applied sciences

MDPI

Article

Evolving a Multi-Classifier System for Multi-Pitch Estimation of Piano Music and Beyond: An Application of Cartesian Genetic Programming

Rolando Miragaia [1,*], Francisco Fernández [2], Gustavo Reis [1] and Tiago Inácio [3]

[1] Computer Science and Communication Research Centre, School of Technology and Management, Polytechnic of Leiria, Campus 2, Morro do Lena-Alto do Vieiro, Apartado 4163, 2411-901 Leiria, Portugal; gustavo.reis@ipleiria.pt
[2] Department of Computers and Communications Technology, University of Extremadura, 06800 Mérida, Spain; fcofdez@unex.es
[3] Farfetch, 1350-352 Lisbon, Portugal; tiagojjoao@gmail.com
* Correspondence: rolando.miragaia@ipleiria.pt; Tel.: +351-919-048-842

Abstract: This paper presents a new method with a set of desirable properties for multi-pitch estimation of piano recordings. We propose a framework based on a set of classifiers to analyze audio input and to identify piano notes present in a given audio signal. Our system's classifiers are evolved using Cartesian genetic programming: we take advantage of Cartesian genetic programming to evolve a set of mathematical functions that act as independent classifiers for piano notes. Two significant improvements are described: the use of a harmonic mask for better fitness values and a data augmentation process for improving the training stage. The proposed approach achieves competitive results using F-measure metrics when compared to state-of-the-art algorithms. Then, we go beyond piano and show how it can be directly applied to other musical instruments, achieving even better results. Our system's architecture is also described to show the feasibility of its parallelization and its implementation as a real-time system. Our methodology is also a white-box optimization approach that allows for clear analysis of the solutions found and for researchers to learn and test improvements based on the new findings.

Keywords: multi-pitch estimation; multiple-F0 estimation; evolutionary computing; Cartesian genetic programming; genetic programming; music transcription; machine learning

check for updates

Citation: Miragaia, R.; Fernández, F.; Reis, G.; Inácio, T. Evolving a Multi-Classifier System for Multi-Pitch Estimation of Piano Music and Beyond: An Application of Cartesian Genetic Programming. *Appl. Sci.* **2021**, *11*, 2902. https://doi.org/10.3390/app11072902

Academic Editor: Federico Divina

Received: 8 March 2021
Accepted: 21 March 2021
Published: 24 March 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Multi-pitch estimation or multiple fundamental frequency estimation is the process of extracting musical notation (pitches) from a given acoustic signal. Multi-pitch estimation is a task that belongs to content-based music information retrieval. Music information retrieval (MIR) has drawn more attention due to the exponential growth of digital music [1]. Howbeit, there is a significant gap between high-level human perception and low-level signal features. Some music features such as rhythm and pitch have an important role in helping bridge this gap since they are more closely related to the human perception of music. Multiple fundamental frequency estimation was introduced by Shields [2] in his research on separating co-channel speech signals. Later, multiple-F0 estimation was extended to polyphonic pitch estimation or multi-pitch estimation (MPE) in the context of automatic music transcription (AMT) for polyphonic music signals [3,4]. Among all musical instruments, the piano is one of the most popular instruments worldwide and one of the most complex in what concerns pitch variety and number of simultaneous notes [5]. These are the main reasons that motivate us to research the multi-pitch estimation of piano sounds.

A single piano note is a monophonic sound comprising a quasi-harmonic spectrum [6], where the lowest frequency of the harmonic series corresponds with the perceived pitch

or fundamental frequency (F0). The combination of multiple piano notes (monophonic sounds) results in a polyphonic sound with multiple pitches. MPE refers to determination of the underlying pitches of an obtained polyphonic sound. Unlike mono-pitch estimation, multi-pitch estimation deals with issues such as the source number ambiguity and the octave ambiguity (Figure 1) (Figure extracted from [7], with the author's consent). Therefore, multi-pitch estimation is a challenging problem. Although there has been a lot of research devoted to it, it still remains completely unsolved.

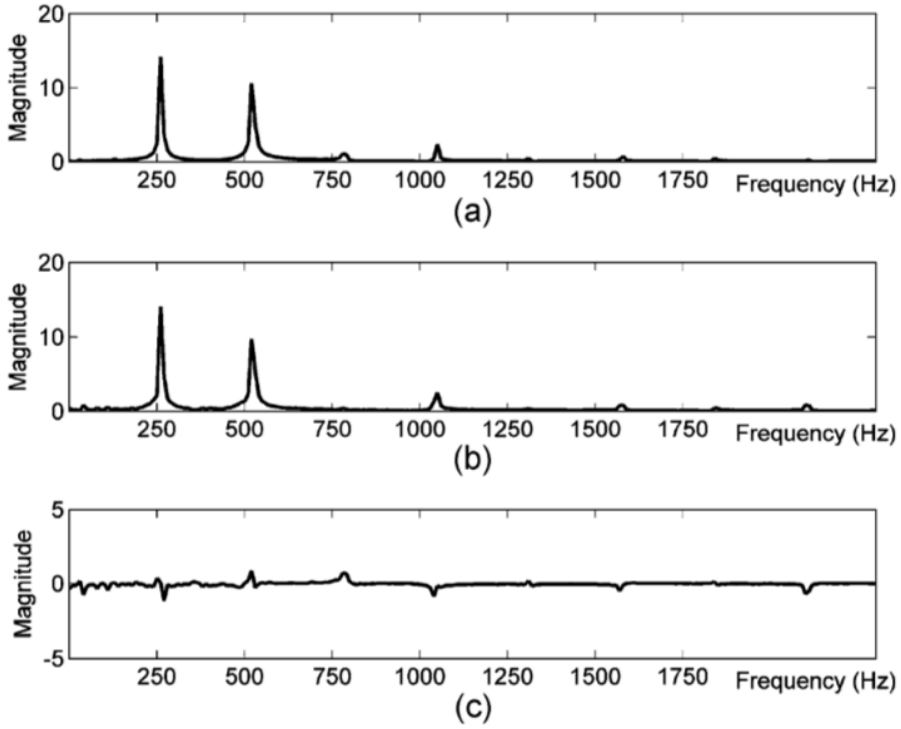

Figure 1. Example of source number and octave ambiguity - The spectrum of a single piano note C4, Musical Instrument Digital Interface (MIDI) note number 60, is almost identical to the spectrum of the two-key combination of C4 and C5. C5 is one octave apart from C4. (**a**) Spectrum of C4. (**b**) Spectrum of the mixture of C4 and C5. (**c**) Signal difference between (**a**,**b**).

In general, multi-pitch estimation approaches and algorithms assume that there might be more than one harmonic source in the same short-time signal. As explained by Yeh et al. [8], that signal may also be expressed as a sum of harmonic sources plus a residual (The residual, $z[n]$, comes from components that are not explained by sinusoids, for instance, the background noise, spurious components, or nonharmonic partials.):

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

where M is the number of harmonic sources, n represents the discrete time, $y_m[n]$ is the quasi-periodic part of the mth source signal, N_m represents the period of the mth source, and $z[n]$ is the residual. According to Fourier series and the signal quasi periodicity, $y[n]$ can be written as follows:

$$\begin{aligned} y[n] &= \sum_{m=1}^{M} \left\{ \sum_{h=1}^{\infty} A_{m,h} \cos(h\omega_m n + \phi_{m,h}) \right\} + z[n] \\ &\approx \sum_{m=1}^{M} \sum_{h=1}^{H_m} A_{m,h} \cos(h\omega_m n + \phi_{m,h}) + z[n]. \end{aligned} \tag{2}$$

Last-step approximation is common for practical usage: a finite and small number of sinusoids H is commonly used to approximate a quasi-periodic signal.

The main difficulty associated with MPE is dealing with the modeling of $y[n]$. This task implies estimating the number of harmonic sources and their related F0s. Decomposition of the observed signal into an unknown number of model sources not only is a problem of pattern matching but also can be seen as a classification problem, i.e., identifying the most likely combination of F0s for the modeling of $y[n]$.

Evolutionary algorithms are very successful in solving both pattern matching and classification problems [9]. This led us to propose a multi-pitch estimation system based on an evolutionary approach: genetic programming (GP). Among all GP variants available, we decided to innovate by addressing this problem using Cartesian genetic programming (CGP). CGP was first presented by Miller in [10] and has been proven to be able to classify problems related to signal and image processing [11].

1.1. Previous Approaches to Multi-Pitch Estimation

There are detailed reviews of the scientific literature on MPE [12–14], and according to [12], those works on MPE can be classified in three different groups, depending on the main technique used: feature-based approaches focus on devising measures of pitch salience and criteria for selecting and scoring pitch candidates from time-frequency representations [8,15–22] and most recently, in 2020, using two-dimensional spectrum [23]; *statistical model-based* approaches use probabilistic methods to model the spectral envelope or peaks and then formulates MPE as a maximum a posteriori or maximum likelihood problem [24–31]; and *spectrogram factorization-based* approaches use templates of spectral patterns of different pitch combinations and then decomposes an input magnitude spectrogram according to the activation of different templates [7,32–37]. Another common categorization of MPE techniques is based on whether there is usage of a training dataset with ground-truth pitches. Early research was usually classified as *unsupervised*; however, today, there is an increase in work based on *supervised* methods. Among them, there are neural networks and some evolutionary algorithms (EAs). Neural networks were used with salient results by Marolt [38], where he presents a connectionist approach to the transcription of piano music using a partial tracking technique. More recently, an end-to-end deep learning model for polyphonic piano music transcription was developed by Sigtia et al. [39], and Böck [40] used recurrent neuronal networks (RNNs) for multi-pitch estimation and onset detection.

Evolutionary Algorithms and Genetic Programming in Multi-Pitch Estimation

There are several approaches to the MPE problem using evolutionary algorithms. Genetic algorithms (GA) were probably the first evolutionary approach to multi-pitch estimation, as described in 2001 by Garcia [41]. Garcia proposes that polyphonic pitch detection can be considered a search space problem where the goal is to find pitches that compose the polyphonic acoustic signal. In 2007, Lu [42] proposed an automatic music transcription system based on genetic algorithms: this approach assumes that a polyphonic audio signal can only be produced by the 128 possible pitches (from low C, frequency 8.18 Hz, to a high G, 12,543.88 Hz) defined in the MIDI. specification [43]. In 2007, Reis et al. [44] proposed the first genetic algorithm approach to music transcription using real audio recordings. The authors explored the influence of the "harmonic overfitting" phenomena, and in 2012, Reis et al. [45] presented a new genetic algorithm to perform automatic transcription of polyphonic piano music with spectral envelope modeling and noise level estimation.

