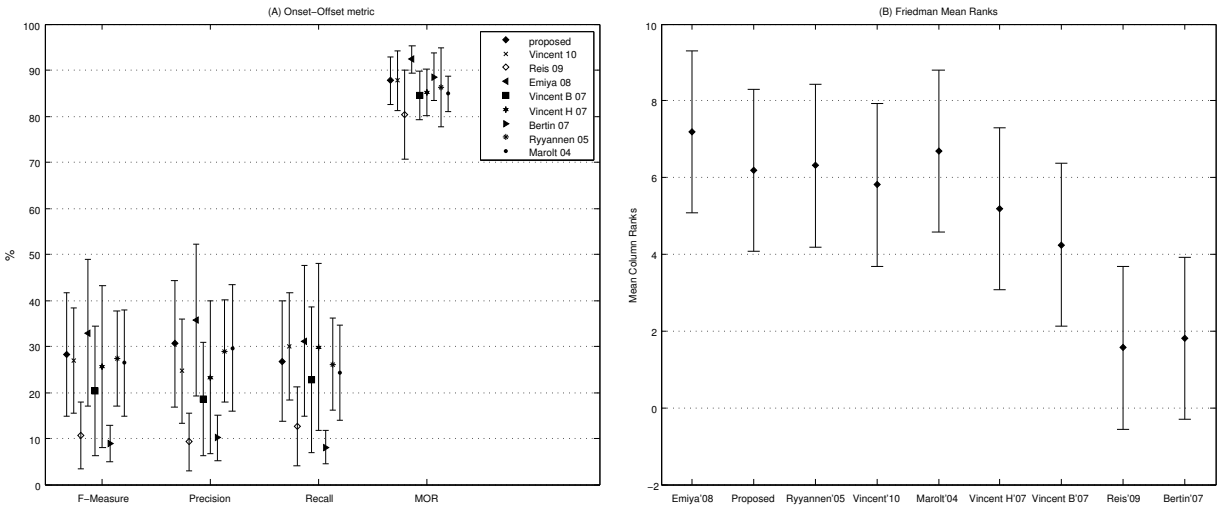

Figure 9.8: (A) Onset-Offset F-measure, Precision, Recall and Mean Overlap Ratio, respectively; (B) Friedman Mean Ranks with regard to F-measure on individual files.

positive rate: this system is much more effective in dealing with the harmonic overfitting.

Figure 9.7 (B) shows that the proposed approach achieves the second best mean rank, which means that, on average, our system ranked second place in each individual file. Table 9.2 shows the Tukey-Kramer Honestly Significant Difference (HSD) multi-comparison of the Friedman Mean Ranks calculated on Figure 9.7 (B). This table shows that the difference between the proposed system and the algorithm that ranked first is not statistically significant. Moreover, on this metric, our proposal is significantly better than Reis'09 (best algorithm of all previous genetic algorithm approaches) and Bertin'07.

Figure 9.8 (A) shows the same benchmark using the Onset-Offset metric. This metric also presents the results as Recall, Precision, F-measure and Mean Overlap Ratio (MOR). In the Onset-Offset metric a correct note implies a correct onset with a deviation up to 50ms and a correct offset with a deviation of up to 20% of the note length or 50ms. Performance rates have also been averaged with respect to all tested musical pieces, and the respective standard deviation is also represented.

According to this metric, our system is also comparable to the state-of-the-art: it ranked the 2nd place, below Emiya'08 and above Vincent'10. It should be noted that, in this context, Emiya'08 is the algorithm with higher F-measure, Precision, Recall and Mean Overlap Ratio. This happens because it is the most effective algorithm estimating the offset time, which reinforces what was mentioned before: the HMM framework used by Emiya'08 for note tracking is efficient in both selecting pitches among candidates, and also in detecting their possible endings. However, Emiya'08 algorithm has a computing time of 200 times real-time while our system's computing time is 60 times real-time. Figure 9.8 (B) shows that the proposed approach achieves the fourth best mean rank, which means that, despite having the second best overall mean on F-Measure, on average, our system ranked fourth place in each individual file. Table 9.3 shows the Tukey-Kramer HSD multi-comparison of the Friedman Mean Ranks calculated on Figure 9.8 (B). This table shows that the difference between the proposed system and the algorithm that ranked first is

Table 9.3: Tukey-Kramer HSD (Honestly Significant Difference) Multi-Comparison Onset-Offset metric

Algorithm	Algorithm	Lower Bound	Mean	Upper Bound	Significance
Emiya'08	Proposed	-3.2339	1.0000	5.2339	FALSE
Emiya'08	Ryyannen'05	-3.3589	0.8750	5.1089	FALSE
Emiya'08	Vincent'10	-2.8589	1.3750	5.6089	FALSE
Emiya'08	Marolt'04	-3.7339	0.5000	4.7339	FALSE
Emiya'08	Vincent H'07	-2.2339	2.0000	6.2339	FALSE
Emiya'08	Vincent B'07	-1.2964	2.9375	7.1714	FALSE
Emiya'08	Reis'09	1.3911	5.6250	9.8589	TRUE
Emiya'08	Bertin'07	1.1411	5.3750	9.6089	TRUE
Proposed	Ryyannen'05	-4.3589	-0.1250	4.1089	FALSE
Proposed	Vincent'10	-3.8589	0.3750	4.6089	FALSE
Proposed	Marolt'04	-4.7339	-0.5000	3.7339	FALSE
Proposed	Vincent H'07	-3.2339	1.0000	5.2339	FALSE
Proposed	Vincent B'07	-2.2964	1.9375	6.1714	FALSE
Proposed	Reis'09	0.3911	4.6250	8.8589	TRUE
Proposed	Bertin'07	0.1411	4.3750	8.6089	TRUE
Ryyannen'05	Vincent'10	-3.7339	0.5000	4.7339	FALSE
Ryyannen'05	Marolt'04	-4.6089	-0.3750	3.8589	FALSE
Ryyannen'05	Vincent H'07	-3.1089	1.1250	5.3589	FALSE
Ryyannen'05	Vincent B'07	-2.1714	2.0625	6.2964	FALSE
Ryyannen'05	Reis'09	0.5161	4.7500	8.9839	TRUE
Ryyannen'05	Bertin'07	0.2661	4.5000	8.7339	TRUE
Vincent'10	Marolt'04	-5.1089	-0.8750	3.3589	FALSE
Vincent'10	Vincent H'07	-3.6089	0.6250	4.8589	FALSE
Vincent'10	Vincent B'07	-2.6714	1.5625	5.7964	FALSE
Vincent'10	Reis'09	0.0161	4.2500	8.4839	TRUE
Vincent'10	Bertin'07	-0.2339	4.0000	8.2339	FALSE
Marolt'04	Vincent H'07	-2.7339	1.5000	5.7339	FALSE
Marolt'04	Vincent B'07	-1.7964	2.4375	6.6714	FALSE
Marolt'04	Reis'09	0.8911	5.1250	9.3589	TRUE
Marolt'04	Bertin'07	0.6411	4.8750	9.1089	TRUE
Vincent H'07	Vincent B'07	-3.2964	0.9375	5.1714	FALSE
Vincent H'07	Reis'09	-0.6089	3.6250	7.8589	FALSE
Vincent H'07	Bertin'07	-0.8589	3.3750	7.6089	FALSE
Vincent B'07	Reis'09	-1.5464	2.6875	6.9214	FALSE
Vincent B'07	Bertin'07	-1.7964	2.4375	6.6714	FALSE
Reis'09	Bertin'07	-4.4839	-0.2500	3.9839	FALSE

not statistically significant. Moreover, on this metric, our proposal is significantly better than Reis'09 and Bertin'07.

Finally, one last metric was employed for algorithm comparison: the Hybrid Decay/-Sustain Score (Nuno Fonseca, 2010). This metric was employed because it is the one that best correlates with the human hearing perception (Nuno Fonseca, 2010). Figure 9.9 (A) shows the results obtained using this metric. Results are presented as Decay Score, Sustain Score and Final Score: Decay Score is used for percussive pitched instruments and employs a note oriented approach considering only pitches and onsets, generating a score ([0-100]%) for each note; Sustain Score is used for sustain musical instruments (eg.: woodwind) and employs a time oriented approach measuring the overlap between the original and transcribed notes; the Final Score is the average value between Sustain Score and Decay Score.

According to this metric, the proposed system ranks the 1st place, above Vincent'10. This means that our system results in an efficient transcription, enhancing the phrasing

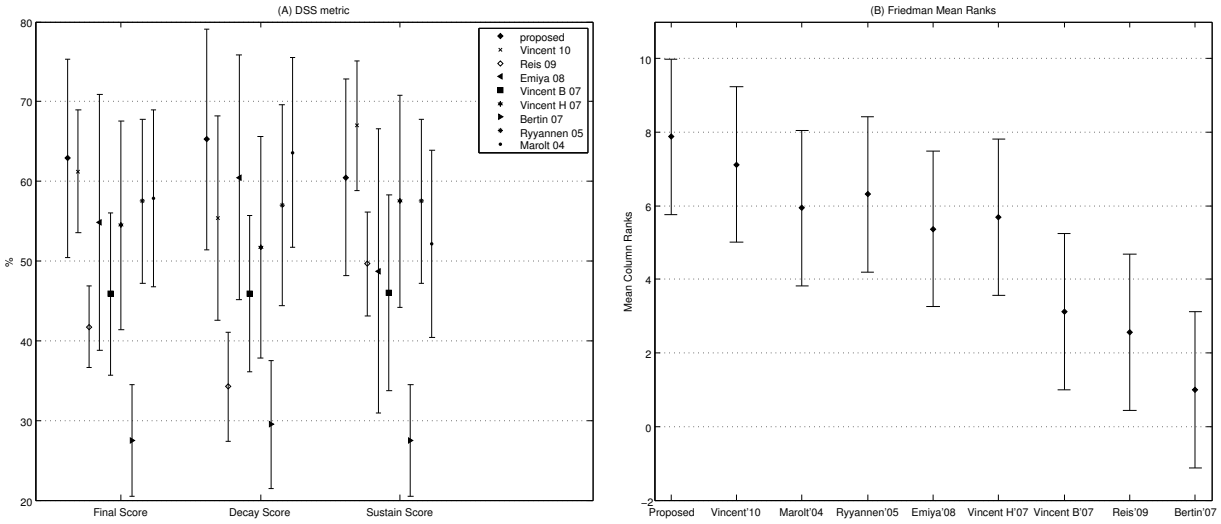

Figure 9.9: Final Score, Decay Score and Sustain Score, respectively.

similarity with the original piece and thus improving the subjective quality when hearing the correctly transcribed notes. Note that, since we are dealing with piano transcriptions, we can consider the value Decay Score instead of the Final score. In this case all algorithms rank the same places (not the same results), except Vincent'10. This happens because the latter algorithm has a relatively high Sustain Score.

Figure 9.9 (B) shows that the proposed approach achieves the best mean rank, which means that, on average, our system ranked first place in each individual file. Moreover, Figure 9.9 (B) along with Table 9.4 shows that our proposal is significantly better than Vincent B'07, Reis'09 and Bertin'07.

We believe that our approach has perceptually better results because, among all the other state-of-the-art algorithms, our system is the only one that tries to mimic the way how professional musicians learn to play a new song by ear: the algorithm "listens" to an audio file and then, during the transcription process, is always comparing the available candidates, to figure out which one is closer to the original song. In the end, it returns the transcription that most resembled the original audio.

9.4.4 Contribution of each module to the overall results

Figure 9.10 shows the contribution of each module to the global results of the proposed system: naïve, adaptive spectral envelope modeling (ASEM), dynamic noise level estimation (DNLE), both spectral envelope modeling and dynamic noise level estimation (ASEM+DNLE) and Hill-climber. Naïve corresponds to the genetic algorithm without the spectral envelope modeling and noise level estimation; ASEM is the naïve version with the spectral envelope modeling; DNLE is the naïve version with the dynamic noise level estimation; ASEM+DNLE corresponds to naïve version with both spectral envelope modeling and dynamic noise level estimation; and, finally, Hill-climber is the ASEM+DNLE with the Hill-climber applied i.e., the whole system. All these versions of the algorithm were run with same parameters, included in Table 9.1.

Table 9.4: Tukey-Kramer HSD (Honestly Significant Difference) Multi-Comparison Hybrid Decay/Sustain metric

Algorithm	Algorithm	Lower Bound	Mean	Upper Bound	Significance
Proposed	Vincent'10	-3.4817	0.7500	4.9817	FALSE
Proposed	Marolt'04	-2.2942	1.9375	6.1692	FALSE
Proposed	Ryyannen'05	-2.6692	1.5625	5.7942	FALSE
Proposed	Emiya'08	-1.7317	2.5000	6.7317	FALSE
Proposed	Vincent H'07	-2.0442	2.1875	6.4192	FALSE
Proposed	Vincent B'07	0.5183	4.7500	8.9817	TRUE
Proposed	Reis'09	1.0808	5.3125	9.5442	TRUE
Proposed	Bertin'07	2.6433	6.8750	11.1067	TRUE
Vincent'10	Marolt'04	-3.0442	1.1875	5.4192	FALSE
Vincent'10	Ryyannen'05	-3.4192	0.8125	5.0442	FALSE
Vincent'10	Emiya'08	-2.4817	1.7500	5.9817	FALSE
Vincent'10	Vincent H'07	-2.7942	1.4375	5.6692	FALSE
Vincent'10	Vincent B'07	-0.2317	4.0000	8.2317	FALSE
Vincent'10	Reis'09	0.3308	4.5625	8.7942	TRUE
Vincent'10	Bertin'07	1.8933	6.1250	10.3567	TRUE
Marolt'04	Ryyannen'05	-4.6067	-0.3750	3.8567	FALSE
Marolt'04	Emiya'08	-3.6692	0.5625	4.7942	FALSE
Marolt'04	Vincent H'07	-3.9817	0.2500	4.4817	FALSE
Marolt'04	Vincent B'07	-1.4192	2.8125	7.0442	FALSE
Marolt'04	Reis'09	-0.8567	3.3750	7.6067	FALSE
Marolt'04	Bertin'07	0.7058	4.9375	9.1692	TRUE
Ryyannen'05	Emiya'08	-3.2942	0.9375	5.1692	FALSE
Ryyannen'05	Vincent H'07	-3.6067	0.6250	4.8567	FALSE
Ryyannen'05	Vincent B'07	-1.0442	3.1875	7.4192	FALSE
Ryyannen'05	Reis'09	-0.4817	3.7500	7.9817	FALSE
Ryyannen'05	Bertin'07	1.0808	5.3125	9.5442	TRUE
Emiya'08	Vincent H'07	-4.5442	-0.3125	3.9192	FALSE
Emiya'08	Vincent B'07	-1.9817	2.2500	6.4817	FALSE
Emiya'08	Reis'09	-1.4192	2.8125	7.0442	FALSE
Emiya'08	Bertin'07	0.1433	4.3750	8.6067	TRUE
Vincent H'07	Vincent B'07	-1.6692	2.5625	6.7942	FALSE
Vincent H'07	Reis'09	-1.1067	3.1250	7.3567	FALSE
Vincent H'07	Bertin'07	0.4558	4.6875	8.9192	TRUE
Vincent B'07	Reis'09	-3.6692	0.5625	4.7942	FALSE
Vincent B'07	Bertin'07	-2.1067	2.1250	6.3567	FALSE
Reis'09	Bertin'07	-2.6692	1.5625	5.7942	FALSE

By analyzing Figure 9.10 we can tell that the naïve (genetic algorithm without hill-climber, spectral envelope modeling and dynamic noise level estimation) has a performance of: 18.25% F-measure, 18.25% Precision, and 18.5% Recall. By enabling the adaptive spectral envelope modeling (ASEM), Precision, F-Measure and Recall have an improvement around 1.75%: ASEM by itself does not bring significant improvements on the quality of the results since differences on spurious components will bias the comparison (candidate evaluation), and thus, the convergence of the genetic algorithm. On the other hand, if we only enable the dynamic noise level estimation (DNLE), F-Measure decreases 2.5%, Precision decreases 1.3% and Recall decreases 3.0%. This happens because the noise estimation discards all the spurious information when evaluating the transcriptions: only the spectral peaks above the noise threshold are considered. This leads the algorithm on adding several notes in harmonic locations to compensate the timbre differences, decreasing both precision and recall.

Adaptive spectral envelope modeling along with dynamic noise level estimation (ASEM

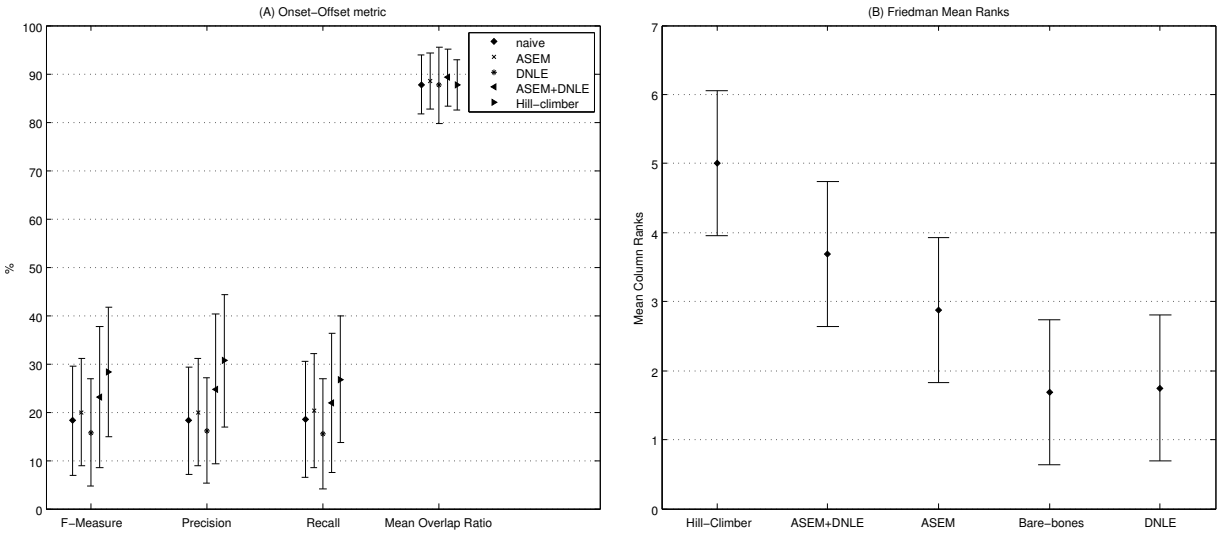

Figure 9.10: (A) Contribution of each module to the global results of the proposed system - Onset-offset; (B) Friedman Mean Ranks with regard to F-measure on individual files.

+ DNLE), improve the system's performance: F-measure increases 4.88% and Precision has a boost of 6.5%. This tells that adaptive spectral envelope modeling together with dynamic noise level estimation have a great impact on reducing the harmonic overfitting: the percentage of correctly transcribed notes increases around 6.5%, which means that the system significantly reduced the false positive rate. Recall improves 3.38%. The algorithm performs well because both ASEM and DNLE were designed to work together so that they can compensate each-other.

Hill-Climber gives the major improvement to the proposed system. It raises the performance of ASEM + DNLE by: 5.13% F-Measure, 5.88% Precision and 4.88% Recall.

Figure 9.10 (B), along with Table 9.5 show that the contributions made by Hill-Climber along with ASEM+DNLE are statistically significant: Hill-Climber with ASEM+DNLE are statistically better than the bare-bones GA, ASEM and DNLE.

The computing time of the proposed system is 60 times real-time. However, the proposed system has several parallelization capabilities since a separate genetic algorithm is run on each audio segment. This way, the transcription task could be run on a separate CPU for each audio segment. Thus, we could have one master CPU to apply onset detection and distribute the resulting audio segments for several CPUs and wait for their results. Afterwards, the master CPU merges their results into one whole transcription and applies the hill-climber.

9.4.5 Impact of the onset detector on the overall results

By taking into account that onsets are used as guidelines for the segmentation of the input signal, the occurrence of false negatives might have a considerable impact in the final results of the event segregation. To evaluate the impact of the chosen onset detection algorithm, we ran the same experiments using the ground-truth onset information as the output of the onset detector. Figure 9.11 shows the comparison between the proposed

Table 9.5: Tukey-Kramer HSD (Honestly Significant Difference) Multi-Comparison Onset-offset metric

Algorithm	Algorithm	Lower Bound	Mean	Upper Bound	Significance
Hill-Climber	ASEM+DNLE	-0.7894	1.3125	3.4144	FALSE
Hill-Climber	ASEM	0.0231	2.1250	4.2269	TRUE
Hill-Climber	Bare-bones	1.2106	3.3125	5.4144	TRUE
Hill-Climber	DNLE	1.1481	3.2500	5.3519	TRUE
ASEM+DNLE	ASEM	-1.2894	0.8125	2.9144	FALSE
ASEM+DNLE	Bare-bones	-0.1019	2.0000	4.1019	FALSE
ASEM+DNLE	DNLE	-0.1644	1.9375	4.0394	FALSE
ASEM	Bare-bones	-0.9144	1.1875	3.2894	FALSE
ASEM	DNLE	-0.9769	1.1250	3.2269	FALSE
Bare-bones	DNLE	-2.1644	-0.0625	2.0394	FALSE

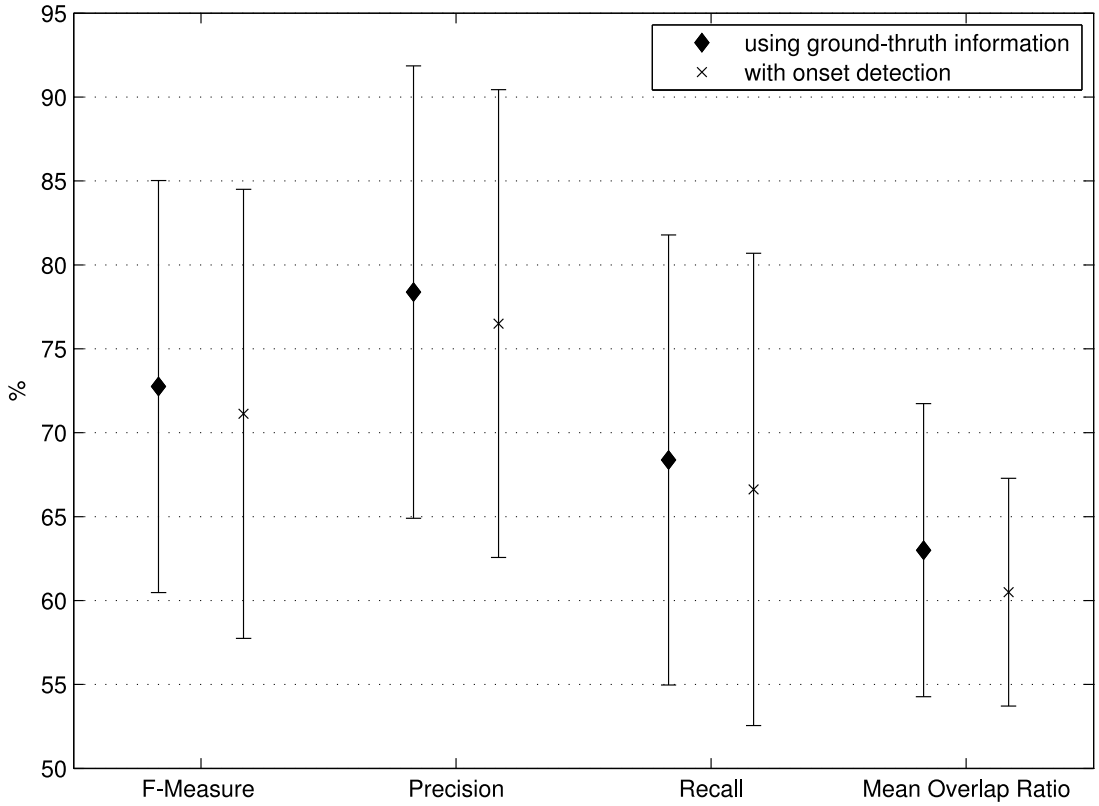

Figure 9.11: F-measure, Precision, Recall and Mean Overlap Ratio, using onset detection and ground-truth information.

system using the implemented onset detector and using the ideal onset detector.

The F-measure difference between using the ground-truth information rather than the onset detector is around 1%, which means that the implemented onset detection has good performance and does not have a significant impact in the proposed system: the performance of the algorithm does not drop significantly.

9.5 Summary

We have proposed a new approach for automatic transcription of polyphonic piano music using genetic algorithms for multi-pitch estimation and also for spectral envelope modeling and dynamic noise level estimation. The transcription process happens in three stages: first an onset detector is applied, so that the audio can be separated in several audio segments; then, for each segment, the genetic algorithm is applied to perform the transcription of the corresponding audio segment; and, finally, a hill-climber is applied to adjust note durations that cross multiple audio segments. The performance of the algorithm does not drop significantly when compared to the usage of the ideal onset detector. Nevertheless, since there is some decay in the quality of the results, the user is able to use any other onset detector and use its data as system input. Due to the fact that the audio segmentation is based on onset information, the duration of each segment is very short, which reduces the search space of the genetic algorithm. Thus, 50 generations are more than enough for the algorithm to find the appropriate solution. Each candidate solution is encoded as a set of discrete note events associated with a timbre and noise model. The evolution of these two models aids the transcription process because it mitigates the spectral differences between different instruments. The hill-climber, by adjusting the duration of the transcribed notes, gives a major contribution to the quality of the results from the perception point of view.

The performance of the method was measured using 7,860 audio files and was also compared to the state-of-the-art. The comparison was made using three different metrics: onset-only (to measure the ability of detecting the F0s), onset-offset (to measure the overlap between the original score and the transcribed score) and, finally, Hybrid Decay/-Sustain score for an evaluation from the human hearing perception point of view. The proposed method achieved satisfying results when compared to other algorithms on all metrics: it ranked as the best algorithm on the metric that best correlates with the human hearing perception - Hybrid Decay/Sustain Score - and it ranked as second best on both Onset-only and Onset-Offset metrics. Also, when compared to previous genetic algorithm approaches, the proposed system brings significant improvements on both computation time and quality of the results.

Chapter 10

Public Evaluations

The method presented along the previous chapter consists in applying an improved version of our Evolutionary Algorithm based approach to the Automatic Transcription of Music problem. As a best way to evaluate our proposal we aimed at submit our system to two specific contests, being one contest related to Music Information Retrieval (the domain of our proposal) and the other related to Evolutionary Algorithms (the technique employed); so that experts from both fields could evaluate the proposal.

This chapter presents the results obtained in both the Music Information Retrieval eXchange (MIREX) and Hummies contest, 2013, held at GECCO 2013 in Amsterdam.

10.1 MIREX

> *"The Music Information Retrieval Evaluation eXchange (MIREX) is the community-based framework for the formal evaluation of Music Information Retrieval (MIR) systems and algorithms."* (Downie, 2008b).

The Music Information Retrieval Evaluation eXchange (MIREX) (Downie, 2008b; Downie et al., 2010a) is an annual evaluation campaign for Music Information Retrieval (MIR) algorithms, coupled to the international conference of the International Society for Music Information Retrieval (ISMIR). MIREX is hosted by the International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL) at the Graduate School of Library Information Sciences (GSLIS), which is part of the University of Illinois at Urbana-Champaign (UIUC)[1].

MIREX allows researchers around the world to submit their algorithms for evaluation, see how they perform and also to compare their results with the results achieved by other researchers. MIREX plays a fundamental role on the comparison between different algorithms and is built upon three basic components:

- a set of standardized collections;

[1]See: `http://www.music-ir.org/mirex/wiki/MIREX_HOME`

- a set of standardized tasks/queries to be performed against these collections;
- a set of standardized evaluations of the results generated with regard to the tasks/-queries.

During 2006, MIREX marked the introduction of two important enhancements to the MIREX framework:

- tests of statistical significance (i.e., Friedman Test);
- the use of human evaluators (i.e., Evalutron 6000).

10.1.1 MIR Tasks Hosted on MIREX

MIR tasks evaluated at several MIREX editions include:

- Audio Train/Test Tasks
  - Audio US Pop Genre Classification
  - Audio Latin Genre Classification
  - Audio Music Mood Classification
  - Audio Classical Composer Identification
- Audio Cover Song Identification
- Audio Tag Classification
- Audio Music Similarity and Retrieval
- Symbolic Melodic Similarity
- Audio Onset Detection
- Audio Key Detection
- Real-time Audio to Score Alignment (a.k.a Score Following)
- Query by Singing/Humming
- Audio Melody Extraction
- Multiple Fundamental Frequency Estimation & Tracking
- Audio Chord Estimation
- Query by Tapping
- Audio Beat Tracking
- Structural Segmentation
- Audio Tempo Estimation

10.2 Multiple Fundamental Estimation & Tracking

Among all MIREX tracks, there is one that aims at evaluating Polyphonic Pitch Estimation and Automatic Music Transcription Algorithms. That is the Multiple Fundamental Estimation & Tracking track. According to this track organizers, *"That a complex music signal can be represented by the F0 contours of its constituent sources is a very useful concept for most music information retrieval systems. There have been many attempts at multiple (aka polyphonic) F0 estimation and melody extraction, a related area. The goal of multiple F0 estimation and tracking is to identify the active F0s in each time frame and to track notes and timbres continuously in a complex music signal."*. In this task, state-of-the-art multiple-F0 estimation and tracking algorithms are evaluated. Since F0 tracking of all sources in a complex audio mixture can be very hard, the problem is restricted to 3 cases:

1. Estimate active fundamental frequencies on a frame-by-frame basis.

2. Track note contours on a continuous time basis (as in audio-to-midi). This task also includes a piano transcription sub task.

3. Track timbre on a continuous time basis. This task is usually canceled due to lack of participation. It was run only on 2010.

10.2.1 Data

The data used for algorithm evaluation includes a woodwind quintet transcription of the fifth variation from Ludwig van Beethoven's Variations for String Quartet Op.18 No. 5. Each part (flute, oboe, clarinet, horn, or bassoon) was recorded separately while the performer listened to the other parts (recorded previously) through headphones. Later the parts were mixed to a monaural 44.1kHz/16bits file. The data also includes synthesized pieces using RWC MIDI and RWC samples, from the RWC Music Database (Goto and Nishimura, 2003). It also includes pieces from Classical and Jazz collections. Polyphony changes from 1 to 4 sources. Polyphonic piano recordings generated using a disklavier playback piano were also included in the evaluation dataset.

There are 6, 30-sec clips for each polyphony (2-3-4-5) for a total of 30 examples, plus there are 10 30-sec polyphonic piano clips. All files are in 44.1kHz / 16 bit wave format. There is also available a development, which can be found at *Development Set for MIREX 2007 MultiF0 Estimation Tracking Task*[2].

10.2.2 Evaluation

Both tracks Multiple Fundamental Frequency Estimation and Note Tracking are evaluated differently: the first is evaluated on a frame level basis and the latter is evaluated according to the note event data.