Genetic programming (GP) is also a subset of evolutionary algorithms. However, GP has some important differences when compared to genetic algorithms (GAs). Genetic programming applies the evolutionary process to a population of computer programs. The aim of the optimization is to evolve a program to produce a behavior as close as possible to some desired goal. Regarding MPE, the goal is multi-pitch estimation of piano music.

As far as we know, there is no record of any work in this area using any form of GP until 2016, when we first proposed it in [46]. In that work, an innovative methodology for pitch estimation of piano music using a genetic programming-based system is presented. This was our first approach to dealing with the MPE problem and a specific type of GP called Cartesian genetic programming was used [47]. We used a divide-and-conquer method, evolving one classifier for each piano key and its corresponding pitch (a total of 61 classifiers) that can run in parallel. Together, those classifiers acted as a multi-pitch estimator system. The CGP system was built using a previously developed toolbox for MatLab called CGP4Matlab [48] (Download available: https://github.com/tiagoinacio/cgp4matlab, accessed on 21 March 2021), which is particularly suited for signal processing and image processing problems and is available for general use. That technique was then improved using a CGP multi-pitch estimation system with the aid of a spectral harmonic mask [49], resulting in better accuracy results. However, there is still room for improvement, and this led us to what we present in this paper.

We propose an improved version of the algorithm, with some innovations and an extended MPE system based on Cartesian genetic programming for piano music. This system was developed for an AMT frame-based usage and takes advantage of the authors' knowledge previously acquired during other research [46,48]. Our system is the first white-box MPE-based approach that uses GP, in particular Cartesian genetic programming. The main innovations of this new CGP system are the use of a new onset detector, real-time computing performance using a parallel system architecture, and an artificial data augmentation process for the training stage using the onset detector, which resulted in accuracy improvements. We can summarize the main advantages of our approach and implementation as follows:

- The accuracy results are in line with the state-of-the-art approaches that use other non-evolutionary methodologies.
- It works with different piano models (upright and grand pianos) and with both digital and acoustic pianos.
- It does not depend on harmony rules, western music rules, or chords construction rules, which was typically the case for previous approaches.
- It uses improved onset detection.
- It provides white-box analyses, which allow other researchers to study the generated models and to use them as an optimization baseline.
- It achieves real-time computing performance.
- The technique allows us to perform MPE for other instruments besides piano.
- The significant accuracy results improved compared to our previous approach.

The rest of this document is structured as follows: Section 2 overviews the proposed system; Section 3 describes the implemented onset detection algorithm; Section 4 describes Cartesian genetic programming as a genetic algorithm technique; Section 5 describes our CGP system and its main features; Section 6 shows our experiments and results; and, finally, Section 7 presents our conclusions and discusses future work.

2. System Overview

Genetic programming is aimed at creating computer programs capable of solving a given problem. Cartesian genetic programming builds programs in the form of graphs. To address the MPE problem, our CGP system generates programs in the form of graphs using a set mathematical functions (function set). This way, each each individual will be an evolved graph encoding a complex mathematical expression.

In our system, each piano note is identified by a CGP evolved classifier (Figure 2). This way, for identifying 61 musical notes (from the C2 to the C7), we need to have 61 evolved classifiers, one for each musical note or piano key. Each one of these classifiers uses several inputs, all of them derived from the acquired audio signal and returning one binary output, indicating whether the corresponding piano note is present in the given signal. Basically, a sound vector is sampled using the onset detector proposed by Martins [50] with some

improvements. Then, five inputs are computed from the original sound vector using several signal processing techniques. These inputs are then used by the classifier CGP functions ($F_n(I)$) to accomplish an output vector. This vector suffers a binarization process to obtain a final binary output. The training process of the system classifiers is explained in Section 5.1.

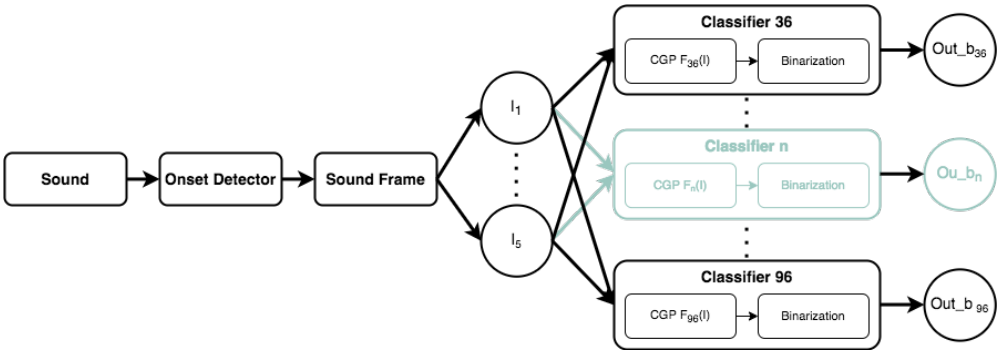

Figure 2. Block diagram of our multi-pitch estimation (MPE) system with multiple classifiers: first, an onset detector is applied to the input audio signal to infer where are the musical notes; then, an audio frame with 4096 audio samples (93 milliseconds) is extracted, starting at the onset time; the extracted audio fragment is then transformed to 5 different representations in the frequency domain, resulting in 5 different inputs ($I_1 \ldots I_5$); and these 5 inputs are fed into the 61 evolved classifiers, so that they can identify whether the corresponding musical notes are present in the extracted audio frame. Each of the 61 classifiers has a binary output.

The source code of our proposal is available for download at https://github.com/rolandomiragaia/CGP-MPE-MC, accessed on 21 March 2021. It is based on our CGP4MatLab toolbox, detailed in [48], as well as a generic implementation of our CGP multi-pitch estimation system.

3. Onset Detection

The onset detection algorithm is based on the onset detection algorithm used by Martins [50], with some modifications and improvements. The approach used is based on spectral flux as the onset detection function, defined as follows:

$$SF[n] = \sum_{k=0}^{\frac{N}{2}} H(|X[n,k]| - |X[n-1,k]|) \quad (3)$$

where $H(x) = \frac{x+|x|}{2}$ is the half wave rectifier function, $X(n,k)$ represents the kth bin of the n^{th} frame of short time Fourier transform (STFT) of the input $x[n]$. Linear magnitude is used instead of logarithmic. N is the Hamming window size. The experiments performed use a 46 ms frame size (i.e., $N = 2048$ samples, with sampling rate $f_s = 44,100$ Hz) and a 11.6 ms hop size ($h = 512$ samples).

As in [50], in order to reduce the false-positive rate, the onset detection function $SF(n)$ is smoothed using a Butterworth filter defined by $H(z) = \frac{0.1173+0.2347z^{-1}+0.1174z^{-2}}{1-0.8252z^{-1}+0.2946^{-2}}$. To avoid phase distortion (which would deviate the time of the detected onsets), the input data are filtered in both the forward and backward directions. The result has precisely zero phase distortion, with the magnitude being the square magnitude of the filter response and with the order of the filter being double the order specified by $H(z)$. We also used an envelope function $EH[p]$, where p represents the discrete time variable of the input signal $x[p]$ (The input signal is in time domain $x[n]$, represented as $x[p]$, because the variable n now represents the frame number and not the discrete time.) and the envelope function is calculated over the input time signal $x[p]$, using Hilbert transformation for discrete time signals [51].

The onsets are detected using a peak-picking algorithm to find a local maximum. A peak at instant $t = \frac{nH}{f_s}$ is chosen as an onset if the following conditions are met:

1. $SF[n] \geq SF[k] \; \forall \; k : n - w \leq k \leq n + m$
2. $SF[n] > \frac{\sum_{k=n-mw}^{n+w} SF[k]}{mw+w+1} \times thres + \delta$
3. $\frac{\sum_{k=n*h+1}^{n*h+N} EH[k]}{N} > \theta$

where $w = 6$ is the window size to achieve the local maxima, $m = 4$ is a multiplier so that the average should be calculated in a broader area before the peak, $thres = 2.0$ is a threshold value relative to the local average that a peak must reach in order to be sufficiently prominent to be selected as an onset, and $\delta = 10^{-20}$ is a residual that is combined with $\theta = 0.01$ to avoid false-positive detection in silent regions of the signal. All these parameters were adjusted empirically on previously performed tests using a collection of several piano compositions played by different pianos. Figure 3 shows an example of a sound sample of 2.1 seconds with 3 onsets detected for frames 29, 90, and 141.

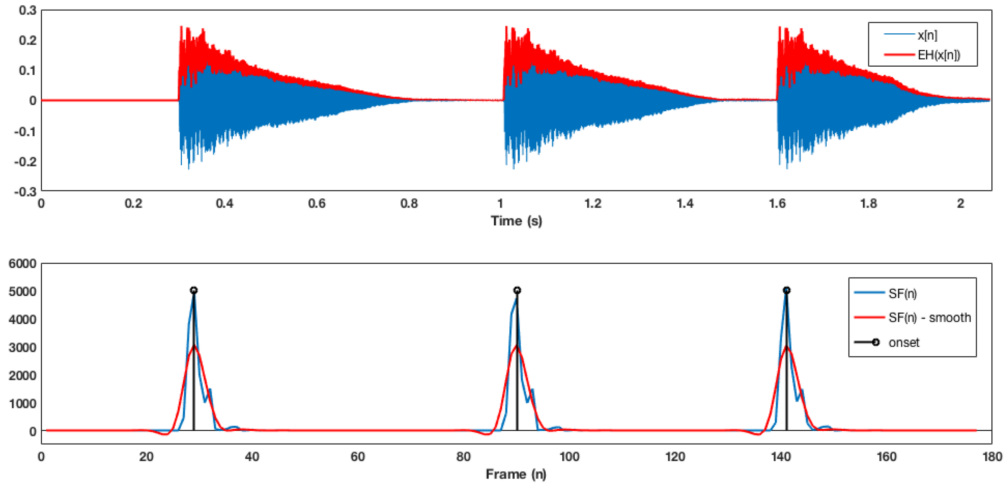

Figure 3. Onset detection process—(**above**) input signal in time domain $x[n]$ and its Hilbert envelope $EH[n]$; (**bellow**) the spectral flux $SF[n]$, where n represents the frame number and the spectral flux smoothed, and in black are the onsets detected and marked.

Onset detection is typically used for audio segmentation: since onset detectors infer where musical notes are present in the input audio signal, the input signal can be split into several audio fragments according to the onset information. This is a fundamental process in automatic music transcription algorithms that are based on note tracking. Despite our approach being frame-based, we also use our onset detector during the training stage for data augmentation purposes, hence improving the training process. The data augmentation process using the onset detector is explained in Section 5.2.

4. Cartessian Genetic Programming

Cartesian genetic programming is an increasingly popular and efficient form of genetic programming [52,53] proposed by Julian Miller in 2000 [47]. It is seen as a flexible form of genetic programming that is concerned with the automatic evolution of computational structures, such as computer programs, or mathematical equations or functions. CGP encodes a graphical representation of computer programs. The computational structures encoded are represented as a string of integers (sometimes real values). These integers, known as genes, determine the functions of nodes in the graph, the connections between nodes, the connections to inputs, and the locations in the graph where outputs are taken from. Therefore, the genotype consists of a list of integers (and possibly real parameters) that represent the program primitives and how they are connected together (Figure 4).

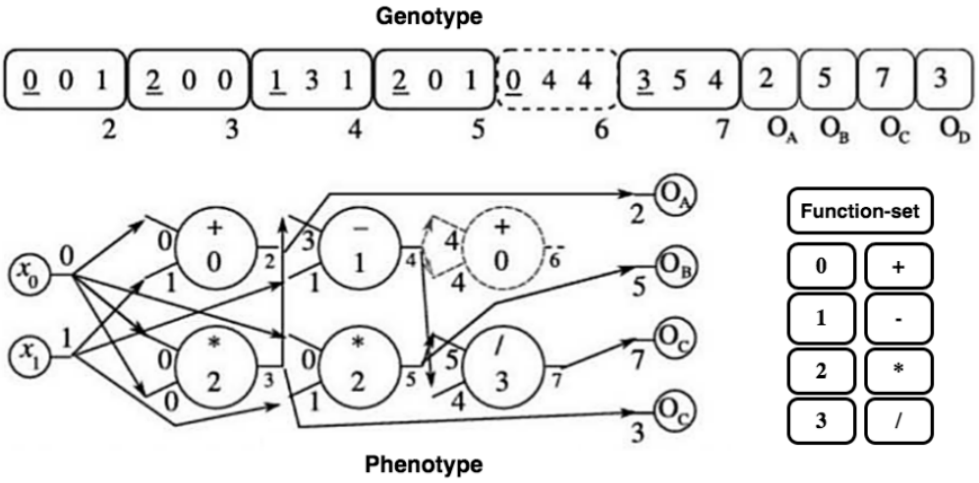

Figure 4. A Cartesian genetic programming (CGP) generic example of a genotype and its corresponding phenotype. There is a grid of nodes connected as a graph in which the functions are chosen from a set of primitive functions, (the function set). There are 2 inputs (x_0 and x_1) and 4 outputs (O_A, O_B, O_C, and O_D). The grid has $n_c = 3$ columns and $n_r = 2$ rows. Each node has 3 genes: the underlined is the function gene, which corresponds to the mathematical function being used, and the other 2 genes are the connection genes or node inputs. These inputs can refer to other nodes or system inputs.

CGP is called "Cartesian" because it considers a grid of nodes that are addressed in a Cartesian coordinate system. Each node may contain additional genes for encoding additional parameters that might be necessary for specific functions, for instance, a threshold value. Typically, all functions have as many inputs as the maximum function arity and unused connections are ignored. As in any evolutionary algorithm, each individual encodes a possible solution to the problem addressed. The set of individuals is called the population. To evaluate the quality of each individual or solution, a fitness function is used. This way, all the solutions among the population are evaluated and the current best solution is found. The next population (next generation) is then generated based on the current best individual. The next generation contains a new set of possible solutions and becomes the current population. This process is repeated over and over, from iteration (generation) to iteration. This way, the quality of the population improves (evolves) from generation to generation, pursuing the best solution to the problem. The general CGP is described in Algorithm 1: it begins with the generation of an initial population; then, a fitness function is used to evaluate the quality of each individual in the population; the evolutionary strategy chooses the fittest one (best individual) and promotes it directly to the next generation; the remaining spots in the population are filled with mutated versions of the fittest individual; and the algorithm stops when the stopping criterion is reached.

Algorithm 1 General CGP algorithm.

1: Generate initial population at random (subject to constraints)
2: **while** stopping criterion not reached **do**
3: Evaluate fitness of genotypes in population
4: Promote fittest genotype to new population
5: Fill remaining places in the population with mutated versions of the fittest
6: Return to step 2 until stopping criterion reached
7: **end while**

5. Proposed CGP System

A CGP-encoded individual, in its general form, consists of a grid of nodes, where each node has connections and a function chosen from a set of primitive functions. This grid of nodes has three main properties: n_c (columns), n_r (rows), and *levels-back* (how many previous columns of cells have their outputs connected to a node in the current column). Depending on n_r, n_c, and *levels-back*, a wide range of graphs can be generated. When $n_r = 1$ and *levels-back* $= n_c$, arbitrarily directed graphs can be created with a maximum depth. Choosing these values imposes the least constraints. Therefore, this is the best and most general choice [54].

In our system, each piano note is identified by a classifier previously evolved using CGP (training stage). Therefore, for identifying the musical notes between C2 (65 Hz) and C7 (2093 Hz), we need to evolve 61 classifiers: one for each musical note or piano key. Each classifier has several inputs, all of them extracted from the input audio signal and returning one binary output, indicating whether the corresponding piano note is present in the given signal. Basically, there are two different stages in our system: the training stage and the test or working stage. Both stages have processes in common, and they share the majority of the blocks illustrated in Figure 5, which depicts the training process. The training stage is crucial, and it is responsible for the learning process that conducts the evolution of individuals. The evolution leads to the final classifiers, which are mathematical expressions evolved through CGP.

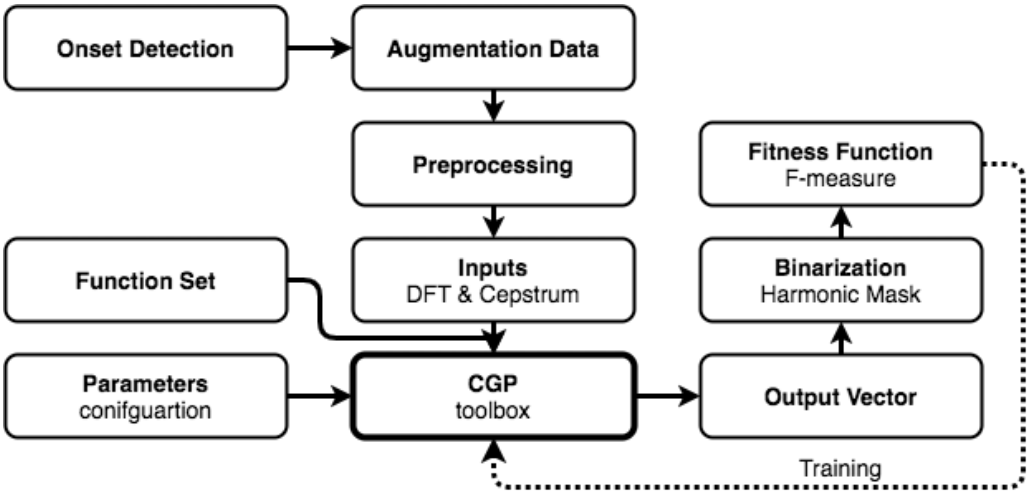

Figure 5. Our CGP system block diagram: the training stage starts with a augmentation data process using onset detection, it generates the inputs, the system engine is CGP toolbox that produces the output vector, then binarization and fitness evaluate the individuals and feedback for a CGP toolbox to proceed to the next generation of the evolutionary process.

5.1. Training

To develop a classifier system, the first stage consists in training the system classifiers. The training process of our system is described in Figure 5. First, onset detection is performed to infer where the musical notes are in the input audio signal. Then, according to the output of the onset detector, a data augmentation process is performed to increase the size of the training dataset and, thus, to improve the variability and inference capability of the system. Then, some preprocessing is applied on the acquired audio frames to generate the desired system inputs for all the classifiers. These inputs, along with the function set and CGP parameters are used in the CGP block, where the classifiers are evolved independently using our CGP toolbox [48]. During this evolutionary process, each classifier generates an output vector. Then, this output vector undergoes a binarization process to obtain a final binary output. Then, a fitness function is computed to evaluate the quality of the corresponding classifier using F-measure. The engine of the implemented

evolutionary process is the CGP block: it is responsible for the evolutionary process that is done during the training stage to accomplish one final classifier for each piano note: a graph of mathematical functions encoding a mathematical expression capable of detecting the presence of the sound produced by the corresponding piano key.

Many decisions and processes had to be made and implemented besides the CGP. We employed a data-augmentation mechanism (see Section 5.2). We implemented a preprocessing task (Section 5.3) to generate the system inputs and had to figure out which inputs to use and how to represent them (Section 5.4). Additionally, we had to create a gene encoding (Section 5.5) and we had to decide what types of mutations (Section 5.6) and which evolutionary strategy (Section 5.7) to use as well. We also developed a binarization strategy (Section 5.8) for the output and fitness computation (Section 5.9).

5.2. Data Augmentation

Data augmentation encompasses a suite of techniques that enhance the size and quality of training datasets [55]. In many classification problems, the available data are insufficient to train accurate and robust classifiers. Thus, to alleviate the relative scarcity of the data compared to the number of free parameters of a classifier, one popular approach is data augmentation (DA). Data augmentation consists in transforming the available samples into new samples using label-preserving transformations.

In our classification problem, the amount of data was large enough; however, it was not balanced for our multi-pitch approach. For each of the 61 classifiers to be properly trained, we needed to build a balanced dataset with two classes: positive cases and negative cases. Our training data built for each classifier and its training instances were extracted from the MIDI Aligned Piano Sounds (MAPS) database [56]. This way, we had 61 different classifiers and 61 different pitches. Therefore, the ratio of positive cases and negative cases in the database for each pitch was $\approx \frac{1}{60}$.

For the data augmentation process, first we applied our onset detector on the input audio signal. Then, based on the inferred onset position (I_{os}), we acquired three different audio fragments translated in time. Each acquired audio fragment had 4096 samples length ($\approx$ 0.93 milliseconds), sampled at frequency 44,100 Hz. An audio fragment was acquired at first, starting at the inferred onset time I_{os}. Then, two additional audio frames were acquired from the original signal, starting at $I_{os} - 512$ and $I_{os} + 512$. In practice, we translated the acquired window signal back and forward in time and, this way, we accomplished data augmentation with a factor of $3\times$. This process is illustrated in Figure 6.