[2] `http://www.music-ir.org/evaluation/MIREX/data/2007/multiF0/index.htm`

Frame Level Evaluation

For Task 1 (frame level evaluation), submitted systems must report the number of active pitches every 10ms, so that precision (the portion of correct retrieved pitches for all pitches retrieved for each frame) and Recall (the ratio of correct pitches to all ground truth pitches for each frame) can be reported. A Returned Pitch is assumed to be correct if it is within a half semitone (± 3%) of a ground-truth pitch for that frame. Only one ground-truth pitch can be associated with each Returned Pitch. Also as suggested, an error score as described in Poliner and Ellis (2007b) is calculated. The frame level ground truth is calculated by the YIN algorithm (De Cheveigné and Kawahara, 2002) and hand corrected.

Besides Precision and Recall, each algorithm is also evaluated on the following metrics:

$$Acc = \frac{TP}{TP + FP + FN} \tag{10.1}$$

$$E_{tot} = \frac{\sum_{t=1}^{T} \max\left(N_{ref}\left(t\right), N_{sys}\left(t\right)\right) - N_{corr}\left(t\right)}{\sum_{t=1}^{T} N_{ref}\left(t\right)} \tag{10.2}$$

$$E_{subs} = \frac{\sum_{t=1}^{T} \min\left(N_{ref}\left(t\right), N_{sys}\left(t\right)\right) - N_{corr}\left(t\right)}{\sum_{t=1}^{T} N_{ref}\left(t\right)} \tag{10.3}$$

$$E_{miss} = \frac{\sum_{t=1}^{T} \max\left(0, N_{ref}\left(t\right) - N_{sys}\left(t\right)\right)}{\sum_{t=1}^{T} N_{ref}\left(t\right)} \tag{10.4}$$

$$E_{fa} = \frac{\sum_{t=1}^{T} \max\left(0, N_{sys}\left(t\right)\right) - N_{ref}\left(t\right)}{\sum_{t=1}^{T} N_{ref}\left(t\right)} \tag{10.5}$$

where N_{ref} is the number of non-zero elements in the ground truth data, N_{sys} is the number of active elements returned by the system, N_{corr} is the number of correctly identified elements and E stands for error.

Note Tracking

For Task 2 (note tracking), again Precision and Recall are reported. A ground truth note is assumed to be correctly transcribed if the system returns a note that is within a half semitone (± 3%) of that note **and** if the returned note's onset is within a 50ms range(± 25ms) of the onset of the ground truth note, and its offset is within 20% range of the ground truth note's offset. Again, one ground truth note can only be associated with one transcribed note.

The ground truth for this task is annotated by hand. An amplitude threshold relative to the file/instrument is determined: note onset is set to the time where its amplitude rises higher than the threshold and the offset is set to the time where the note's amplitude decays lower than the threshold. Moreover, the ground truth is set as the average F0 between the onset and the offset of the note. In the case of legato, the onset/offset is set to the time where the F0 deviates more than 3% of the average F0 through out the note up to that point. There are no vibratos larger than a half semitone in the test data.

This subtask is evaluated in two different ways. In the first setup, a returned note is assumed to be correct if its onset is within ±50ms of a reference note and its F0 is within ± quarter tone of the corresponding reference note, ignoring the returned offset values (Onset-only metric). In the second setup, on top of the above requirements, a correct returned note is required to have an offset value within 20% of the reference notes duration around the reference note's offset, or within 50ms, whichever is larger (Onset-Offset metric). The Overlap ratio is calculated for an individual correctly identified note as:

$$overlapratio = \frac{\min(offsets) - \max(onsets)}{\max(offsets) - \min(onsets)} \tag{10.6}$$

10.3 Multiple F0 Estimation and Tracking: Note Tracking Piano Subtask Results 2007-2011

The main issue about MIREX submissions and their respective results is that the submitted algorithms are only compared with those submitted in the same year. There is no information about how each algorithm compares to those submitted in previous editions.

This way, and to have a better idea of how our submission compares to the other previous submissions, we have compiled the results of all years, from 2007 to 2011. The approach presented on the previous chapter was submitted to 2011 edition of MIREX and had the identifier name "RFF 2011", where RFF stands for the name of the authors (Reis, Fernandéz and Ferreira) and 2011 is the submission year. Since we submitted two different versions of the same algorithm those are identified as "RFF1 2011" and "RFF2 2011".

Figure 10.1 shows the piano subtask results of the note tracking task of all previous submissions of MIREX. Results are presented as F-Measure on both Onset-only and Onset-Offset metrics. Most of the submissions presented in this figure are from the sames authors. For instance, RFF1 2011 and RFF2 2011 are two slightly different versions of the same algorithm, where the latter uses other kinds of pitched instruments besides piano, as internal samplers. As matter of fact several submissions represent exactly the same systems with slight changes. Moreover, different editions have also repeated submissions. As an example, both RK 2007 and RK 2008 submissions represent the same system. Furthermore, several teams submit the same system over the years but with improvements, where each new submission results in the improvements achieved during the last year of research. For instance, the RFF 2008 submission is a previous version (Reis et al., 2008) of our approach, that evolved over the years during our research process and was resubmitted in 2011 as RFF 2011, resulting in an F-Measure improvement of 25.25%. This way, and for a better comparison, we have discarded all redundant submissions and only considered the best algorithm of each team (see Figure 10.2). As in Figure 10.1, the results presented are F-Measure on both Onset-Onset and Onset-only metrics.

Figure 10.2 also shows that our approach is ranked on the Top 9, among all the 19 different state-of-the art algorithms, submitted since 2007. This way, we can conclude

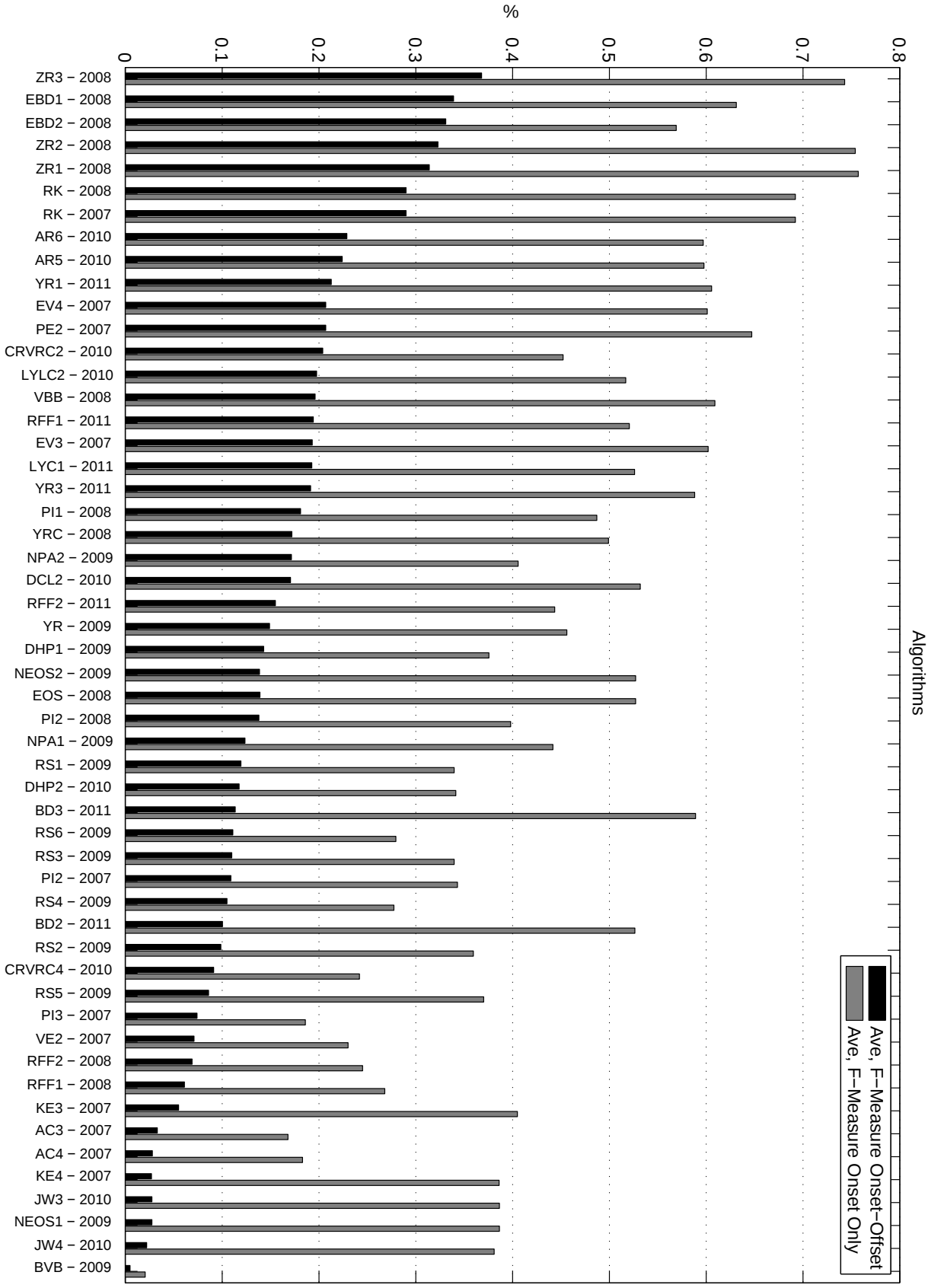

Figure 10.1: Mirex piano results.

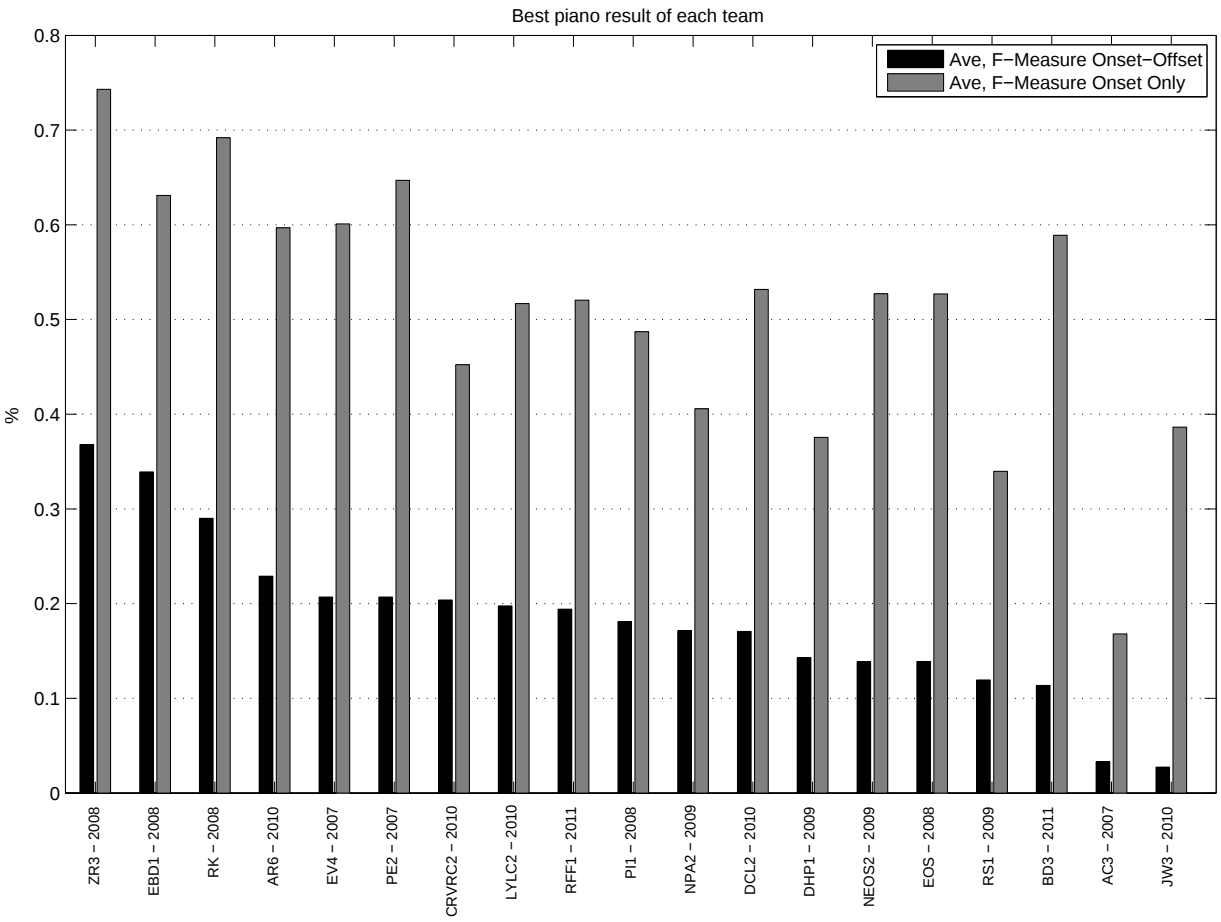

Figure 10.2: MIREX piano by teams.

that our approach achieves the state-of-the art performance, according to the MIREX Note Tracking Piano subtask. The 19 different algorithms submitted from 2007 to 2011 are the following:

ZR3 - 2008 Ruohua Zhou and Joshua D. Reiss, method described in: Zhou and Reiss (2008); Zhou (2006); Zhou and Mattavelli (2007);

EBD1 - 2008 Valentin Emiya, Roland Badeau and Bertrand David. Method described in: Emiya et al. (2008b,a); Emiya (2008); Emiya et al. (2010);

RK - 2008 Matt Ryynänen, Anssi Klapuri. Method described in: Ryynänen and Klapuri (2008, 2005);

AR6 - 2010 Chunghsin Yeh and Alex Roebel. Method described in: Yeh and Roebel (2010); Yeh et al. (2010); Raiford (1942); Chang et al. (2008); Yeh et al. (2005);

EV4 - 2007 Emmanuel Vincent, Nancy Bertin and Roland Badeau. Method described in: Vincent et al. (2007); Bertin et al. (2007); Smaragdis and Brown (2003); Vincent et al. (2008);

PE2 - 2007 Graham Poliner and Daniel Ellis. Method described in: Poliner and Ellis (2007b,a);

CRVRC2 - 2010 Francisco Jesus Canadas Quesada, Francisco José Rodriguez Serrano, Pedro Vera-Candeas, Nicolas Ruiz Reyes and Julio José Carabias-Orti. Method described in: Canadas et al. (2010); Canadas Quesada et al. (2010);

LYLC2 - 2010 Cheng-Te Lee, Yi-Hsuan Yang, Keng-Sheng Lin and Homer Chen. Method described in: Lee et al. (2010); Chen (2011);

RFF1 - 2011 Gustavo Reis, Francisco Fernandéz and Aníbal Ferreira. Method described on the previous chapter (Reis et al. (2012));

PI1 - 2008 Antonio Pertusa, José M. Iñesta. Method described in: Pertusa and Inesta (2008b,a);

NPA2 - 2009 Paolo Nesi, Gianni Pantaleo and Fabrizio Argenti. Method described in: Nesi et al. (2009b); Argenti et al. (2011); Nikias and Mendel (1993); Nikias and Raghuveer (1987);

DCL2 - 2010 Arnaud Dessein, Arshia Cont and Guillaume Lemaitre. Method described in: Dessein et al. (2010a,b);

DHP1 - 2009 Zhiyao Duan, Jinyu Han and Bryan Pardo. Method described in: Duan et al. (2009b,a, 2008);

NEOS2 - 2009 Masahiro Nakano, Koji Egashira, Nobutaka Ono and Shigeki Sagayama. Method described in: Nakano et al. (2009); Kameoka et al. (2007);

EOS - 2008 Kogi Egashira, Nobutaka Ono and Shigeki Sagayama. Method described in: Egashira et al. (2008); Kameoka et al. (2007);

RS1 - 2009 Stanislaw Andrzej Raczyński and Shigeki Sagayama. Method described in: Raczyński and Sagayma (2009); Raczyński et al. (2007);

BD3 - 2011 Emmanouil Benetos and Simon Dixon. Method described in: Benetos and Dixon (2011a,b); Smaragdis et al. (2008);

AC3 - 2007 Arshia Cont. Method described in: Cont (2007, 2006b); Cont et al. (2007);

JW3 - 2010 Jun Wu, Nobutaka Ono and Shigeki Sagayama. Method described in: Wu et al. (2010);

10.3.1 Chroma Evaluation

Systems submitted after 2008 were also evaluated on chroma results, where all F0's are mapped to a single octave before evaluating. Figure 10.3 shows these Chroma F-Measure values. As we can see, our approach achieves the 5th place, on the Onset-Offset metric, according to chroma results. This means that our approach is efficient in transcribing non-harmonically related notes.