This innovation in our CGP systems is useful to enlarge the number of positive cases on the training dataset and consequently allows us to enlarge the number of negative original cases maintaining the balance in the two classes. This way, we can train the classifiers with a more extensive and diverse negative training datasets. Another important advantage is that, this way, we also force the classifiers to learn and infer not only when the onset detector is accurate but also when it does not estimate the onset time with high precision (delays and advances).

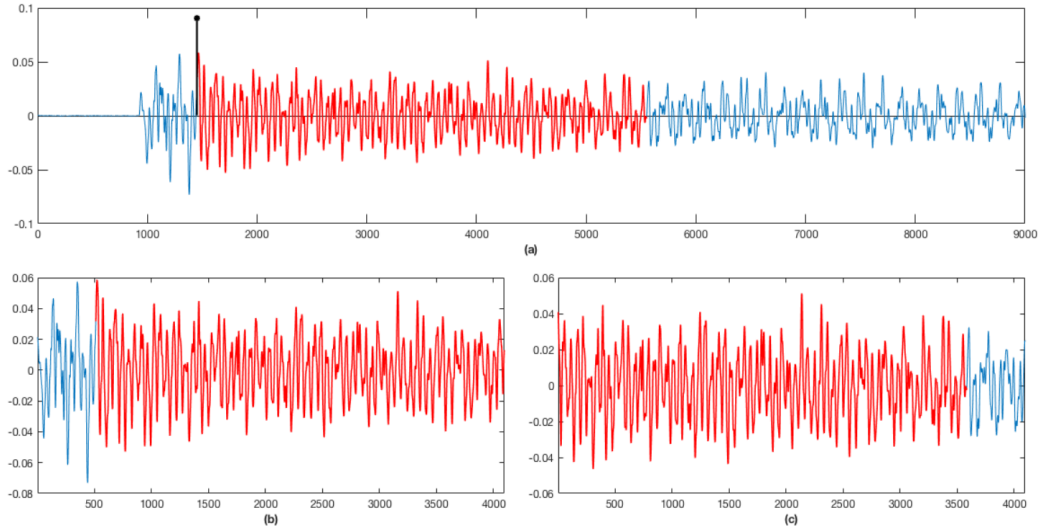

Figure 6. Data augmentation process using time translation—(**a**) in blue is the original piano signal in the time domain (≈0.2 s), in black is the inferred onset location I_{os}, and in red is the extracted frame, starting at I_{os}; (**b**) audio frame extracted from the original signal, starting at instant $I_{os} - 512$; and (**c**) audio frame extracted from the original signal, starting at $I_{os} + 512$.

5.3. Preprocessing

One important decision in designing classifier systems is defining the inputs and their structure. The original sound signals acquired from wave files are float vectors, representing input audio signals in the time domain. However, the frequency domain contains a lot of important information for multi-pitch estimation problems. As mentioned in Section 1, the polyphonic signals can also be expressed as a sum of harmonic sources plus a residual (Equation (1)). Thus, in the frequency domain, we have the information we need about the Fundamental Frequencies (F0) that composes the acoustic input signal. This stresses for a need for converting the input audio signal to the frequency domain. This transformation is done by employing the discrete Fourier transform (Equation (6)). For the preprocessing task, the three audio frames acquired in the data augmentation process are windowed (Equation (4)) using an Hanning window function (Equation (5)) to avoid spectral leakage. The windowing process is illustrated in Figure 7.

$$x_w[n] = w[n] \cdot x[n]. \tag{4}$$

$$w[n] = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right). \tag{5}$$

The windowed audio frames are transformed from the time domain to the frequency domain using discrete Fourier transform (DFT):

$$X[k] = \sum_{n=0}^{N-1} x_w[n] e^{-j\left(\frac{2\pi}{N}\right)nk}, (k = 0, 1, \cdots, N-1), \tag{6}$$

In discrete time processing, DFT is typically computed using the fast Fourier transform algorithm (FFT), which is much faster. We obtained the signal in the frequency domain $X[k]$ using Equation (6), where $x_w[n]$ is the windowed time signal and N is the number of samples. Any time signal cannot be uniquely represented for frequencies above $\frac{f_s}{2}$ (also known as the Nyquist frequency), where f_s is the sampling frequency of the sequence. Due to the periodicity of the frequency domain signal resulting from DFT, period $[0; f_s]$, and the Nyquist theorem where the symmetry of the real part and the anti-symmetry of the imaginary part relative to Nyquist frequency, $\frac{f_s}{2}$, we use half of the resulting signal of the DFT. Thus, from the resulting frequency signal representing in the frequency interval $[0; f_s]$

with a size of 4096, we can use only the first half represented in the interval $[0; \frac{f_s}{2}]$, with a 2048 length. The preprocessing process is illustrated in Figure 7.

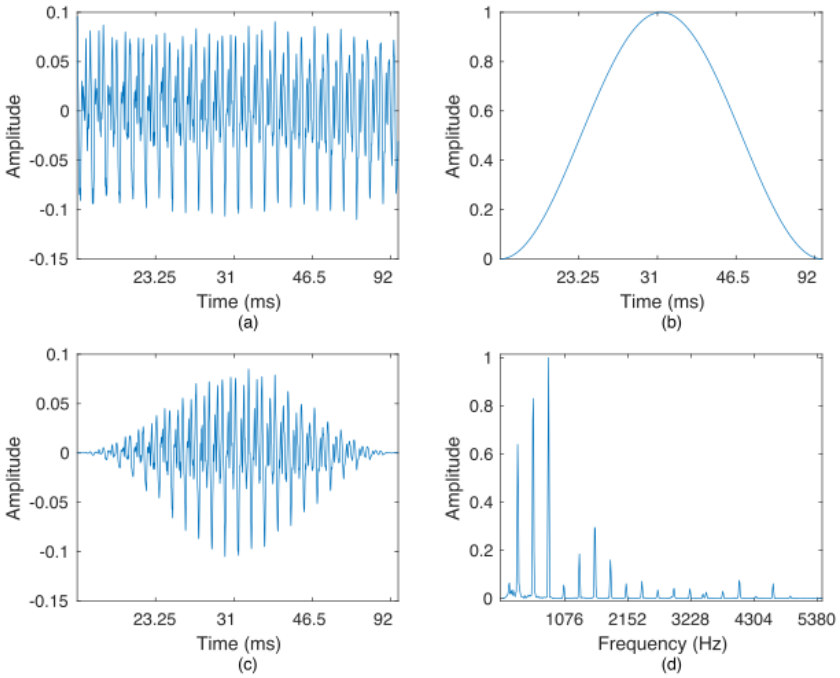

Figure 7. Preprocessing—(**a**) acquired audio frame in the time domain, (**b**) Hanning window representation, (**c**) windowed audio frame, and (**d**) windowed signal represented in the frequency domain after applying the DFT.

5.4. Inputs

As depicted in Figure 8, our system uses 5 input vectors, all acquired from the sample time vector. The vector $X[k]$ resulting from Equation (6) is in the frequency domain and is a vector of complex numbers. This lets us use two different vector representations of complex numbers (z): cartesian (Equation (7)) and polar (Equation (8)).

$$z = a + bi \tag{7}$$

$$z = r(\cos\theta + i\sin\theta) \tag{8}$$

This way, we have a pair of vectors for each representation (2 components), making 4 usable inputs: a, b, r, and θ. An additional 5th input was added to the system: the signal cepstrum. The cepstrum is a mathematical transformation used in audio signal processing, in particular, for period estimation. Cepstral analysis is one of several methods that enables us to find out whether a signal contains periodic elements. This way, cepstral information can also be used to determine the pitch of a signal [57]. Cepstrum, $c[n]$, is defined as the inverse DFT of the log magnitude of the DFT of a signal:

$$c[n] = F^{-1}\{log|F\{x[n]\}|\}, \tag{9}$$

where F represents the DFT and F^{-1} is the inverse discrete Fourier transform (IDFT). For a windowed frame audio signal $x[n]$, the cepstrum is as follows:

$$c[n] = \sum_{n=0}^{N-1} log\Big(\Big|\sum_{n=0}^{N-1} x[n]e^{-j\left(\frac{2\pi}{N}\right)nk}\Big|\Big)^{j\left(\frac{2\pi}{N}\right)nk}, (k = 0, 1, \cdots, N-1). \tag{10}$$

The cepstral coefficients describe the periodicity of the spectrum. A peak in the cepstrum denotes that the signal is a linear combination of multiples of the frequency pitch. The pitch period can be found as the number of the coefficient where the peak occurs.

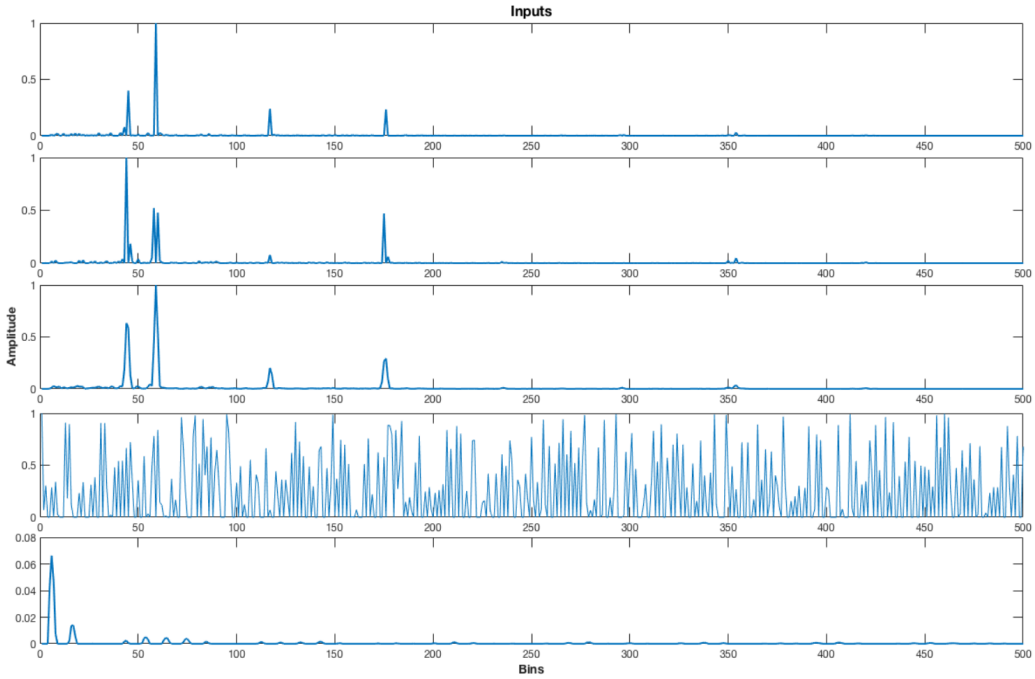

Figure 8. System inputs for a chord with 3 pitches (70 + 75 + 94), bottom down—real part of DFT, imaginary part of DFT, radius of DFT, angle of DFT, and real Cepstrum calculated from the time domain signal. Only the first 500 of 2048 bins of all the 5 vectors are shown.