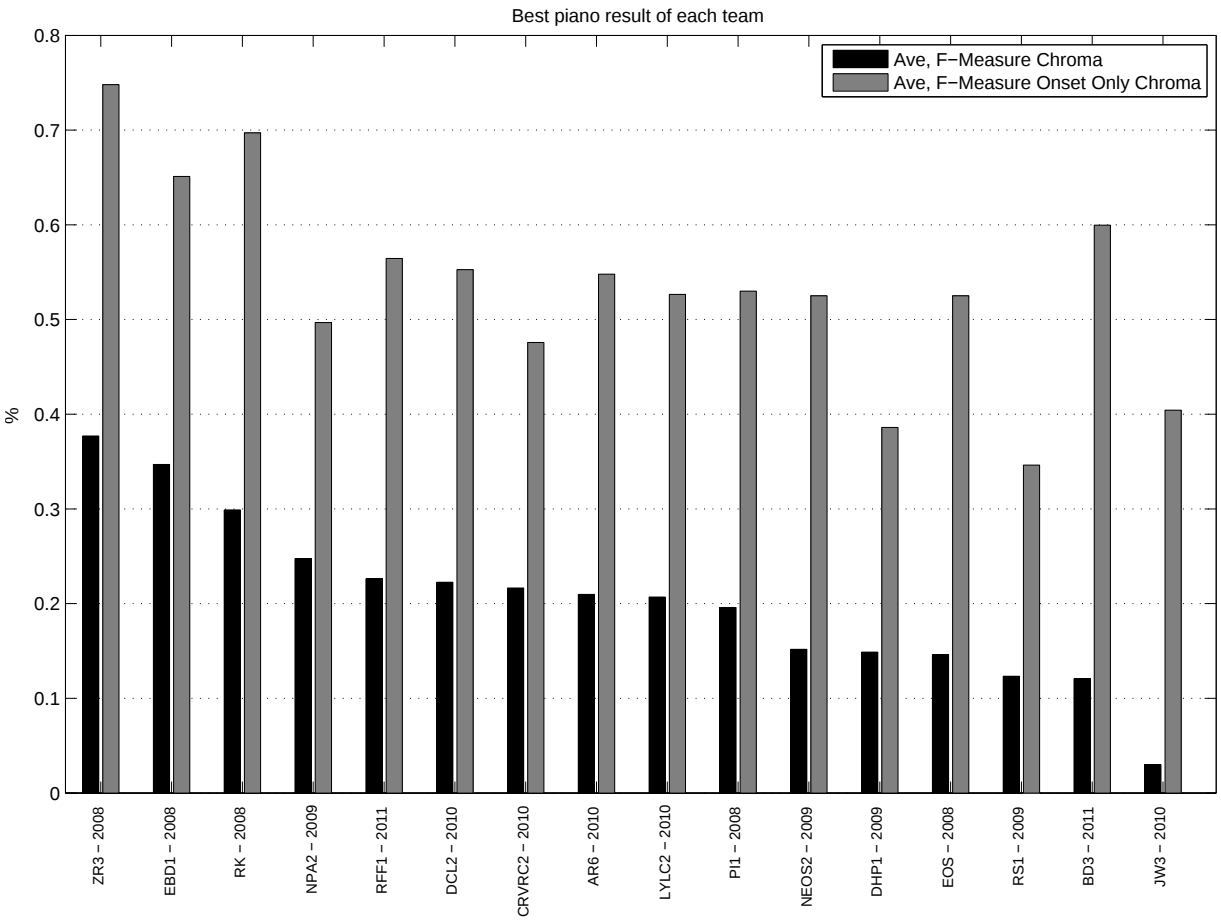

Figure 10.3: MIREX piano by teams - chroma evaluation.

10.4 Humies Awards

The *"Humies" Awards for Human-Competitive Results Produced by Genetic and Evolutionary Computation*[3] is an award for researchers that employ techniques of evolutionary computation to difficult real-world problems and have results that are human-competitive.

Entries are solicited for awards totaling $10,000. Any method presenting human-competitive results and that have been produced by any form of genetic and evolutionary computation and that have been published in the open literature between the deadline of the previous Humies competition and the deadline for the current competition can be submitted for the awards.

The prize competition is based on published results. The publication may be a paper at any conference, a paper published anywhere in the open literature, or a paper in final form that has been unconditionally accepted by a publication and is actually and unconditionally "in press" (that is, the entry must be identical to something that will be published imminently). Also, the submitted publication must meet the usual standards of a scientific publication in that is must clearly describe a problem, the methods used to address the problem, the results obtained, and sufficient information to enable the work described to be replicated by an independent person.

[3]see `http://www.genetic-programming.org/hc2005/main.html`

10.4.1 Human-Competitiveness

An automatically created result is considered "human-competitive" if it satisfies at least one of the eight criteria below:

A The result was patented as an invention in the past, is an improvement over a patented invention, or would qualify today as a patentable new invention.

B The result is equal to or better than a result that was accepted as a new scientific result at the time when it was published in a peer-reviewed scientific journal.

C The result is equal to or better than a result that was placed into a database or archive of results maintained by an internationally recognized panel of scientific experts.

D The result is publishable in its own right as a new scientific result independent of the fact that the result was mechanically created.

E The result is equal to or better than the most recent human-created solution to a long-standing problem for which there has been a succession of increasingly better human-created solutions.

F The result is equal to or better than a result that was considered an achievement in its field at the time it was first discovered.

G The result solves a problem of indisputable difficulty in its field.

H The result holds its own or wins a regulated competition involving human contestants (in the form of either live human players or human-written computer programs).

10.4.2 Submission

We presented a submission of the method (Reis et al., 2012), based on the following criteria:

B and E Several state-of-the-art algorithms published in peer-reviewed scientific journals were compared with our approach and our algorithm ranked first with the metric that best resembles the human earing perception, also it ranked as the 2nd best algorithm on the other two metrics: Onset-Offset and Onset only (see Chapter 9, Section 9.4.3). Furthermore, our proposal was also ranked as 2nd best algorithm on MIREX 2011 contest (i.e. 2nd best algorithm among all MIREX 2011 submissions), on Piano subtask (Multiple Fundamental Frequency Estimation and Tracking).

H Our algorithm was submitted to MIREX 2011, Piano subtask (Multiple Fundamental Estimation and Tracking) and ranked as 2nd best. The other competing algorithms were created by humans.

Our method was selected as one of the 10 finalists, among 14 entries, where it achieved the 4th place[4].

[4]The list of the selected finalists is available on:
`http://http://www.genetic-programming.org/combined.php`

10.5 Summary

Considering the best submission of each team of researchers to MIREX, our method ranks 9 out of 19 on piano transcription. Moreover, if we take into account the chroma results, our method ranks the 5th place, which means that the algorithm can cope with non-harmonically related notes. From the perception point of view, our method is on the top 5 among the state of the art. This makes us believe that Genetic Algorithms are a valid approach and present competitive results, when compared to other state-of-the-art algorithms. This was also the opinion of the Humies committee, when selecting our research as one of the 7 finalists for the $5,000 gold prize.

RFF2 2011 MIREX submission presented slightly worst results because it was blindly adapted to multi-timbral (it is exactly the same algorithm than RFF1 2011, but with a larger sample database). Since RFF2 deals with a bigger database of internal samples, the size of the search space is much greater and this requires different parameters, namely a greater number of generations to converge to the desired solution.

Chapter 11

Conclusions and Future Work

11.1 Conclusions

1. During our research, we have employed several genetic algorithm approaches to address the problem of multi-pitch estimation. We first started with simple synthesized models of instruments. Then, we moved to real audio recordings and performed several experiments. Those experiments included different domains (log spectra, linear spectra, filter banks, real cepstrum, hybrid cepstral and spectral analysis, ACF and SACF) for audio similarity measurement and several error measurements (from Hamming and Itakura-Saito distance to area intersection, correlation and other variations). We faced the problem of Harmonic Overfitting, which is related to timbre differences, and proposed a spectral envelope modeling technique to address this issue. Furthermore, we have employed this approach on musical signals with different audio instruments to show the feasibility of the approach on multi-timbral music.

2. During our research, we have also made contributions to the field of Evolutionary Computation, namely: the evolutionary algorithm **Gene Fragment Competition**, which can be used is most decomposable problems in signal or image processing. We presented an analysis of how decomposable approaches are suitable to decomposable problems and took advantage of the modular and hierarchical structure of the Royal Road functions so we could use them as test functions to show how single-population decomposable approaches, such as the Gene Fragment Competition, can overcome the spurious correlation or hitchhiking. We have shown empirically that both Parisian approach and Gene Fragment Competition clearly outperform not only the standard genetic algorithm and the multiple-population co-evolutionary approach but also the random mutation hill-climber, except for the GFC in the instances with string length $L = 256$. Hitchhiking is known to be, in general, one of the major bottlenecks of the genetic algorithms performance. This way, avoiding hitchhiking boosts the performance of the algorithm. Applying problem decomposition in building blocks is an advantageous optimization technique, since this avoids the hitchhiking phenomena. Despite the random mutation hill-climber algorithm

has been revealed to be the ideal for the Royal Road functions in the past, we have shown that single population decomposable approaches can explore more efficiently the search space on Royal Road functions.

3. We have also proposed a new approach for automatic transcription of polyphonic piano music using genetic algorithms for **multi-pitch** estimation and also for **spectral envelope modeling** and dynamic **noise level estimation**. This transcription method happens in three stages: first an onset detector is applied, so that the audio can be separated in several audio segments; then, for each segment, a genetic algorithm is applied to perform the transcription of the corresponding audio segment; and, finally, a hill-climber is applied to adjust note durations that cross multiple audio segments. We have shown that the performance of this algorithm does not drop significantly when compared with the usage of the ideal onset detector. Nevertheless, since there is some decay in the quality of the results, one can use any other onset detector and fed its data to the system input. Due to the fact that the audio segmentation is based on onset information, the duration of each segment is very short, which reduces the search space of the genetic algorithm. This way, we have shown that 50 generations are more than enough for the algorithm to find an appropriate solution. Each candidate solution is encoded as a set of discrete note events associated with a spectral envelope and noise model. The evolution of these two models aids the transcription process because it mitigates the spectral differences between different instruments. The hill-climber, by adjusting the duration of the transcribed notes, gives a major contribution to the quality of the results from the perception point of view. The performance of the method was measured using 7,860 audio files and was also compared to the state-of-the-art. The comparison was made using three different metrics: onset-only (to measure the ability of detecting the F0s), onset-offset (to measure the overlap between the original score and the transcribed score) and, finally, Hybrid Decay/Sustain score for an evaluation from the human hearing perception point of view. The proposed method achieved satisfying results when compared to other algorithms on all metrics: it ranked as the best algorithm on the metric that best correlates with the human hearing perception - Hybrid Decay/Sustain Score - and it ranked as second best on both Onset-only and Onset-Offset metrics. Also, when compared to previous genetic algorithm approaches, the proposed system brings significant improvements on both computation time and quality of the results.

11.2 Future Work

For future steps, we would like to propose a more general approach, i.e. not sample based, to the problem addressed. The current method relies on an internal synthesizer consisting of previous recorded audio samples, which are used to search for the most-likely combination of pitches. Spectral envelope modeling is also done, along with the pitch estimation, to avoid spurious notes due to timbre differences. One interesting feature

should be to evolve from scratch the entire harmonic model, instead of starting from previous recorded samples. This way, the system would be general enough to transcribe any kind of harmonic model.

We also intend to exploit the parallelization capabilities of the proposed system to reduce its computing time.

Planning ahead we are looking forward to apply a deeper analysis on both Gene Fragment Competition and Parisian approaches and see how they perform in several classical problems in comparison to standard evolutionary algorithms.

We also aim to apply Gene Fragment Competition and Parisian approaches to real world problems and study how they perform when compared with classical approaches.

Bibliography

(2007). Music information retrieval evaluation exchange (mirex 2007). http://www.music-ir.org/mirex/2007/index.php. [cited at p. 111]

Abdallah, S. and Plumbley, M. (2003). An independent component analysis approach to automatic music transcription. *PREPRINTS-AUDIO ENGINEERING SOCIETY*. [cited at p. 46]

Abdallah, S. and Plumbley, M. (2004). Polyphonic music transcription by non-negative sparse coding of power spectra. In *Proceedings of the Fifth International Conference on Music Information Retrieval (ISMIR04)*, pages 10–14. [cited at p. 46, 56]

Alonso, M., Richard, G., and David, B. (2005). Extracting note onsets from musical recordings. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, page 4 pp. [cited at p. 52]

Anderson, R. (1953). Recent advances in finding best operating conditions. *Journal of the American Statistical Association*, 48(264):789–798. [cited at p. 64]

Argenti, F., Nesi, P., and Pantaleo, G. (2011). Automatic transcription of polyphonic music based on the constant-q bispectral analysis. [cited at p. 172]

Association, M. M. (1999/2008). *Complete MIDI 1.0 Detailed Specification*. [cited at p. 12, 14, 15, 85, 86]

Ausiello, G., Protasi, M., Marchetti-Spaccamela, A., Gambosi, G., Crescenzi, P., and Kann, V. (1999). *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition. [cited at p. 59]

Bach, F. and Jordan, M. (2005). Discriminative training of hidden markov models for multiple pitch tracking [speech processing examples]. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 5, pages 489–492. IEEE. [cited at p. 41]

Bäck, T. (1993). Optimal mutation rates in genetic search. In *Proceedings of the fifth International Conference on Genetic Algorithms*, pages 2–8. Morgan Kaufmann. [cited at p. 65]

Baker, T., Gill, J., and Solovay, R. (1975). Relativizations of the p=?np question. *SIAM Journal on computing*, 4(4):431–442. [cited at p. 61]

Baskind, A. and de Cheveigné, A. (2012). Pitch-tracking of reverberant sounds, application to spatial description of sound scenes. *Watermark*, 1. [cited at p. 37]

Bay, M. and Beauchamp, J. W. (2006). Harmonic source separation using prestored spectra. In *ICA*, pages 561–568. [cited at p. 116]

Beauchamp, J., Maher, R., and Brown, R. (2012). Detection of musical pitch from recorded solo performances. *Watermark*, 1. [cited at p. 37]

Bello, J., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., and Sandler, M. (2005). A tutorial on onset detection in music signals. *Speech and Audio Processing, IEEE Transactions on*, 13(5):1035–1047. [cited at p. 37]

Bello, J., Daudet, L., and Sandler, M. (2002). Time-domain polyphonic transcription using self-generating databases. In *112th Convention of the Audio Engineering Society*, Munich, Germany. [cited at p. 45]

Bello, J. and Sandler, M. (2000). Blackboard system and top-down processing for the transcription of simple polyphonic music. In *Proceedings of the COST G-6 Conference on digital Audio Effects (DAFX-00)*. [cited at p. 45, 53, 56]

Bello, J. P. (2003). *Towards the automated analysis of simple polyphonic music: A knowledge-based approach*. PhD thesis, University of London, London, UK. [cited at p. 4]

Benetos, E. and Dixon, S. (2011a). Multiple-f0 estimation and note tracking using a convolutive probabilistic model. In *Proceedings of the Fifth Music Information Retrieval Evaluation eXchange (MIREX 2011)*, Miami. [cited at p. 172]

Benetos, E. and Dixon, S. (2011b). Multiple-instrument polyphonic music transcription using a convolutive probabilistic model. *8th Sound and Music Computing Conf*, page 1924. [cited at p. 45, 172]

Bertin, N., Badeau, R., and Richard, G. (2007). Blind signal decompositions for automatic transcription of polyphonic music: NMF and K-SVD on the benchmark. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 65–68. [cited at p. 46, 154, 171]

Bogert, B., Healy, M., and Tukey, J. (1963). The quefrency alanysis of time series for echoes: Cepstrum, Pseudo-Autocovariance, Cross-Cepstrum and Saphe Cracking. In *Proc. Symp. on Time Series Analysis*, pages 209–243. [cited at p. 149]

Bregman, A. (1990). *Auditory scene analysis*. MIT press Cambridge, Mass. [cited at p. 23]

Bremermann, H. J. (1962). Optimization through evolution and recombination. In Yovits, M. C., Jacobi, G. T., and Golstine, G. D., editors, *Proceedings of the Conference on Self-Organizing Systems – 1962*, pages 93–106, Washington, DC. Spartan Books. [cited at p. 64]

Brown, A. R. and Sorensen, A. C. (2000). Introducing jMusic. In Brown, A. R. and Wilding, R., editors, *Australasian Computer Music Conference*, pages 68–76, Queensland University of Technology, Brisbane. ACMA. [cited at p. 92]

Brown, J. (1991). Calculation of a constant Q spectral transform. *The Journal of the Acoustical Society of America*, 89:425. [cited at p. 48]

Brown, J. (1992). Musical fundamental frequency tracking using a pattern recognition method. *The Journal of the Acoustical Society of America*, 92:1394. [cited at p. 29]

Burred, J., Röbel, A., and Rodet, X. (2006). An accurate timbre model for musical instruments and its application to classification. In *Workshop on Learning the Semantics of Audio Signals, Athens, Greece*. [cited at p. 35]

Canadas, F., Rodriguez, F., Vera, P., Ruiz, N., and Carabias, J. (2010). Multiple fundamental frequency estimation & tracking in polyphonic music for mirex 2010. In *Proceedings of the Fourth Music Information Retrieval Evaluation eXchange (MIREX 2010)*, Utrecht. [cited at p. 172]

Canadas-Quesada, F., Vera-Candeas, P., Ruiz-Reyes, N., and Carabias-Orti, J. (2009). Polyphonic transcription based on temporal evolution of spectral similarity of gaussian mixture models. In *17th European Signal Processing Conference (EUSIPCO)*, pages 10–14. Citeseer. [cited at p. 50]

Cañadas-Quesada, F., Vera-Candeas, P., Ruiz-Reyes, N., Mata-Campos, R., and Carabias-Orti, J. (2008). Note-event detection in polyphonic musical signals based on harmonic matching pursuit and spectral smoothness. *Journal of New Music Research*, 37(3):167–183. [cited at p. 42]

Canadas Quesada, F. J., Ruiz Reyes, N., Vera Candeas, P., Carabias, J. J., and Maldonado, S. (2010). A multiple-f0 estimation approach based on gaussian spectral modelling for polyphonic music transcription. *Journal of New Music Research*, 39(1):93–107. [cited at p. 172]

Cemgil, A. (2004). *Bayesian music transcription*. PhD thesis, Radboud University of Nijmegen, Netherlands. [cited at p. 4]

Cemgil, A., Kappen, B., and Barber, D. (2003). Generative model based polyphonic music transcription. In *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, pages 181–184. IEEE. [cited at p. 51]

Cemgil, A., Kappen, H., and Barber, D. (2006a). A generative model for music transcription. *IEEE Transactions on Audio Speech and Language Processing*, 14(2):679. [cited at p. 74]

Cemgil, A. T., Kappen, H. J., and Barber, D. (2006b). A Generative Model for Music Transcription. *IEEE Transactions on Audio, Speech and Language Processing*, 14(2):679–694. [cited at p. 3, 51, 153]

Chafe, C. and Jaffe, D. (1986). Source separation and note identification in polyphonic music. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86.*, volume 11. [cited at p. 54]

Chang, W.-C., Su, A. W., Yeh, C., Roebel, A., and Rodet, X. (2008). Multiple-f0 tracking based on a high-order hmm model. In *Digital Audio Effects (DAFx-08)*, Espoo, Finland. [cited at p. 49, 171]

Chen, H. (2011). Automatic transcription of piano music by sparse representation of magnitude spectra. *2011 IEEE International Conference on Multimedia and Expo*, pages 1–6. [cited at p. 172]

Chien, Y. and Jeng, S. (2002). An automatic transcription system with octave detection. In *IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS SPEECH AND SIGNAL PROCESSING*, volume 2. IEEE; 1999. [cited at p. 54]

Collet, P., Lutton, E., Schoenauer, M., Collet, P., Lutton, E., Raynal, F., and Schoenauer, M. (2000). Polar ifs + parisian genetic programming = efficient ifs inverse problem solving. *Genet. Programm. Evolvable Mach. J*, 1:361. [cited at p. 129, 131]

Comon, P. (1994). Independent component analysis, a new concept? *Signal processing*, 36(3):287–314. [cited at p. 46]

Conklin Jr, H. (1999). Generation of partials due to nonlinear mixing in a stringed instrument. *The Journal of the Acoustical Society of America*, 105:536. [cited at p. 36]

Cont, A. (2006a). Realtime multiple pitch observation using sparse non-negative constraints. In *Proc. Int. Conf. on Music Information Retrieval (ISMIR)*, pages 206–212. [cited at p. 46]

Cont, A. (2006b). Realtime multiple pitch observation using sparse non-negative constraints. *Signal Processing*, pages 206–211. [cited at p. 172]

Cont, A. (2007). Real-time transcription of music signals: Mirex 2007 submission description. *Proc of the 3rd Music Information Retrieval Evaluation eXchange (MIREX)*. [cited at p. 172]

Cont, A., Dubnov, S., and Wessel, D. (2007). Realtime multiple-pitch and multiple-instrument recognition for music signals using sparse non-negative constraints. In *Proceedings of Digital Audio Effects Conference (DAFx)*, pages 10–12. [cited at p. 172]

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM. [cited at p. 61, 62]

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297. [cited at p. 47]

Darwin, C. (1859). *On the Origin of the Species by Natural Selection*. Murray, London, UK. [cited at p. 62, 68]

Daudet, L. (2004). Sparse and structured decompositions of audio signals in overcomplete spaces. In *Proc. Intl Conference on Digital Audio Effects (DAFx), Naples, Italy*, pages 22–26. [cited at p. 37]

Davy, M. (2006). Multiple fundamental frequency estimation based on generative models. *Signal Processing methods for music transcription*, pages 203–227. [cited at p. 50]

Davy, M. and Godsill, S. (2003). Bayesian harmonic models for musical signal analysis. *Bayesian Statistics*, 7. [cited at p. 50]

De Cheveigné, A. (1993). Separation of concurrent harmonic sounds: Fundamental frequency estimation and a time-domain cancellation model of auditory processing. *JOURNAL-ACOUSTICAL SOCIETY OF AMERICA*, 93:3271–3271. [cited at p. 40, 43]

de Cheveigné, A. (2005). Pitch perception models. *Springer Handbook of Auditory Research*, 24:169. [cited at p. 30, 43, 55]

De Cheveigné, A. and Kawahara, H. (2001). Comparative evaluation of f0 estimation algorithms. In *Proc. Eurospeech*, volume 1, pages 2451–2454. [cited at p. 25]

De Cheveigné, A. and Kawahara, H. (2002). Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111:1917. [cited at p. 25, 168]

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, Ann Arbor, MI, USA. AAI7609381. [cited at p. 65]

De Jong, K. A. and Spears, W. M. (1989). Using genetic algorithms to solve np-complete problems. In *ICGA*, pages 124–132. [cited at p. 63]

Dessein, A., Cont, A., and Lemaitre, G. (2010a). Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence. In *6th Music Information Retrieval Evaluation eXchange (MIREX)*, Utrecht, Netherlands. [cited at p. 46, 172]

Dessein, A., Cont, A., Lemaitre, G., and Umr, I. C. (2010b). Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence. *October*, (Ismir):489–494. [cited at p. 172]

Donoho, D. (2006). For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution. *Communications on pure and applied mathematics*, 59(6):797–829. [cited at p. 45]

Doval, B. and Rodet, X. (1991). Estimation of fundamental frequency of musical sound signals. In *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, pages 3657–3660. IEEE. [cited at p. 29]

Downie, J., Ehmann, A., Bay, M., and Jones, M. (2010a). The music information retrieval evaluation exchange: Some observations and insights. *Advances in Music Information Retrieval*, pages 93–115. [cited at p. 104, 165]

Downie, J., Ehmann, A., Bay, M., and Jones, M. (2010b). The music information retrieval evaluation exchange: Some observations and insights. In Ras, Z. and Wieczorkowska, A., editors, *Advances in Music Information Retrieval*, volume 274 of *Studies in Computational Intelligence*, pages 93–115. Springer Berlin / Heidelberg. [cited at p. 154]

Downie, J. S. (2008a). The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255. [cited at p. 154]

Downie, J. S. (2008b). The music information retrieval evaluation exchange (2005-2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255. [cited at p. 104, 119, 165]

Droste, S., Jansen, T., and Wegener, I. (1998). A rigorous complexity analysis of the (1 + 1)- evolution strategy for separable functions with boolean inputs. In *In Proceedings of the 5th IEEE International Conference on Evolutionary Computation (ICEC'98)*, pages 499–504. IEEE Press, Piscataway (NJ). [cited at p. 65]

Duan, Z., Han, J., and Pardo, B. (2009a). Harmonically informed multi-pitch tracking. In *Proc. ISMIR*. [cited at p. 52, 172]

Duan, Z., Han, J., and Pardo, B. (2009b). A multi-pitch tracking system (mirex 2009). In *Proceedings of the Fifth Music Information Retrieval Evaluation eXchange (MIREX 2009)*, Kobe. [cited at p. 172]

Duan, Z. D. Z., Zhang, Y. Z. Y., Zhang, C. Z. C., and Shi, Z. S. Z. (2008). Unsupervised single-channel music source separation by average harmonic structure modeling. [cited at p. 172]

Duifhuis, H., Willems, L., and Sluyter, R. (1982). Measurement of pitch in speech: An implementation of goldsteins theory of pitch perception. *The Journal of the Acoustical Society of America*, 71:1568. [cited at p. 29]

Dunn, E., Olague, G., and Lutton, E. (2005). Automated photogrammetric network design using the parisian approach. In *8th European Workshop on Evolutionary Computation in Image Analysis and Signal Processing. Lecture Notes in Computer Science*, volume 3449, pages 356–365. Springer. [cited at p. 131]

Dunn, E., Olague, G., and Lutton, E. (2006). Parisian camera placement for vision metrology. *Pattern Recognition Letters*, 27(11):1209–1219. [cited at p. 131]

Egashira, K., Ono, N., and Sagayama, S. (2008). Sequential estimation of multiple fundamental frequencies through harmonic-temporal-structured clustering. *Proc of the 4th Music Information Retrieval Evaluation eXchange (MIREX)*. [cited at p. 172]

Eggink, J. and Brown, G. J. (2003). Application of missing feature theory to the recognition of musical instruments in polyphonic audio. In *ISMIR*. [cited at p. 51]

Ellis, D. (1996). *Prediction-driven computational auditory scene analysis*. PhD thesis, Massachusetts Institute of Technology. [cited at p. 54]

Emiya, V. (2008). *Transcription automatique de la musique de piano*. These, Télécom ParisTech. [cited at p. 4, 151, 152, 171]

Emiya, V., Badeau, R., and David, B. (2008a). Automatic transcription of piano music based on HMM tracking of jointly-estimated pitches. In *Proc. of European Conference on Signal Processing (EUSIPCO)*. [cited at p. 52, 144, 154, 171]

Emiya, V., Badeau, R., and David, B. (2008b). Automatic transcription of piano music based on HMM tracking of jointly-estimated pitches. In *Proceedings of the Fourth Music Information Retrieval Evaluation eXchange (MIREX 2008)*, Philadelphia, PA, United States. [cited at p. 171]

Emiya, V., Badeau, R., and David, B. (2010). Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(6):1643 –1654. [cited at p. 52, 144, 152, 171]

Engelmore, R. and Morgan, T. (1988). *Blackboard systems: edited by Robert Engelmore, Tony Morgan.* Insight series in artificial intelligence. Addison-Wesley. [cited at p. 53]

Eronen, A. (2003). Musical instrument recognition using ica-based transform of features and discriminatively trained hmms. In *Signal Processing and Its Applications, 2003. Proceedings. Seventh International Symposium on*, volume 2, pages 133–136. IEEE. [cited at p. 51]

Every, M. and Szymanski, J. (2004). A spectral-filtering approach to music signal separation. In *Proc. DAFx*, pages 197–200. [cited at p. 33]

Fernandez-Cid, P.; Casajus-Quiros, F. (1998). "multi-pitch estimation for polyphonic musical signals". [cited at p. 54]

Févotte, C., Bertin, N., and Durrieu, J.-L. (2009). Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis. *Neural computation*, 21(3):793–830. [cited at p. 74]

FitzGerald, D. (2004). *Automatic drum transcription and source separation.* PhD thesis, Dublin Institute of Technology. [cited at p. 46]

Fletcher, H., Blackham, E. D., and Stratton, R. (1962). Quality of piano tones. *The Journal of the Acoustical Society of America*, 34(6):749–761. [cited at p. 144]

Fletcher, N. H. and Rossing, T. D. (1998). *The physics of musical instruments / Neville H. Fletcher, Thomas D. Rossing.* Springer, New York ; London :, 2nd ed. edition. [cited at p. 36, 144]

Fogel, D. B. (1992). *Evolving artificial intelligence.* PhD thesis, La Jolla, CA, USA. UMI Order No. GAX93-03240. [cited at p. 64]

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution.* John Wiley, New York, USA. [cited at p. 64]

Forrest, S. and Mitchell, M. (1993). Relative building-block fitness and the building-block hypothesis. In *Foundations of Genetic Algorithms 2*, pages 109–126. Morgan Kaufmann. [cited at p. 132, 133, 134, 135, 139]

Fortnow, L. (2009). The status of the p versus np problem. *Communications of the ACM*, 52(9):78–86. [cited at p. 61]

Fraser, A. S. (1957). Simulation of genetic systems by automatic digital computers. II. Effects of linkage on rates of advance under selection. *Australian Journal of Biological Science*, 10:492–499. [cited at p. 64]