Regarding the importance of the cepstral information in pitch estimation, we use it as our 5th system input. Figure 8 shows the 5 system inputs: four of them are directly obtained from the DFT of the original input audio signal: real (a), imaginary (b), radius (r), and angle (θ). The 5th input ($c[n]$) is obtained from the original sound and is represented on a different scale (All the inputs are normalized to a max value of 1. The y scale of the cepstrum graph is limited to 0.1). Due to the variety and redundancy of information, in relation to the 5 inputs, we ensure that the CGP system has a variety of representations of the same data, so that it can be able to choose the one that best fits to the problem.

5.5. Individual Encoding

Usually, Cartesian genetic programming contains three node types: input nodes, function nodes, and output nodes. Our system has 5 input nodes and only one output node for each classifier. We used a CGP, graph which is a grid of nodes with one row and 100 columns. Thus, each individual will have a maximum allowance of 100 nodes, which is a common value for this type of approaches [10]. Each function node contains 5 different genes (Figure 9). There is a gene that represents the function number (F_n) from the pre-established function set. The next two genes encode the node inputs: they can be the system inputs or the outputs of other nodes, and they are 2 because that is the maximum arity of our function set. Finally, there are two genes for the real parameters (P_1 and P_2) used as parameters for correct work of the corresponding mathematical functions from the function set. The F_n gene represents the function used by that node chosen from the function set represented by a lookup table. Note that all the functions are prepared to receive one or two vectors and that all of them return a vector. Our function set is almost comprised by filtering operations on vectors and by arithmetic operations with constants and vectors.

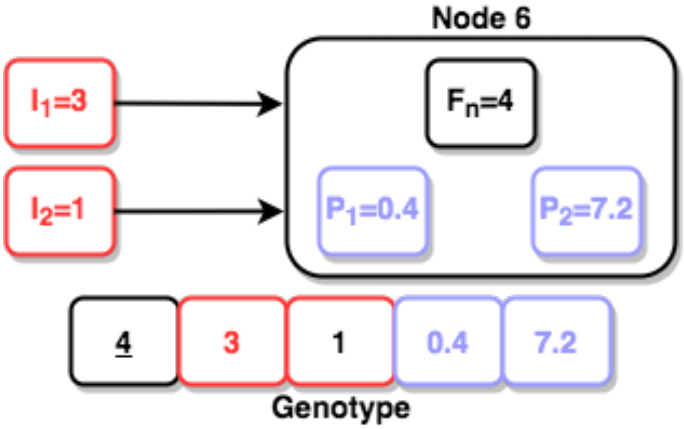

Figure 9. Node genotype and its graphical representation. Node 6 has five genes: the first gene (**black**) codifies the function from the function set table (value 4), the second and third genes (**red**) are connection genes (values 3 and 1: node 3 output and node 1 output), and the fourth and fifth genes (**blue**) are real parameters for function use (values 0.4 and 7.2).

Figure 9 also shows that there are two real parameters in the node genotype. These parameters are useful for the functions, since each function uses at most 2 parameters to accomplish its task. In fact, most functions need real parameters. However, the same parameter has a different meaning and a different domain from function to function. To avoid a tremendous increase in parameters and genes in each node, we used one real parameter to represent constants and another one to represent a percentage of an interval. This way, with only two additional genes, we managed to fulfill all function set needs. As the same parameter may have one meaning and domain for one particular function and another meaning and domain for another function, we normalized the domain of the parameter P_2 to a fix interval. Each function is responsible for mapping that fix interval to the correct domain for its own purpose. The parameter p_2 of each function with its own range is normalized into $[0;1]$: all intervals are transformed from $[a;b]$ to a normalized one $[0;1]$. By using this technique, the actual value of any parameter can be seen as a number between 0 and 1 or a percentage of the interval, and the mutation process becomes standard and easier.

5.6. Mutation

Mutation process plays a fundamental role in the evolutionary process of Cartesian genetic programming-based systems. Without crossover, mutation is the only process responsible for the generation of new individuals in an offspring. The mutation process depends on two different stages, ruled by different probability distributions functions: the first stage decides if and which gene or genes will be mutated; the second stage decides how these genes will be mutated. There is a configurable parameter, the mutation probability that represents the probability of each gene undergoing a mutation. For instance, $p = 0.015$ means that each gene will mutate with a 1.5% probability. Different mutations are performed according to the gene type and domain: if a function gene happens to be mutated, then a valid value must be chosen for selecting a new function in the function set lookup table; if a mutation occurs in a gene node input, then a valid value is the output of any previous node in the genotype or any system input; and the valid values for the system output genes are the output of any node in the genotype or the address of a system input. All these mutations happen according to the discrete uniform probability distribution function for integers. Two additional genes can also mutate: the real parameters used by the functions. These parameters are important because they are used by those functions to perform specific tasks. According to each function, each parameter has a specific meaning and has its own domain range. In this case, we use the normalized interval $[0, 1]$ as the

mutation domain. The mutation of real genes (function parameters) is done using the normal distribution in order to address the entire range:

$$f(x) = \frac{e^{-(x-\mu)^2/(2\sigma^2)}}{\sigma\sqrt{2\pi}}, \tag{11}$$

where $f(x)$ represents the density function of an x variable, with a normal distribution. This function is also represented as $N(\mu, \sigma)$, where μ is the mean and σ is the standard deviation. To perform the mutation of a function parameter, P_{old}, we generated a new random P_{mutate} using the normal distribution $N(\mu = P_{old}, \sigma)$, with σ being configurable in our system. This way, we ensured that, when a mutation occurs on a real parameter, all parameter intervals are reachable but with higher probability to mutate to closer values.

For better performance, we maintained a list of active and inactive genes and, when there is no mutation in one of the active genes, the fitness function is not computed.

5.7. Evolutionary Strategy

The evolutionary strategy widely used for CGP is a special case of the $\mu + \lambda$ [58], where $\mu = 1$ (see Algorithm 2).

Algorithm 2 Algorithm $((1 + \lambda)EA)$.

1: $t \leftarrow 0$;
2: Set current individual I_0 as the best of λ individuals created randomly;
3: **while** a stop condition is not fulfilled, **do**
4: **for** i = 1 to λ **do**
5: Create a copy x_i of current individual I_t;
6: Mutate each gene of x_i with probability p;
7: **end for**
8: Set new current individual I_{t+1} as the best of $I_t \cup \{x_1, \ldots, x_\lambda\}$;
9: $t \leftarrow t + 1$;
10: **end while**

This means that, in this special case, the population size is always $1 + \lambda$. First, we select the best from λ randomly created individuals. Then, at each iteration (generation), λ new individuals are generated by applying mutations on the individual previously selected as the best among the population. Then, the best among the current individuals (1 + λ) becomes the current individual for the next iteration (generation). An offspring can become the current individual in the next iteration when it has the same fitness as the current individual and there is no other individual with a better fitness. According to Goldman [59], an empirical value for λ is 4, which was the value we used.

5.8. Binarization

Our CGP system architecture evolves one classifier for each piano note. The final output of each classifier should be a binary output: when the corresponding note is detected, the output is 1; otherwise, it is 0. In order to evolve each classifier, there is a training stage (see Section 5.1). During this stage, each generated individual for each classifier is evaluated using a fitness function. To compute the fitness value, we use F-measure, which is a measure of a test's accuracy for binary classification. Therefore, for each audio frame used as input, we need to generate a binary output for each classifier. However, our CGP system is comprised of mathematical functions in which the arguments are vectors and that also return vectors. Thus, at the end of each CGP graph (output node), we have a vector of float values. A binarization process was added at the end of each

classifier to transform its output vector into a binary output. This was done by applying a spectral mask with harmonic information: each classifier uses its own mask built by the system. The fundamental frequency of the corresponding note detected by the classifier (F_0) is computed and 2 or more triangles are placed centered in $n \times F_0$ (see Figure 10b); in this case, we use a harmonic mask parameter $n = 2$. This means that we use 2 triangles ($n = 1$ and $n = 2$). The amplitude and width of each triangle are also configurable parameters as well as the number of harmonics. To calculate F_0 for a piano note (key), we use the following equation that gives the fundamental frequency $F_0(n)$ of the n^{th} key:

$$F_0(n) = 440 \times 2^{\frac{n-49}{12}} Hz, \tag{12}$$

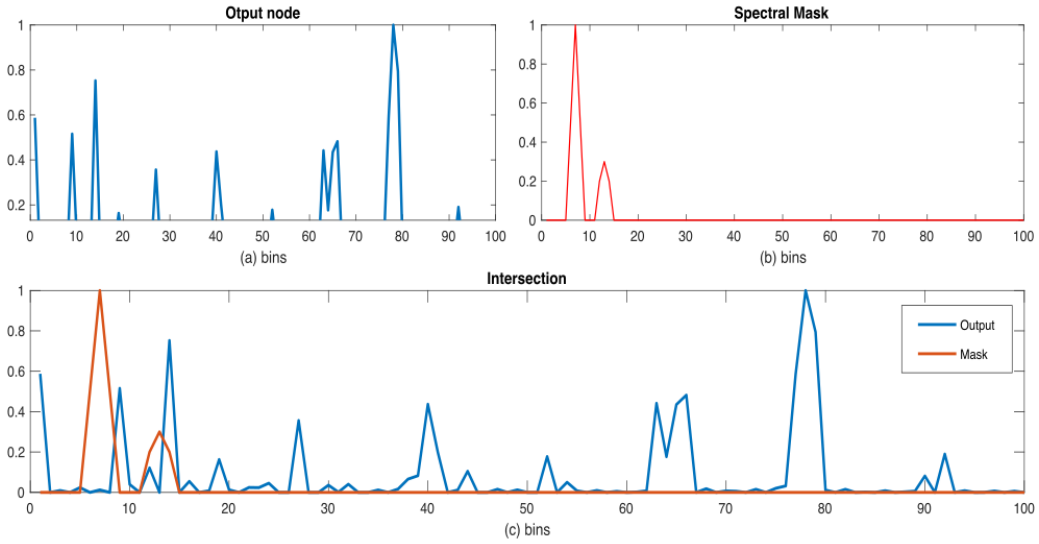

Figure 10. (**a**) CGP output signal, (**b**) harmonic mask, and (**c**) computing intersection.