Friedberg, R. M. (1958). A learning machine: Part i. *IBM Journal of Research and Development*, 2(1):2 –13. [cited at p. 64]

Gabor, D. (1946). Theory of communication. *J. of the Institute of Electrical Engineers Part III*, 93:429–457. [cited at p. 42]

Gabor, D. (1947). Acoustical quanta and the theory of hearing. *Nature*, 159(4044):591–594. [cited at p. 42]

Garcia, G. (2001). A genetic search technique for polyphonic pitch detection. In *Proceedings of the International Computer Music Conference (ICMC)*, Havana, Cuba. [cited at p. 81, 82, 84, 85, 86, 89, 91, 95]

Gersting, J. L. and Schneider, G. M. (1995). An invitation to computer science. [cited at p. 57]

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional. [cited at p. 69, 70, 78, 81, 83, 84, 87, 90, 100, 110, 147, 148]

Goldstein, J. (1973). An optimum processor theory for the central formation of the pitch of complex tones. *The Journal of the Acoustical Society of America*, 54(6):1496–1516. [cited at p. 29]

Gómez, E., Klaupuri, A., and Meudic, B. (2003). Melody description and extraction in the context of music content processing. *Journal of New Music Research*, 32(1). [cited at p. 85]

Goto, M. (2000). A robust predominant-f0 estimation method for real-time detection of melody and bass lines in cd recordings. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 2, pages II757–II760. IEEE. [cited at p. 51]

Goto, M. and Nishimura, T. (2003). Rwc music database: Music genre database and musical instrument sound database. pages 229–230. [cited at p. 114, 116, 167]

Gribonval, R. and Bacry, E. (2003). Harmonic decomposition of audio signals with matching pursuit. *Signal Processing, IEEE Transactions on*, 51(1):101–111. [cited at p. 42]

Groble, M. (2008). Multiple fundamental frequency estimation. In *Proceedings of the Fourth Music Information Retrieval Evaluation eXchange (MIREX)*, pages 1–4, Philadelphia, USA. [cited at p. 45]

Hainsworth, S. W. (2003). *Techniques for the automated analysis of musical audio*. PhD thesis, University of Cambridge. [cited at p. 4, 54]

Harris, C. and Weiss, M. (1963). Pitch extraction by computer processing of high-resolution fourier analysis data. *The Journal of the Acoustical Society of America*, 35(3):339–343. [cited at p. 29]

Harris, F. J. (1978). On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83. [cited at p. 21, 98]

Hart, W., Krasnogor, N., and Smith, J. (2004). *Recent Advances in Memetic Algorithms*, chapter Memetic Evolutionary Algorithms. Springer. [cited at p. 101, 122, 150]

Hartmanis, J. and Stearns, R. E. (1965). On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306. [cited at p. 59]

Hartmann, W. (1997). *Signals, Sound, and Sensation*. Modern Acoustics and Signal Processing. American Inst. of Physics. [cited at p. 23]

Hartmann, W. M. (1996). Pitch, periodicity, and auditory organization. *The Journal of the Acoustical Society of America*, 100:3491. [cited at p. 23]

Hermansky, H. and Morgan, N. (1994). RASTA processing of speech. *IEEE transactions on speech and audio processing*, 2(4):578–589. [cited at p. 40]

Hess, W. (1983). *Pitch determination of speech signals: algorithms and devices*. Springer series in information sciences. Springer-Verlag. [cited at p. 25]

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA. [cited at p. 64, 65, 129, 132]

Holland, J. H. (1992a). *Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, USA. [cited at p. 130]

Holland, J. H. (1992b). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press. [cited at p. 75, 149]

Jančařík, A. (2007). *Algorithms and Solving Strategies*. Antonin Jancarik. [cited at p. 57]

Jensen, K. (1999). *Timbre models of musical sounds*. Københavns Universitet, Datalogisk Institut. [cited at p. 35]

Jordan, M. (2004). Graphical models. *Statistical Science*, pages 140–155. [cited at p. 41]

Jutten, C. and Herault, J. (1991). Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal processing*, 24(1):1–10. [cited at p. 46]

Kameoka, H., Nishimoto, T., and Sagayama, S. (2005a). Audio stream segregation of multi-pitch music signal based on time-space clustering using Gaussian Kernel 2-dimensional model. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05)*, volume 3. [cited at p. 52]

Kameoka, H., Nishimoto, T., and Sagayama, S. (2005b). Harmonic-temporal-structured clustering via deterministic annealing EM algorithm for audio feature extraction. In *Proc. ISMIR*, pages 115–122. [cited at p. 52]

Kameoka, H. K. H., Nishimoto, T. N. T., and Sagayama, S. S. S. (2007). A multipitch analyzer based on harmonic temporal structured clustering. [cited at p. 52, 172]

Keren, R., Zeevi, Y., and Chazan, D. (1998). Automatic transcription of polyphonic music using the multiresolution fourier transform. In *Electrotechnical Conference, 1998. MELECON 98., 9th Mediterranean*, volume 1, pages 654 –657 vol.1. [cited at p. 54]

Klapuri, A. (1998). Number theoretical means of resolving a mixture of several harmonic sounds. In *Proceedings of the European Signal Processing Conference*, page 400. [cited at p. 32]

Klapuri, A. (2001). Multipitch estimation and sound separation by the spectral smoothness principle. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, volume 5, pages 3381 –3384 vol.5. [cited at p. 41]

Klapuri, A. (2003). Multiple fundamental frequency estimation based on harmonicity and spectral smoothness. *IEEE Transactions on Speech and Audio Processing*, 11(6):804–816. [cited at p. 40, 42]

Klapuri, A. (2004a). *Signal processing methods for the automatic transcription of music*. PhD thesis, Tampere University of Technology Finland. [cited at p. xv, xvi, 4, 26, 29, 30, 36, 39, 55, 103]

Klapuri, A. (2005). A perceptually motivated multiple-f0 estimation method. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005*, pages 291–294. [cited at p. 41, 54]

Klapuri, A. (2006). Multiple fundamental frequency estimation by summing harmonic amplitudes. In *Proc. ISMIR*, pages 216–221. [cited at p. 41]

Klapuri, A. (2008). Multiplitch analysis of polyphonic music and speech signals using an auditory model. *IEEE Transactions on Audio and Language Processing*, 16(2):255–264. [cited at p. 41, 103]

Klapuri, A. P. (2004b). Automatic music transcription as we know it today. *Journal of New Music Research*, 33(3):269–282. [cited at p. 13]

Klapuri, A. P. and Astola, J. T. (2002). Efficient calculation of a physiologically-motivated representation for sound. In *Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on*, volume 2, pages 587–590. IEEE. [cited at p. 29]

Kobzantsev, A., Chazan, D., and Zeevi, Y. (2005). Automatic transcription of piano polyphonic music. In *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pages 414–418. IEEE. [cited at p. 54]

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems)*. A Bradford Book, 1 edition. [cited at p. 65]

Ladner, R. E. (1975). On the structure of polynomial time reducibility. *Journal of the ACM (JACM)*, 22(1):155–171. [cited at p. 61]

Lahat, M., Niederjohn, R., and Krubsack, D. (1987). A spectral autocorrelation method for measurement of the fundamental frequency of noise-corrupted speech. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(6):741–750. [cited at p. 28]

Landrin-Schweitzer, Y., Collet, P., and Lutton, E. (2006). Introducing lateral thinking in search engines. *Genetic Programming and Evolvable Machines*, 7(1):9–31. [cited at p. 131]

Large, E. W. and Kolen, J. (1994). Resonance and the perception of musical meter. *Connection science*, 6(2-3):177–208. [cited at p. 47]

Lea, A. (1970). *Auditory model of vowel perception*. PhD thesis, University of Nottingham. [cited at p. 40]

Lee, C.-T., Yang, Y.-H., Lin, K.-S., and Chen, H. (2010). Multiple fundamental frequency estimation of piano signals via sparse representation of fourier coefficients. In *Proceedings of the Fourth Music Information Retrieval Evaluation eXchange (MIREX 2010)*, Utrecht. [cited at p. 45, 172]

Lemon, S. M., Hamburg, M. A., Sparling, P. F., Choffnes, E. R., Mack, A., et al. (2007). *Global Infectious Disease Surveillance and Detection: Assessing the Challenges–Finding Solutions, Workshop Summary*. National Academies Press. [cited at p. 57]

Leveau, P., Vincent, E., Richard, G., and Daudet, L. (2008). Instrument-specific harmonic atoms for mid-level music representation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(1):116–128. [cited at p. 42]

Li, Y. and Wang, D. (2007). Pitch detection in polyphonic music using instrument tone models. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 2, pages II–481. IEEE. [cited at p. 52]

Licklider, J. (1951). A duplex theory of pitch perception. *Cellular and Molecular Life Sciences (CMLS)*, 7(4):128–134. [cited at p. 43]

Louchet, J., Guyon, M., Lesot, M., and Boumaza, A. (2002). Dynamic flies: a new pattern recognition tool applied to stereo sequence processing. *Pattern Recognition Letters*, 23(1-3):335–345. [cited at p. 131]

Loureiro, M., De Paula, H., and Yehia, H. (2004). Timbre classification of a single musical instrument. In *Proceedings of the 5th International Symposium on Music Information Retrieval (ISMIR 2004)*, pages 546–549. [cited at p. 35]

Lu, D. (2007). Automatic music transcription using genetic algorithms and electronic synthesis. Computer Science Undergraduate Research, University of Rochester, New York, USA. [cited at p. 85, 86, 88, 89, 90, 91, 92, 95]

Lyon, R. (1984). Computational models of neural auditory processing. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84.*, volume 9. [cited at p. 43]

Maher, R. (1989). *An approach for the separation of voices in composite musical signals.* PhD thesis, University of Illinois, IL, USA. [cited at p. 4]

Maher, R. (1990). Evaluation of a method for separating digitized duet signals. *Journal of the Audio Engineering Society*, 38(12):956–979. [cited at p. 49]

Maher, R. C. and Beauchamp, J. (1993). Fundamental frequency estimation of musical signals using a two-way mismatch procedure. *Journal of the Acoustical Society of America.* [cited at p. 43]

Mallat, S. and Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on*, 41(12):3397–3415. [cited at p. 42]

Marolt, M. (2002). *Automatic Transcription of Piano Music with Neural Networks.* PhD thesis, University of Ljubljana, Slovenia. [cited at p. 4]

Marolt, M. (2004a). A connectionist approach to automatic transcription of polyphonic piano music. *IEEE Transactions on Multimedia*, 6(3):439–449. [cited at p. 46, 48, 54, 154]

Marolt, M. (March 01, 2004b). Networks of adaptive oscillators for partial tracking and transcription of music recordings. *Journal of New Music Research*, 33:49–59(11). [cited at p. 46, 48]

Martin, K. D. (1996). A blackboard system for automatic transcription of simple polyphonic music. Technical report, Tech. Rep. 385, MIT Media Lab, Perceptual Computing Section. [cited at p. 53, 56]

Martin, P. (1981). Mesure de la frquence fondamentale par intercorrlation avec une fonction peigne. Technical report, JEP. [cited at p. 82]

Martin, P. (1982). Comparison of pitch detection by cepstrum and spectral comb analysis. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.*, volume 7, pages 180–183. IEEE. [cited at p. 29]

McIntyre, M., Schumacher, R., and Woodhouse, J. (1983). On the oscillations of musical instruments. *The Journal of the Acoustical Society of America*, 74:1325. [cited at p. 37]

McKay, C. (2003). Using blackboard systems for polyphonic transcription: A literature review. *Course Paper, McGill University.* [cited at p. 54]

Meddis, R. (1986). Simulation of mechanical to neural transduction in the auditory receptor. *The Journal of the Acoustical Society of America*, 79:702. [cited at p. 47]

Meddis, R. and Hewitt, M. J. (1991a). Virtual pitch and phase sensitivity of a computer model of the auditory periphery. i: Pitch identification. *The Journal of the Acoustical Society of America*, 89:2866. [cited at p. 29]

Meddis, R. and Hewitt, M. J. (1991b). Virtual pitch and phase sensitivity of a computer model of the auditory periphery. ii: Phase sensitivity. *The Journal of the Acoustical Society of America*, 89:2883. [cited at p. 29]

Mendel, G. and Bateson, W. (1925). *Experiments in plant-hybridisation /.* Cambridge, Mass. :Harvard University Press,. http://www.biodiversitylibrary.org/bibliography/4532. [cited at p. 62, 66]

Meng, X. and Rubin, D. (1993). Maximum likelihood estimation via the ecm algorithm: A general framework. *Biometrika*, 80(2):267–278. [cited at p. 52]

Michael, R. G. and Johnson, D. S. (1979). Computers and intractability: A guide to the theory of np-completeness. *WH Freeman & Co., San Francisco.* [cited at p. 61]

Michalewicz, Z. (1994). *Genetic algorithms + data structures = evolution programs (2nd, extended ed.).* Springer-Verlag New York, Inc., New York, NY, USA. [cited at p. 64]

Min, K., Chien, D., Li, S., and Jones, C. (1988). Automated two speaker separation system. In *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pages 537–540. [cited at p. 43]

Mitchell, M., Forrest, S., and Holland, J. H. (1992). The royal road for genetic algorithms: Fitness landscapes and ga performance. In *Proceedings of the First European Conference on Artificial Life*, pages 245–254. MIT Press. [cited at p. 132, 133, 134]

Mitchell, M., Holland, J. H., and Forrest, S. (1994). When will a genetic algorithm outperform hill climbing. In *Advances in Neural Information Processing Systems 6*, pages 51–58. Morgan Kaufmann. [cited at p. 133]

Mitianoudis, N. and Davies, M. (2002). Intelligent audio source separation using independent component analysis. In *Audio Engineering Society Convention 112*. Audio Engineering Society. [cited at p. 51]

Molla, S. and Torrésani, B. (2004). Determining local transientness of audio signals. *Signal Processing Letters, IEEE*, 11(7):625–628. [cited at p. 37]

Monti, G. and Sandler, M. (2002). Automatic polyphonic piano note extraction using fuzzy logic in a blackboard system. In *Proceedings of the 5th International Conference on Digital Audio Effects (DAFx-02)*, pages 26–28. [cited at p. 54]

Moorer, J. (1975). *On the segmentation and analysis of continuous musical sound by digital computer.* PhD thesis, Stanford University. [cited at p. 4]

Moorer, J. A. (1977). On the transcription of musical sound by computer. *Computer Music journal*, 1(4):32–38. [cited at p. 39]

Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826:1989. [cited at p. 66]

Nakano, M., Egashira, K., Ono, N., and Sagayama, S. (2009). Harmonic temporal structured clustering with unsupervised model learning for multipitch estimation. In *proceedings of the fifth music information retrieval evaluation exchange (mirex 2009)*, kobe. [cited at p. 172]

Nesi, P., Argenti, F., and Pantaleo, G. (2009a). Automatic transcription of real, polyphonic and multi-instrumental music based on constant-q bispectral analysis. Tech. report, Dep. of Systems and Informatics, University of Florence, Italy. [cited at p. 48]

Nesi, P., Pantaleo, G., and Argenti, F. (2009b). Automatic transcription of polyphonic music based on constant-Q bispectral analysis for mirex 2009. In *proceedings of the fifth music information retrieval evaluation exchange (mirex 2009)*, kobe. [cited at p. 172]

Nguyen, L. and Imai, S. (1977). Vocal pitch detection using generalized distance function associated with a voice-unvoice decision logic. *Bull. PME*, 39:11–21. [cited at p. 25]

Nikias, C. L. and Mendel, J. M. (1993). Signal processing with higher-order spectra. *Signal Processing Magazine IEEE*, 10(3):10–37. [cited at p. 48, 172]

Nikias, C. L. and Raghuveer, M. R. (1987). Bispectrum estimation: A digital signal processing framework. *Proceedings of the IEEE*, 75(7):869–891. [cited at p. 48, 172]

Noll, A. (1967). Cepstrum pitch determination. *The journal of the acoustical society of America*, 41(2):293–309. [cited at p. 26]

Nostrand, V. (1962). *Mathematics of Statistics*, volume 1 of *3*, chapter Harmonic Mean, pages 57–58. Princeton, NJ. [cited at p. 111, 152]

Nuno Fonseca, A. F. (2010). Measuring music transcription results based on a hybrid decay/sustain evaluation. *ESCOM 2009 - 7th Triennial Conference of European Society for the Cognitive Sciences of Music, Finland*. [cited at p. 158]

Ochoa, G., Lutton, E., and Burke, E. K. (2007). The cooperative royal road: Avoiding hitchhiking. In *Artificial Evolution*, pages 184–195. [cited at p. 130, 135, 139]

Olague, G. and Puente, C. (2006). Parisian evolution with honeybees for three-dimensional reconstruction. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 191–198. ACM New York, NY, USA. [cited at p. 131]

Oliveto, P. S. and Witt, C. (2012). On the analysis of the simple genetic algorithm. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO '12, pages 1341–1348, New York, NY, USA. ACM. [cited at p. 64, 79]

Oppenheim, A. and Schafer, R. (2004). From frequency to quefrency: a history of the cepstrum. *Signal Processing Magazine, IEEE*, 21(5):95–106. [cited at p. 26]

Oppenheim, A., Willsky, A., and Nawab, S. (1997). *Signals and systems*. Prentice-Hall signal processing series. Prentice Hall. [cited at p. 16]

Ortiz-Berenguer, L., Casajus-Quiros, F., and Torres-Guijarro, S. (2005). Multiple piano note identification using a spectral matching method with derived patterns. *Journal of the Audio Engineering Society*, 53(1/2):32–43. [cited at p. 40]

Parsons, T. W. (1976). Separation of speech from interfering speech by means of harmonic selection. *The Journal of the Acoustical Society of America*, 60(4):911–918. [cited at p. 33, 40]

Patterson, R. D. and Holdsworth, J. (1996). A functional model of neural activity patterns and auditory images. *Advances in speech, hearing and language processing*, 3(Part B):547–563. [cited at p. 47, 54]

Peeters, G. (2006). Music pitch representation by periodicity measures based on combined temporal and spectral representations. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 5, pages V–V. IEEE. [cited at p. 43]

Pertusa, A. (2010). *Computationally efficient methods for polyphonic music transcription*. PhD thesis, Universidad de Alicante. [cited at p. 4, 55]

Pertusa, A. and Inesta, J. (2008a). Multiple fundamental frequency estimation using gaussian smoothness. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 105 –108. [cited at p. 49, 50, 172]

Pertusa, A. and Inesta, J. (2008b). Multiple Fundamental Frequency Estimation using Gaussian Smoothness and Short Context. In *Proceedings of the Fourth Music Information Retrieval Evaluation eXchange (MIREX 2008)*, Philadelphia, PA, United States. [cited at p. 49, 172]

Piszczalski, M. and Galler, B. A. (1977). Automatic music transcription. *Computer Music Journal*, 1(4):24–31. [cited at p. 39]

Plante, F. (1995). A pitch extraction reference database. *Children*, 8(12):30–50. [cited at p. 41]

Plumbley, M., Abdallah, S., Bello, J., Davies, M., Monti, G., and Sandler, M. (2002). Automatic music transcription and audio source separation. *Cybernetics &Systems*, 33(6):603–627. [cited at p. 46, 51, 54]

Poliner, G. and Ellis, D. P. W. (2007a). Improving generalization for polyphonic piano transcription. In *Proc. IEEE Workshop on Apps. of Sig. Proc. to Audio and Acous.*, pages 86–89, Mohonk NY. [cited at p. 47, 171]

Poliner, G. E. and Ellis, D. P. W. (2007b). A discriminative model for polyphonic piano transcription. *EURASIP J. Appl. Signal Process.*, 2007(1):154–154. [cited at p. 47, 168, 171]

Rabiner, L., Cheng, M., Rosenberg, A., and McGonegal, C. (1976). A comparative performance study of several pitch detection algorithms. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 24(5):399–418. [cited at p. 26, 27]

Raczyński, S., Ono, N., and Sagayama, S. (2007). Multipitch analysis with harmonic nonnegative matrix approximation. In *in ISMIR 2007, 8th International Conference on Music Information Retrieval*. Citeseer. [cited at p. 46, 172]

Raczyński, S. and Sagayma, S. (2009). Multiple frequency estimation for piano recordings with concatenated regularized harmonic nmf. In *Proceedings of the Fifth Music Information Retrieval Evaluation Exchange (MIREX 2009)*, Kobe. [cited at p. 172]

Raiford, T. E. (1942). Skewness of combined distributions. *j-J-AM-STAT-ASSOC*, 37(219):391–393. [cited at p. 171]

Rechenberg, I. (1973). *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog. [cited at p. 64, 65]

Reeves, C. and Rowe, J. (2003). *Genetic algorithms : principles and perspectives ; a guide to GA theory*. Kluwer Acad. Publ. [cited at p. 132]

Reis, G. and Fernandez, F. (2007a). Electronic synthesis using genetic algorithms for automatic music transcription. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1959–1966, New York, NY, USA. ACM Press. [cited at p. 89, 90, 95]

Reis, G. and Fernandez, F. (2007b). Polyphonic music transcription by means of genetic algorithms. In *CEDI '07: Proceedings of the I Jornadas sobre Algoritmos Evolutivos y Metaheuristicas of the II Congreso Español de Informática*, Zaragoza, Spain. ACM Press. [cited at p. 92]

Reis, G., Fernandez, F., and Ferreira, A. (2009). Transcripcin de música multi-timbre mediante algoritmos genéticos. In *MAEB 2009 VI Congreso Espaol sobre Metaheursticas, Algoritmos Evolutivos y Bioinspirados*. [cited at p. 154]

Reis, G., Fernandéz, F., and Ferreira, A. (2012). Automatic transcription of polyphonic piano music using genetic algorithms, adaptive spectral envelope modeling and dynamic noise level estimation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(6):1643 –1654. [cited at p. 172, 174]

Reis, G., Fonseca, N., and Fernandez, F. (2007). Genetic algorithm approach to polyphonic music transcription. *Proceedings of WISP 2007 IEEE International Symposium on Intelligent Signal Processing*, pages 321–326. [cited at p. 88, 89, 95]

Reis, G., Fonseca, N., Fernandez, F., and Ferreira, A. (2008). A genetic algorithm approach with harmonic structure evolution for polyphonic music transcription. In *The 8th IEEE International Symnposium on Signal Processing and Information Technology*. [cited at p. 169]

Reyes, N., Candeas, P., Cañadas Quesada, F., and Carabias, J. (2009). New algorithm based on spectral distance maximization to deal with the overlapping partial problem in note–event detection. *Signal Processing*, 89(8):1653–1660. [cited at p. 42]

Röbel, A. (2003). Transient detection and preservation in the phase vocoder. In *Proc. Int. Computer Music Conference (ICMC)*, pages 247–250. [cited at p. 37]

Rodet, X. and Jaillet, F. (2001). Detection and modeling of fast attack transients. In *Proceedings of the International Computer Music Conference*, pages 30–33. [cited at p. 37]

Ross, M., Shaffer, H., Cohen, A., Freudberg, R., and Manley, H. (1974). Average magnitude difference function pitch extractor. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 22(5):353 – 362. [cited at p. 25]

Rudolph, G. (1998). Finite markov chain results in evolutionary computation: a tour d'horizon. *Fundam. Inf.*, 35(1-4):67–89. [cited at p. 64, 70, 79]

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition. [cited at p. 142]

Ryynänen, M. (2008). *Automatic Transcription of Pitch Content in Music and Selected Applications*. PhD thesis, Tampere University of Technology. [cited at p. 4]

Ryynänen, M. and Klapuri, A. (2005). Polyphonic music transcription using note event modeling. In *Proc. 2005 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 319–322, New Paltz, New York, USA. [cited at p. 41, 52, 154, 171]

Ryynänen, M. and Klapuri, A. (2008). Polyphonic Music Transcription using Note Event Modeling for MIREX 2008. In *Proceedings of the Fourth Music Information Retrieval Evaluation eXchange (MIREX 2008)*, Philadelphia, PA, United States. [cited at p. 171]

Ryynnen, M. P. and Klapuri, A. P. (2004). Modelling of note events for singing transcription. In *in Proc. ISCA Tutorial and Research Workshop on Statistical and Perceptual Audio*, page 6. MIT Press. [cited at p. 41]

Sagayama, S., Takahashi, K., Kameoka, H., and Nishimoto, T. (2004). Specmurt anasylis: A piano-roll-visualization of polyphonic music signal by deconvolution of log-frequency spectrum. In *ISCA Tutorial and Research Workshop (ITRW) on Statistical and Perceptual Audio Processing*. ISCA. [cited at p. 48, 54]

Saito, S., Kameoka, H., Takahashi, K., Nishimoto, T., and Sagayama, S. (2008). Specmurt analysis of polyphonic music signals. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(3):639–650. [cited at p. 48]

Schaffer, J. and Eshelman, L. (1991). On crossover as an evolutionarily viable strategy. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 61–68. Morgan Kaufmann Publishers. [cited at p. 133]

Schmidt, M. and Larsen, J. (2008). *Single-channel source separation using non-negative matrix factorization*. PhD thesis, Ph. D. dissertation, Technical University of Denmark. [cited at p. 46]

Schroeder, M. (1968). Period histogram and product spectrum: New methods for fundamental-frequency measurement. *The Journal of the Acoustical Society of America*, 43(4):829–834. [cited at p. 28]

Schwefel, H. (1994). On the evolution of evolutionary computation. In *Proceedings of 3rd International Conference of IEEE World Congress on Computer Intelligence, Orlando, Florida*, pages 116–124. Citeseer. [cited at p. 62, 65]

Schwefel, H. (1995). *Evolution and Optimum Seeking*. John Wiley & Sons. [cited at p. 49]

Schwefel, H. P. (1975). Evolutionsstrategie und numerische optimierung. *PhD Thesis*. [cited at p. 64, 65]

Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., New York, NY, USA. [cited at p. 64]

Sha, F. and Saul, L. (2004). Real-time pitch determination of one or more voices by nonnegative matrix factorization. *Departmental Papers (CIS)*, page 168. [cited at p. 46]

Shannon, C. E. (1949). Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21. [cited at p. 18]

Shields, V. C. (1970). *Separation of added speech signals by digital comb filtering*. PhD thesis, Massachusetts Institute of Technology. [cited at p. 39]

Slaney, M., Lyon, R., Inc, A., and Cupertino, C. (1990). A perceptual pitch detector. In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 357–360. [cited at p. 43]

Sluyter, R., Kotmans, H., and Claasen, T. (1982). Improvements of the harmonic-sieve pitch extraction scheme and an appropriate method for voiced-unvoiced detection. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.*, volume 7, pages 188–191. IEEE. [cited at p. 29]

Smaragdis, P. and Brown, J. (2003). Non-negative matrix factorization for polyphonic music transcription. In *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, pages 177–180. [cited at p. 46, 74, 171]

Smaragdis, P., Raj, B., and Shashanka, M. (2008). Sparse and shift-invariant feature extraction from non-negative data. *2008 IEEE International Conference on Acoustics Speech and Signal Processing*, pages 2069–2072. [cited at p. 45, 172]

Sreenivas, T. and Rao, P. (1981). Functional demarcation of pitch. *Signal Processing*, 3(3):277–284. [cited at p. 29]

Stewart, E. (2004). Intel integrated performance primitives: How to optimize software applications using intel ipp. [cited at p. 92]

Talkin, D. (1995). A robust algorithm for pitch tracking (RAPT). *Speech coding and synthesis*, 495:518. [cited at p. 26]

Taylor, S. (2007). *Optimizing applications for multi-core processors, using the Intel integrated performance primitives*. Intel Press. [cited at p. 92]

Tolonen, T. and Karjalainen, M. (2000). A computationally efficient multipitch analysis model. *IEEE Transactions on Speech and Audio Processing*, 8(6):708–716. [cited at p. 27, 43]

Vincent, E. (2004). *Modeles dinstruments pour la separation de sources et la transcription denregistrements musicaux*. Ph. D. dissertation, IRCAM, Paris, France, 2004. [cited at p. 4, 52]

Vincent, E., Bertin, N., and Badeau, R. (2007). Two nonnegative matrix factorization methods for polyphonic pitch transcription. In *2007 Music Information Retrieval Evaluation eXchange (MIREX)*, Vienna, Austria. [cited at p. 46, 171]

Vincent, E., Bertin, N., and Badeau, R. (2008). *Harmonic and inharmonic nonnegative matrix factorization for polyphonic pitch transcription*, pages 109–112. Number 1. IEEE. [cited at p. 154, 171]

Vincent, E., Bertin, N., and Badeau, R. (2010). Adaptive harmonic spectral decomposition for multiple pitch estimation. *Trans. Audio, Speech and Lang. Proc.*, 18(3):528–537. [cited at p. 154]

Vincent, E. and Rodet, X. (2004). Music transcription with isa and hmm. *Independent Component Analysis and Blind Signal Separation*, pages 1197–1204. [cited at p. 51]

Virtanen, T. (2003a). Algorithm for the separation of harmonic sounds with time-frequency smoothness constraint. In *Proc. Int. Conf. on Digital Audio Effects (DAFx)*, pages 35–40. [cited at p. 33]

Virtanen, T. (2003b). Sound source separation using sparse coding with temporal continuity objective. In *Proc. ICMC*, volume 3, pages 231–234. [cited at p. 46]

Virtanen, T. (2006). Unsupervised learning methods for source separation in monaural music signals. *Signal Processing Methods for Music Transcription*, pages 267–296. [cited at p. 46]

Virtanen, T. (2007). Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(3):1066–1074. [cited at p. 46]

Viste, H. and Evangelista, G. (2002). An extension for source separation techniques avoiding beats. In *Proc. Int. Conf. on Digital Audio Effects (DAFx)*, pages 71–75. Citeseer. [cited at p. 33]

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37(3):328–339. [cited at p. 47]

Walmsley, P., Godsill, S., and Rayner, P. (1999). Polyphonic pitch tracking using joint bayesian estimation of multiple frame parameters. [cited at p. 50]

Wei, B. and Gibson, J. (2000). Comparison of distance measures in discrete spectral modeling. In *Proc. 9th DSP Workshop & 1st Signal Processing Education Workshop*. Citeseer. [cited at p. 147]

Weintraub, M. (1986). A computational model for separating two simultaneous talkers. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86.*, volume 11. [cited at p. 43]

Wood, A. (2007). *The Physics of Music*. Read Books. [cited at p. 34]

Wu, J., Racczyński, A., Kitano, Y., and Nishimoto, T. (2010). Flexible harmonic model for multipitch estimation. In *Proceedings of the Fourth Music Information Retrieval Evaluation eXchange (MIREX)*, Utrecht. [cited at p. 172]

Wu, M., Wang, D., and Brown, G. (2003). A multipitch tracking algorithm for noisy speech. *IEEE Transactions on Speech and Audio Processing*, 11(3):229. [cited at p. 43, 54]