In order to accomplish a binary output, we use a comparison process between the CGP output vector normalized in amplitude $O_{cgp}[n]$ and the spectral harmonic mask with the frequency corresponding to the pitch of the estimator, $M_{F0}[n]$. The first step is normalization of the output vector in amplitude. This way, all the elements of the vector fall in the interval [0; 1]. Then, we generate the following scalar computing the inner product between the 2 vectors:

$$x = \sum_{n=0}^{N} O_{cgp}[n] \times M_{F0}[n], \tag{13}$$

where x measures the discrete intersection between two discrete signals. If we approximate these signals to a continuous domain, we could see x as the intersected area between the two signals.

Finally, we use a threshold function to accomplish the binary result:

$$T(x) = \begin{cases} 1, & \text{if } x > \theta \\ 0, & \text{if } x <= \theta \end{cases} \tag{14}$$

where θ is the threshold value. Since both signals are normalized, the max value for x is as follows:

$$x_{\max} = \sum_{n=0}^{N} M_{F0}[n]. \tag{15}$$

This threshold value also evolves (mutates) during the training stage. This way, besides the node genes, the threshold may also adapt to reach better fitness. The threshold mutation probability is configurable and was empirically set to the same mutation probability of the other genes. When the system decides to mutate the threshold, its mutation values are ruled by the normal distribution $N(\mu = \theta_{old}, \sigma)$, which is confined to the interval $[0; x_{\max}]$. The new threshold depends on the old threshold, that is, the mean of the normal probability distribution that generates the new value.

5.9. Fitness Evaluation

Thanks to the binarization process, each CGP generated graph can act as a classifier and dictate whether the corresponding piano key is present on the input audio signal. This way, during the evolutionary training process, each classifier can be evaluated according its classification. This evaluation is done using F-measure (Equation (16)).

$$F_{measure} = 2 \times \frac{recall \times precision}{recall + precision}, \tag{16}$$

where

$$precision = \frac{tp}{tp + fp}, recall = \frac{tp}{tp + fn}. \tag{17}$$

The main goal of the training or evolutionary process is to maximize the $F_{measure}$ in order to reach the most accurate classifier.

6. Experiments and Results

To make a detailed study about the quality of the proposed system results, we made several experiments. The main goal was to evolve 61 classifiers during the training stage, each one for the correspondent piano key, from C2 (MIDI note 36) to C7 (MIDI note 96). Each piano key is represented by the corresponding MIDI note number, with 60 being the MIDI note number corresponding to the C4 musical note (middle C).

To train and test our system, we used the MAPS (MAPS stands for MIDI-Aligned Piano Sounds, and it is available for use under previous request.) database [56]. It is a piano sound database dedicated to research on multi-F0 (multi-pitch) estimation and automatic music transcription. It contains piano sound samples of polyphonic and monophonic sounds, usual chords, and random chords played by 7 different pianos in different recording conditions.

Throughout this section, we describe all the steps we performed to test our proposal and their advantages. We started by (i) training and testing some classifiers using the K-fold cross validation method to estimate the skill of our machine learning model on unseen data. Then, (ii) we expanded the experiments to train the CGP system evolving all 61 classifiers. To obtain a comparative base with other algorithms, we tested our system composed by the 61 classifiers previously trained with a larger amount of polyphonic piano sounds: 3000 random chords and monophonic notes from seven different pianos. We extended the experiments to other musical instruments (iii), such as an electric guitar. We also tested (iv) the time performance of our CGP system. Finally, (vi) we decoded one classifier to show the mathematical expressions and functions resulting from the evolutionary processes and the possibility of white-box optimization.

6.1. Validating the Proposed Methodology

Cross-validation is a model validation technique for assessing how the results of a statistical analysis are able to generalize to a different daset. It is mainly used in settings where the goal is prediction or classification and to estimate how accurately a model will perform in practice. We used a 5-fold cross-validation with folds of 100 cases, corresponding to a 500-case dataset for 10-pitch classifiers, randomly chosen. For a specific classifier, we built a dataset with 500 cases, 250 positive cases, and 250 negative cases to obtain a balanced dataset. All of them were extracted from the MAPS database, where there are monophonic

sounds (polyphony 1) and polyphonic sounds, from polyphony 2 to polyphony 6. All the chords were extracted from the random chords subset, which contains random-pitched chords from C2 to B6 uniformly distributed. These chords are composed by random notes without any music rules, such as harmoncity and musical consonance. Training the classifiers with random-pitched chords is an important characteristic because, this way, the evolved classifiers will neither be constrained to any type of music, for example, western music nor be constrained to any music rules.

We trained each of the ten classifiers using 4 folds and tested it with the remaining fold, and we repeated this process for the five combinations of 4 folds. For each of the training and test processes, we computed 10 runs. The k-fold cross-validation and the configurable parameters of our proposed CGP system used are presented in Table 1. The evolutionary process consisted of 10 runs with 10,000 generations each, using 200 positive and 200 negative cases for each fold combination.

Table 1. The 5-fold cross-validation parameters trained/tested.

Parameter	Value
K-folds	5
Data Augmentation	3
Positive Test Cases	250
Negative Test Cases	250
Frame Size	4096
Fitness Initial Threshold	1.5
Outputs	1
Rows	1
Columns	100
Levels Back	100
λ ($E.S.\ 1 + \lambda$)	4
Mutation Probability	5%
Threshold Mutation Probability	6%
Harmonic Mask	2
Runs	10
Generations	10,000

For individual encoding, we used one row with 100 nodes. The "harmonic mask" value was empirically set to 2: the CGP system was prepared using a harmonic mask from the first harmonic to the fifth harmonic. "Data augmentation" was set to 3, which means that, for each initial sound file, we generated 3 different ones with time translations forward and backward. The evolutionary process consisted of 10 runs with 10,000 generations each, using the 5-fold dataset division for training and test. The classifiers were evaluated using the F-measure metric, explained in Equation (16).

Figure 11 shows the 5-fold cross-validation results. The mean value (μ) for F-measure is 0.93 with 0.91 of precision and 0.93 of recall; these values have standard deviations (σ) of 0.027, 0.025, and 0.025, respectively. These values prove that our CGP-based technique is able to learn.

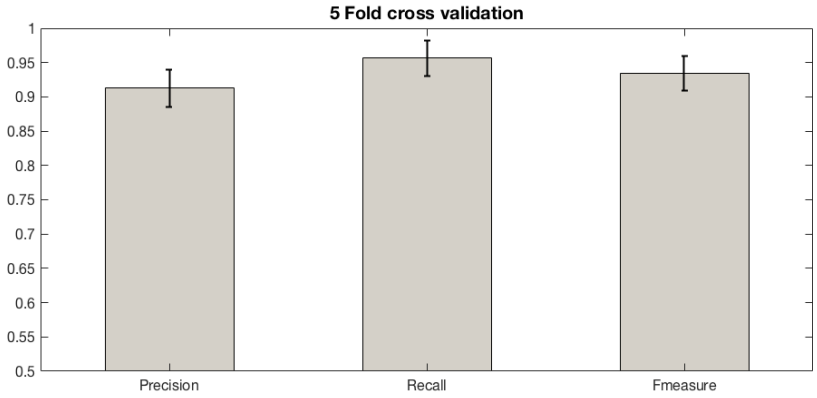

Figure 11. Results of 5-fold cross-validation for ten classifiers, using 500 cases dataset, half positive half negative. Precision ($\mu = 0.91, \sigma = 0.027$), recall ($\mu = 0.95, \sigma = 0.025$), and F-measure ($\mu = 0.93$, $\sigma = 0.025$).

6.2. Classifier Training

After demonstrating the quality of our machine learning-based system in the previous section, using the 5-fold cross-validation for a set of classifiers, we had to train all of the 61 pitched-based classifiers in a way to obtain the best possible results for the testing stage. Thus, we performed complete training for all 61 different piano keys to evolve the 61 classifiers. We used a bigger dataset for the training process, consisting of 600 cases of monophonic and polyphonic sounds extracted from the same database used before. The CGP system parameters are identical to those in Table 1 except the number of runs per classifier, which we increased to 20. Therefore, each classifier uses a set of 600 cases, 300 positive case, and 300 negative to evolve 10,000 generations.

Figure 12 shows that the mean F-measure calculated over all the 61 classifiers is 0.95 with $\sigma = 0.024$. These are training results, as usually we obtained high values for all 3 measures computed. This trained classifiers will be tested below on a bigger dataset to obtain the final test results and for comparison with other state-of-the-art algorithms.

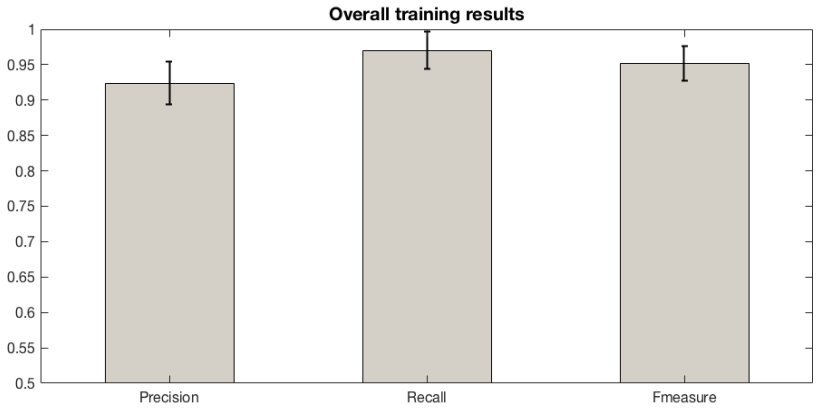

Figure 12. Results of the overall training stage for 61 classifiers. Precision ($\mu = 0.92, \sigma = 0.029$), recall ($\mu = 0.97, \sigma = 0.026$), and F-measure ($\mu = 0.93, \sigma = 0.024$).

6.3. Testing

The cross-validation process results gave us an idea about the quality of the classifiers. However, the important and comparable results are those obtained in the test stage. For the test phase, we used the classifiers evolved during the training stage to identify the presence of their corresponding pitches in monophonic and polyphonic audio files. We measured the quality of the classifiers by testing them with a different dataset: the test set. The test set for each classifier consisted in 3000 piano audio files of chords and single notes extracted from the MAPS database [56], corresponding to almost 5 min of music:

3000 random pitched chords and single notes between C2 (65.406 Hz) to C7 (2093.0 Hz), with polyphony levels ranging from 1 to 6. For each sound, a single 93 ms frame located after the onset time was extracted. The test results are illustrated in Figure 13.

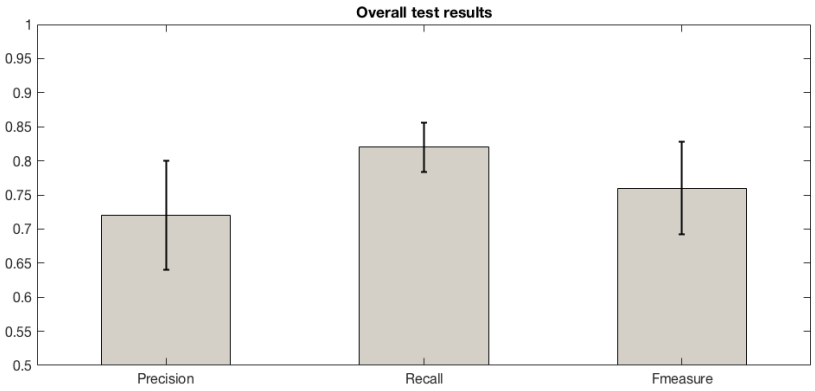

Figure 13. Results of the overall testing stage for 61 classifiers. Precision ($\mu = 0.72, \sigma = 0.08$), recall ($\mu = 0.82, \sigma = 0.036$), and F-measure ($\mu = 0.764, \sigma = 0.06$).

We accomplished a final F-measure value of 0.76 (76%). As expected, these are slightly lower results when compared to the values obtained during the training stage. However, they are very competitive in the MPE context. We also calculated the accuracy using F-measure for each polyphony from 1 to 6 during the test process. Our system returns F-measures in percentages of 77%, 79%, 77%, 73%, 69%, and 66% for polyphony (Polyphony 0 corresponds to silence, easily detected by an analytic power spectrum analysis.) 1, 2, 3, 4, 5, and 6, respectively (Figure 14). Our CGP system presents higher F-measure values for polyphony 2 and 3, and then, it decreases until polyphony 6, with a 13% gap between the top and bottom polyphony F-measure values.

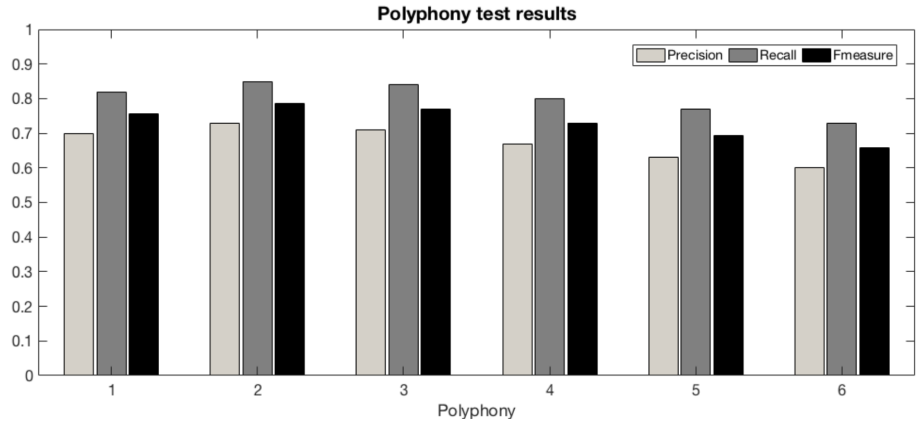

Figure 14. Polyphony test results from polyphony 1 (monophonic) to polyphony 6 for precision, recall, and F-measure.

The entire database includes sounds from seven different pianos. Analysing the results by piano, we have about 4% standard deviation. This means that our system accuracy does not significantly depend on piano differences, recording conditions, or whether a real piano or a software-based one is used, which suggests robustness for varying production conditions.

Our final results are expressed in Figure 13. These are the results obtained for each of the improvements we applied. Previous versions and several improvements were made since we initially proposed our toolbox for CGP [48] until this last version called CGP-harmonic mask (HM)-DA. All the previous published results are shown in Figure 15. Our first F-measure result was 62.7% using CGP-1 [46,48]. We improved our system using a

harmonic mask, CGP-HM, in [49]; later, some functions were added to the function set and minor improvements were made, CGP-HM2 [60], resulting in a better F-measure, 73%. Finally, we were able to extend the training set using a artificial data augmentation process, CGP-HM-DA, producing our best results described in this article, reaching F-measure values over 76%.

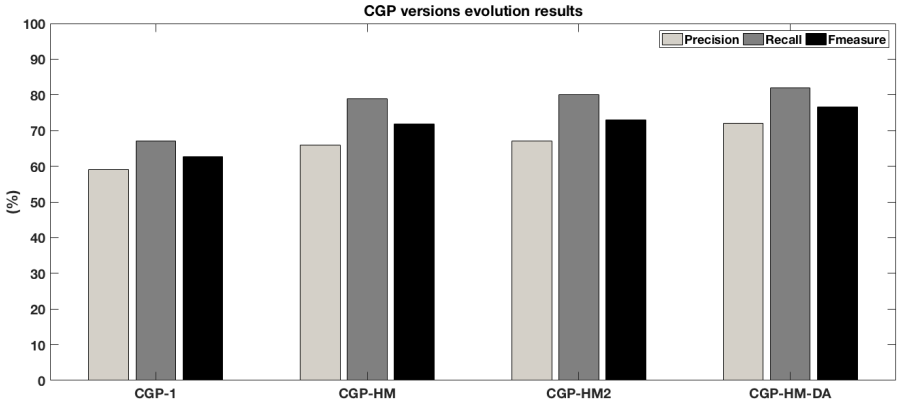

Figure 15. Our CGP system versions: from CGP-1, our first approach, to CGP-harmonic mask (HM)-data augmentation (DA), the last version, and it is the described approach with harmonic mask and data augmentation. Results from 62.7% to 76.6%.

Our training stage consumes a lot of time: to train 10 runs with 10,000 generations using a multi-core (4 core) processor, it takes more than a day. Nevertheless, for some classifiers, we performed additional tests increasing the number of generations to 20,000. Unfortunately, we did not have the necessary time or the computer resources to completely train with 20,000 generations for all the classifiers. Our preliminary results show a mean increase in F-measure of 1.2%.

6.4. White-Box Optimization

Many machine learning algorithms were proven to successfully accomplish their tasks. However, some of them featured "opacity": they do not provide information about how they do it in an analytic way and were thus labelled as *black box*. Our CPG multi-classifier system is completely transparent. It is known that each classifier is composed by a graph of nodes, where each node is essentially a mathematical function from a defined function set. Thus, we can pick any evolved classifier, decode it, and analyze its readable function graph or mathematical expression. This provides a *white-box* analysis. Using classifiers as a white box has several advantages. Among them, two stand out: one is the possibility to learn from the classifiers obtained, by checking what mathematical functions and inputs are used to make a classification; the second is the possibility to optimize the classifier, tuning the constant values used by the encoded functions.

Figure 16 shows one of the evolved classifiers, classifier 55, which as a F-measure of (95%). Looking at our classifier, the output is computed by a sum (last node) of the resulting expressions: one uses a Gaussian filter of two different normalizations of the cosine of I_4 that is the angle of the DFT of the sound signal, and the other branch of the graph uses two consecutive sines of I_3, which is the radius of the number of DFTs. Among all the functions, "SPGaussFilter" executes a sliding window filtering process using a Gaussian function. This function uses a real parameter to design the Gaussian filter. That parameter is the standard deviation (σ). Therefore, a good start to optimizing this classifier could be tuning this real constant value.

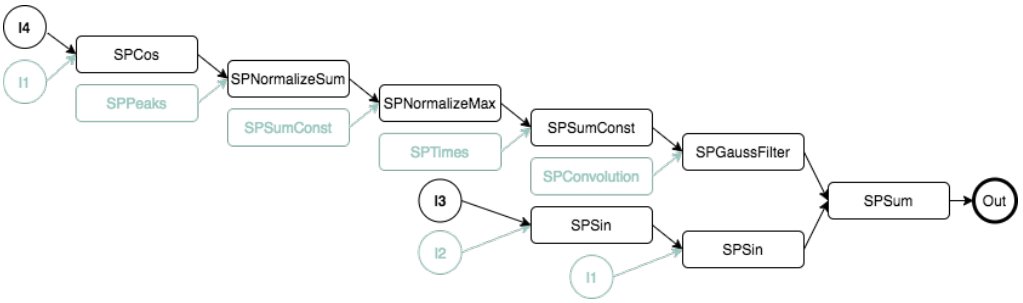

Figure 16. Resulting graph after decoding the genotype of classifier 55. The rectangles are functions from the function set, the circles are inputs, and the bold one is the output. In grey are the nodes that are not used to calculate the output due to the arity of some functions.

The evolved classifiers are diverse as well as the functions used, although there are almost always functions that use one or two real parameters. Our function set is detailed in Table A1 in Appendix A.

6.5. Comparing to Other Approaches

Although there is a lack of evolutionary approaches on multi-pitch estimation problems in the literature, specially using genetic programming and with multiple independent classifiers, we compare our results with 4 different state-of-the-art algorithms for MPE of piano. These were the algorithms that were tested previously by Emiya [27] in the same conditions as ours: MPE for random chords using the MAPS database including all pianos, artificial and real.

The 4 algorithms were tested on the same database for comparison purposes. The first one was Tolonen's multipitch estimator [20]. As the performance of the algorithm decreases when F0s are greater than 500 Hz (C5), the system was additionally tested in restricted conditions—denoted Tolonen-500—by selecting from the database the subset of sounds composed of notes between C2 and B4 only. The third was Emiya [27]. The fourth one was Klapuri's system [15], for which we also had access to the code provided by the author and which was upgraded later [22]. The overall results shown in Figure 17 demonstrate that our system reaches 76% in F-measure, being the third best in quality, only behind Emiya with 80% and Klapuri with 82%. Tolonen reached 47%, and Tolonen-500 reached 61%. These results confirm the suitability of the technique we employed. Moreover, as we will show below, our algorithm is able to go beyond piano and has some unique features that stand out.