Yeh, C. (2008). *Multiple Fundamental Frequency Estimation of Polyphonic Recordings*. Thése de doctorat, University Paris 6 (UPMC), Paris. [cited at p. xvi, 4, 24, 31, 32, 33, 35, 36, 39, 48, 49, 54]

Yeh, C. and Röbel, A. (2006). Adaptive noise level estimation. In *Proc. of the 9th Int. Conf. on Digital Audio Effects (DAFx06)*, pages 145–148. [cited at p. 48]

Yeh, C., Robel, A., and Rodet, X. (2005). Multiple fundamental frequency estimation of polyphonic music signals. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, volume 3, pages iii/225 – iii/228 Vol. 3. [cited at p. 49, 171]

Yeh, C., Röbel, A., and Rodet, X. (2006). Multiple f0 tracking in solo recordings of monodic instruments. In *120th AES Convention*, pages 20–23. Citeseer. [cited at p. 37, 49]

Yeh, C. and Roebel, A. (2009). The expected amplitude of overlapping partials of harmonic sounds. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 3169–3172. IEEE. [cited at p. 33]

Yeh, C. and Roebel, A. (2010). Multiple-f0 estimation for mirex 2010. In *Proceedings of the Fourth Music Information Retrieval Evaluation eXchange (MIREX)*, Utrecht. [cited at p. 171]

Yeh, C., Roebel, A., and Rodet, X. (2010). Multiple fundamental frequency estimation and polyphony inference of polyphonic music signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 18:1116–1126. [cited at p. 30, 48, 145, 171]

Young, S., Russell, N., and Thornton, J. (1989). Token passing: a simple conceptual model for connected speech recognition systems. Technical report. [cited at p. 41]

Zhou, R. (2006). *Feature Extraction of Musical Content for Automatic Music Transcription*. PhD thesis, Ecole Polytechnique Fdrale de Lausanne, Swiss. [cited at p. 4, 43, 44, 48, 171]

Zhou, R. and Mattavelli, M. (2007). A new time-frequency representation for music signal analysis: Resonator time-frequency image. In *Signal Processing and Its Applications, 2007. ISSPA 2007. 9th International Symposium on*, pages 1 –4. [cited at p. 43, 171]

Zhou, R. and Reiss, J. D. (2008). A real-time polyphonic music transcription system. In *Proceedings of the Fourth Music Information Retrieval Evaluation eXchange (MIREX)*, pages 1–4, Philadelphia, USA. [cited at p. 171]

Appendices

Appendix A

Publications

The work presented in this Thesis is original work undertaken between June 2006 and June 2012 at the University of Extremadura, Spain. Portions of this work have been published elsewhere.

Journal Proceedings

- Reis, G. and Fernandéz, F. and Ferreira, Aníbal, "Audio Analysis and Synthesis-Automatic Transcription of Polyphonic Piano Music Using Genetic Algorithms, Adaptive Spectral Envelope Modeling, and Dynamic Noise Level Estimation". *IEEE Transactions on Audio Speech and LanguageProcessing* Volume 20, Issue 8, pages 2313–2328.
  DOI: `http://dx.doi.org/10.1109/TASL.2012.2201475`

Conference Proceedings

- Gustavo Reis, Francisco Fernandéz, and Aníbal Ferreira, "Evolutionary algorithms and automatic transcription of music". In *Proceedings of the fourteenth international conference on Genetic and Evolutionary Computation Conference Companion*, pages 477–484. Philadelphia, USA. ACM Press, 2012.

- Gustavo Reis, Francisco Fernandez, and Aníbal Ferreira, "Genetic algorithm approach to polyphonic music transcription for mirex 2008". *MIREX (2011), Multiple-F0 Estimation and Tracking Contest.*. Miami, USA. 2011.

- Gustavo Reis, Francisco Fernandéz, Gustavo Olague, "Cooperative and Decomposable Approaches on Royal Road Functions: Overcoming the Random Mutation Hill-Climber". In *GECCO 09: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pages 1875–1876. Montreal, Canada. ACM Press, 2009.

- Gustavo Reis, Francisco Fernandéz, Aníbal Ferreira, "Transcripción de Música Multi-Timbre mediante Algoritmos Genéticos". In *Proceedings of MAEB 2009 VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*, pages 167–174 . Malaga, Spain.

- Gustavo Reis, Nuno Fonseca, Francisco Fernandéz, Aníbal Ferreira, "A Genetic Algorithm Approach with Harmonic Structure Evolution for Polyphonic Music Transcription". In *Proceedings of ISSPIT 2008 IEEE International Symposium on Signal Processing and Information Technology (2008)*, pages 491–496. Sarajevo, Bosnia and Herzegovina. IEEE, 2008.

- Gustavo Reis, Francisco Fernandez, and Aníbal Ferreira, "Genetic algorithm approach to polyphonic music transcription for mirex 2008". *MIREX (2008), Multiple-F0 Estimation and Tracking Contest.*. Philadelphia, USA. 2008.

- Gustavo Reis, Nuno Fonseca, Francisco Fernández de Vega, and Aníbal Ferreira, "Hybrid Genetic Algorithm Based on Gene Fragment Competition for Polyphonic Music Transcription". In *EvoIASP 2008*, pages 305–314. Naples, Italy. Lecture Notes in Computer Science 4974/2008.

- Gustavo Reis, Francisco Fernandez, "Genetic Algorithms for Polyphonic Music Transcription". In EPIA'07: Proceedings of the 2007 Doctoral Symposium on Artificial Intelligence. Guimarães, Portugal. 2007

- Gustavo Reis, Nuno Fonseca, and F Ferndandéz, "Genetic Algorithm Approach to Polyphonic Music Transcription". In *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on*, pages 321–326. Alcalá de Henares, Spain. IEEE, 2007.

- Gustavo Miguel Jorge dos Reis, Francisco Fernandez, "Polyphonic Music Transcription by means of Genetic Algorithms". In *CEDI0'7: Proceedings of the I Jornadas sobre Algoritmos Evolutivos y Metaheuristicas of the II Congreso Español de Informática*. ISBN:978-84-9732-593-6. Zaragoza, Spain, September 2007.

- Gustavo Reis and Francisco Fernandéz, "A novel approach to automatic music transcription using electronic synthesis and genetic algorithms". In *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, pages 2915–2922. London, UK. ACM Press, 2007.

- Gustavo Reis and Francisco Fernandéz, "Electronic synthesis using genetic algorithms for automatic music transcription". In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1959–1966. London, UK. ACM Press, 2007.

Other papers published

- Lee Scott Reis and Gustavo Reis and João Barroso and António Pereira, "AMIGA-An Interactive Musical Environment for Gerontechnology". In *Journal of Procedia Computar Science*, pages 208–217. Elsevier, 2012.

- Tiago Francisco and Gustavo Miguel Jorge dos Reis, "Evolving Combat Algorithms to Control Space Ships in a 2D Space Simulation Game with Co-evolution using Genetic Programming and Decision Trees". In *Proceedings of the 2008 GECCO Conference Companion on Genetic and Evolutionary Computation*, pages 1887–1892. Atlanta, USA. ACM Press, 2008.

- Tiago Francisco and Gustavo Miguel Jorge dos Reis, "Evolving Predator and Prey Behaviours with Co-evolution Using Genetic Programming and Decision Trees". In *Proceedings of the 2008 GECCO Conference Companion on Genetic and Evolutionary Computation*, pages 1893–1900. Atlanta, USA. ACM Press, 2008.

Papers referring our work

- Mert Bay, Andreas F Ehmann, and J Stephen Downie. Evaluation of multiple-f0 estimation and tracking systems. In *10th International Society for Music Information Retrieval Conference*, pages 315–320, 2009.

- Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: Breaking the glass ceiling. 2012.

- JJ Carabias-Orti, FJ Rodriguez-Serrano, P Vera-Candeas, FJ Cañadas-Quesada, and N Ruiz-Reyes. Constrained non-negative sparse coding using learnt instrument templates for realtime music transcription. *Engineering Applications of Artificial Intelligence*, 2013.

- Julio César Carvajal Ramírez and Fabián Andrés Giraldo Giraldo. Composing music through genetic algorithms. *Tecnura*, 16(33):145–157, 2012.

- Zhiyao Duan, Bryan Pardo, and Changshui Zhang. Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(8):2121–2133, 2010.

- Nuno Fonseca. Singing voice resynthesis using concatenative-based techniques. 2011.

- Nuno Fonseca and Aníbal Ferreira. Measuring music transcription results based on a hybrid decay/sustain evaluation. 2009.

- Nuno Fonseca and Ana Paula Rocha. Fragmentation and frontier evolution for genetic algorithms optimization in music transcription. In *Advances in Artificial Intelligence–IBERAMIA 2008*, pages 442–451. Springer, 2008.

- Peter Grosche, Bjorn Schuller, Meinard Muller, and Gerhard Rigoll. Automatic transcription of recorded music. *Acta Acustica united with Acustica*, 98(2):199–215, 2012.

- Herve Kabamba Mbikayi. Toward evolution strategies application in automatic polyphonic music transcription using electronic synthesis. *arXiv preprint arXiv:1304.0969*, 2013.

- Antonio Pertusa and José M Iñesta. Efficient methods for joint estimation of multiple fundamental frequencies in music signals. *EURASIP Journal on Advances in Signal Processing*, 2012(1):1–13, 2012.

- Antonio Pertusa-Ibáñez. *Computationally efficient methods for polyphonic music transcription*. PhD thesis, Universidad de Alicante, 2010.

- Joseph Scarr and Richard Green. Retrieval of guitarist fingering information using computer vision. In *Image and Vision Computing New Zealand (IVCNZ), 2010 25th International Conference of*, pages 1–7. IEEE, 2010.

- Suyog Sonwalkar and Itthi Chatnuntawech. Autoscore: The automated music transcriber project proposal 18-551, spring 2011 group. 2011.

- Pat Taweewat and Chai Wutiwiwatchai. Musical pitch estimation using a supervised single hidden layer feed-forward neural network. *Expert Systems with Applications*, 2012.

- Mahdi Triki and Dirk TM Slock. Perceptually motivated quasi-periodic signal selection for polyphonic music transcription. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 305–308. IEEE, 2009.

- Mahdi Triki, Dirk TM Slock, and Ahmed Triki. Periodic signal extraction with frequency-selective amplitude modulation and global time-warping for music signal decomposition. In *Multimedia Signal Processing, 2008 IEEE 10th Workshop on*, pages 972–977. IEEE, 2008.

- Yasunori Uchida and Shigeo Wada. Melody and bass line estimation method using audio feature database. In *Signal Processing, Communications and Computing (ICSPCC), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011.

- Sonmaz Zehtabi. *Vibraphone Transcription from Noisy Audio Using Factorization Methods*. PhD thesis, University of Victoria, 2012.

Appendix B

Rendering an Individual into an Audio Signal

We considered a dynamic range of 16dB, that is: one note can vary its dynamic between 1 and 127. In particular, 127 MIDI velocity value corresponds to +8dB gain and 1 MIDI velocity corresponds to −8dB gain and 64 MIDI velocity corresponds to 0 dB gain. The gain Equation, according to each note dynamic, is given by:

$$gain = 10^{\frac{vel-64}{80}}. \tag{B.1}$$

After each note offset the following release equation is applied:

$$release(t) = \frac{2000 - \frac{t}{36}}{2000 + t} \tag{B.2}$$

where t varies from $t = 0 \ldots 72000$.

Appendix C

Proof of Octave Normalization

Proof. The following equation gives the MIDI note for a given frequency:

$$Note(freq) = 69 + 12\log_2\left(\frac{freq}{440}\right). \tag{C.1}$$

For sound using Fs sampling rate and using a window size of N, the frequency resolution of the STFT is $\frac{Fs}{N}$. Thus the center frequency of each bin in the STFT is given by:

$$Freq(k) = k \times \frac{Fs}{N}. \tag{C.2}$$

Considering previous Equations, the MIDI note corresponding to the center frequency of each bin is given by $Note(Frequency(k))$. Therefore, the MIDI notes elapsed between the center frequency of bin k and bin $k+1$ are given by $Note(Freq(k+1)) - Note(Freq(k))$ - Equation C.3.

This equation shows that the number of notes between the center frequency of the first bin and the second is 12 (an octave), which is correct because the frequency doubles from the first bin to the second: $12\log_2(2) - 12\log_2(1) = 12$ notes.

$$
\begin{aligned}
Note(Freq(k+1)) - Note(Freq(k)) &= Note\left((k+1)\times\frac{Fs}{N}\right) - Note\left(k\times\frac{Fs}{N}\right) \\
&= 69 + 12\log_2\left(\frac{(k+1)\times\frac{Fs}{N}}{440}\right) - \left[69 + 12\log_2\left(\frac{k\times\frac{Fs}{N}}{440}\right)\right] \\
&= 69 + 12\log_2\left(\frac{(k+1)\times\frac{Fs}{N}}{440}\right) - 69 - 12\log_2\left(\frac{k\times\frac{Fs}{N}}{440}\right) \\
&= 12\log_2\left((k+1)\times\frac{Fs}{N}\right) - 12\log_2(440) - 12\log_2\left(k\times\frac{Fs}{N}\right) + 12\log_2(440) \\
&= 12\log_2(k+1) + 12\log_2\left(\frac{Fs}{N}\right) - 12\log_2(k) - 12\log_2\left(\frac{Fs}{N}\right) \\
&= 12\log_2(k+1) - 12\log_2(k) \qquad \text{(C.3)}
\end{aligned}
$$

To have an octave normalization with weight = 1 per octave, the previous equation (Equation C.3) should be divided by 12, resulting on the normalization $\log_2(k+1)-\log_2(k)$ applied on Equation 9.4. This way: $\log_2(2) - \log_2(1) = \log_2(4) - \log_2(2) = \log_2(8) - \log_2(4) = 1$ octave.

□

Appendix D

Get Possible Notes

The function GetPossibleNotes generates a list of possible notes and is described on Algorithm D.1. This function analyzes the power spectra of the acoustic signal and then returns the list of possible notes: for each frame n the biggest α peaks are selected, and then, for each peak, the corresponding MIDI notes are added to the possible notes list. α was empirically set to 10. The MIDI notes corresponding to frequency bin *bin* are those which verify the following equation:

$$bin = \text{Round}\left(\frac{freq_{note}}{resolution}\right) \tag{D.1}$$

where the $freq_{note}$ is the frequency of a MIDI note:

$$freq_{note} = 6.875 \times 2^{\frac{3+note}{12}} \tag{D.2}$$

and where the *resolution* is the frequency bin resolution:

$$resolution = \frac{Fs}{N}. \tag{D.3}$$

Algorithm D.1: Get Possible Notes algorithm.

Require: spectrum X, hop size R

for each frame $X(n)$ **do**
  eliminate all non-peak values from current frame
  for each peak p in $X(n)$ **do** {p is a 2-order tuple *(magnitude, bin)*}
    $P \leftarrow P + p$ {P is the list of peaks}
  end for
  sort P according to the power magnitude
  $P \leftarrow$ first α elements of P
  for each peak p in P **do**
    $beginNote \leftarrow$ first MIDI note belonging to bin $p.bin$
    $endNote \leftarrow$ first MIDI note belonging to bin $p.bin + 1$
    for $i = beginNote$ to $endNote - 1$ **do**
      $newNote.note = i$
      $newNote.start = n \times R$
      $newNote.duration = R$
      for each *note* in $possibleNotes$ **do**
        if *note* overlaps with $newNote$ **then**
          $note.duration \leftarrow note.duration + R$
        else
          $possibleNotes \leftarrow possibleNotes + newNote$
        end if
      end for
    end for
  end for
end for
return $possibleNotes$