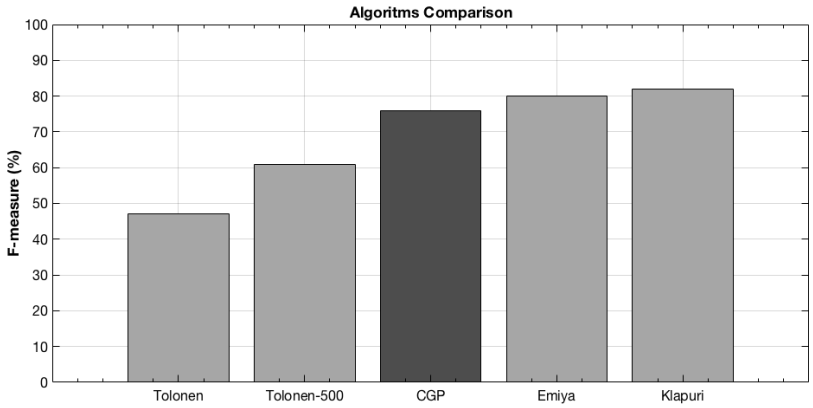

Figure 17. F-measure results comparison in (%) with state-of-the-art algorithms: Tolonen, Tolonen-500, Emiya and Klapuri. Our proposal is referenced to as CGP and it is the last version of our algorithm, CGP-HM-DA.

We are aware that there are more recent state-of-the-art algorithms with salient published results, such as [23]. However, these recent algorithms were designed and prepared

to perform music transcription, and they encapsulate the process of multi-pitch estimation. Moreover, those were trained and tested in different conditions and, thus, cannot be compared with our approach.

6.6. Reaching for the Piano and Beyond

Our main task was to develop a multi-classifier system strong and flexible enough that it can be tested with piano music and then easily extended to other instruments. After the exhaustive study on MPE for piano music, we extended our experiments to other instruments, starting with an electric guitar (Figure 18).

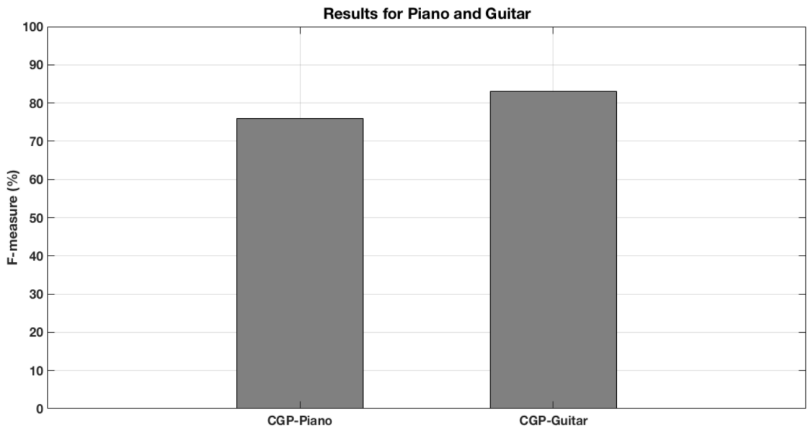

Figure 18. F-measure results for MPE on 2 different instruments: for piano, the F-measure is 76%, and for guitar, it rises to 83%.

We used a dataset with 400 guitar chords and single notes extracted from the guitar database, IDMT-SMT-GUITAR [61]. Seven different guitars in standard tuning were used with varying pick-up settings and different string measures to ensure a sufficient diversification in the field of electric guitars. The results depicted in Figure 18 show that the F-measure for guitar reaches 83%, which is even better than the piano results. This way, we proved that our proposed multi-classifier system-based CGP has the ability to perform MPE for other musical instruments besides piano, with even better results.

6.7. Discussion

To discuss the quality of our proposal we have to look beyond the accuracy results. Besides the F-measure results, which have demonstrated the competitiveness of the approach not only for piano but also for electric guitars, there are a set of features that should be considered when a machine learning technique must be selected for solving a problem. We summarize some of these desirable features in Table 2.

Table 2. The algorithm's main features.

Algorithm	Real-Time	Other Instruments	White Box	Piano FM (%)	Guitar FM (%)
Tolonem	√	x	x	47%	-
Tolonen-500	√	x	x	61%	-
CGP	√	√	√	**76%**	**83%**
Emiya	x	√	x	80%	-
Klapuri	x	√	x	82%	-

Our CGP algorithm for MPE is unique in terms of the technique used to address the problem we face. It is the first one that embodies genetic programming and a distinctive architecture with multiple classifiers that can work independently and in parallel. As far as we know, there is no other approach for this kind of problem with a similar multi-classifier architecture. In fact, our architecture is highly parallelizable: since the evolved classifiers are independent, they can run in parallel. Our system has been preliminary tested on a 8 core processor and, due to its parallelization capabilities, it works in real time. Therefore, as shown in the table, our CGP-based approach is among real-time techniques that provides the best F-measures.

We easily extended our CGP system to other instruments with accurate results. We trained and tested our system for electric guitars and we reached an outstanding 83% F-measure. This suggests that this is the only real-time technique developed for piano that may be applied without changes to any instrument. Although we have successfully tested it for a guitar, we hope the results will be positive with other instruments, given that, after piano, the guitar is another instrument with a high polyphony degree.

One of the most important features that stands out from our system is the white-box method that allows us to study the solutions provided by avoiding opacity and by allowing future improvements based on what we can learn. Again, this is the only technique among state-of-the-art techniques featuring white-box optimization.

Summarizing, as shown in Table 2, the CGP-based system is the only one that provides a real-time approach, provides white-box optimization, is easily extendable to other instruments, has competitive F-measure for piano music, and has outstanding F-measure values for guitar music.

7. Conclusions and Future Work

We proposed in this paper a multi-pitch estimation system for piano music based on Cartesian genetic programming using an harmonic mask and improved with a data augmentation process for a better training stage.

We presented a detailed description of all the system features, which perform multi-pitch estimation using Cartesian genetic programming, harmonic masks, and a data augmentation process. We also introduced an onset detection analytic algorithm to aid the system working process.

The main disadvantage of our proposal is the fact that one first needs to evolve the 61 classifiers, so that the system is able to employ those classifiers to perform transcription and so that the training (evolutionary) process takes time: each single classifier takes one day to train (ten runs). However, our technique produces competitive results for any type of piano sound without the need for harmony rules for chords construction. This way, it is more generalized and works with random notes, which could be of interest for contemporary music. Moreover, we have also shown that the methodology can be applied without changes to other polyphonic instruments, such as the guitar.

The CGP-HM-DA approach based on a graphs of nodes with functions extracted from a dataset can be easily decoded once the classifiers are evolved for a given instrument: the classifiers are made up of mathematical expressions with inputs, internal functions, and outputs, which provide *white-box* optimization.

The results are shown and compared with state-of-the-art algorithms, and we demonstrated the feasibility of the approach: this technique is the only one providing simultaneously (i) competitive results (F-measure) together with (ii) real-time capabilities thanks to their intrinsic parallel nature; (iii) white-box optimization; (iv) and direct application to other polyphonic instruments with outstanding results (F-measure).

Future work will focus on demonstrating the capabilities of the proposed system on other types of pitched instruments, namely woodwind and string ensembles. We also look forward to proposing a similar system for non-pitched instruments, such as the drums. Given the parallel nature of our approach and its ability to perform in real-time, we also aim to implement a real-time improvisation music composition system, using the proposed

algorithm as the system input. Here, the goal is to create a system that is capable of interacting through its music and to play with other musicians.

Author Contributions: Conceptualization, investigation, and methodology, R.M., F.F. and G.R.; software, R.M. and T.I.; supervision, F.F. and G.R.; funding acquisition, F.F.; writing—original draft preparation, R.M., F.F. and G.R.; writing—review and editing, F.F. and G.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Spanish Ministry of Economy and Competitiveness under project TIN2017-85727-C4-{2,4}-P; by the Regional Government of Extremadura, Department of Commerce and Economy, the European Regional Development Fund, a way to build Europe, under project IB16035; and by Junta de Extremadura, project GR15068 and project GR18049.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank to P. Domingues for providing powerful hardware at Polytechnic of Leiria to perform important tests; P. Chavez for his hard work on configuring the blade machines at the Universidad de Extremadura, Mérida, so that we could perform tests; and Jorge Alvarado for technical support on installing and configuring software and virtualization solutions. We also would like to thank CIIC—Computer Science and Communication Research Centre—for the logistic support and co-funding by FCT—Fundação para a Ciência e Tecnologia, I.P., under the projectUIDB/04524/2020.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AMT	Automatic Music Transcription
MIR	Music Information Retrieval
MIDI	Musical Instrument Digital Interface
CGP	Cartegian Genetic Programming
MPE	Multi-Pitch Estimation
EA	Evolutionary Algorithm
CGP4Matlab	Cartesian Genetic Programming for Matlab
DFT	Discrete Fourier Transform
IDFT	Inverse Discrete Fourier Transform
FFT	Fast Fourier Transform
STFT	Short Time Fourier Transform
HM	Harmonic Mask
DA	Data Augmentation

Appendix A. Function Set

Our function set contains 27 different functions with arities 1 and 2. The functions may use a maximum of 2 real parameters. This function set is composed by arithmetic functions, filtering functions, and transforms. It was used in all of our experiments.

Table A1. Function set lookup table.

Index	Function	Description
1	SPAbs	Absolute value
2	SPBPGaussFilter	Band-pass Gaussian filter
3	SPCceps	Cepstrum transform
4	SPConvolution	Convolution
5	SPCos	Cosine
6	SPDivide	Point-to-point Division
7	SPEnvelopeHilbert	Hilbert envelope
8	SPFFT	Absolute value of the DFT
9	SPGaussfilter	Gaussian filter
10	SPHighPassFilter	High-pass filter
11	SPIFFT	Absolute value of inverst DFT
12	SPLog	Natural logarithm
13	SPLog10	Common logarithm
14	SPLowPassFilter	Low-pass filter
15	SPMedFilter	Median filter
16	SPMod	Remainder after division
17	SPMulConst	Multiplication by constant
18	SPNormalizeMax	Normalization maximum
19	SPNormalizeSum	Normalization sum
20	SPPeaks	Find peaks
21	SPPowe	Power
22	SPSin	Sine
23	SPSubtract	Subtraction
24	SPSum	Sum
25	SPSumConst	Sum with a constant
26	SPThreshold	Tresholding
27	SPTimes	Multiplication

References

1. Casey, M.A.; Veltkamp, R.; Goto, M.; Leman, M.; Rhodes, C.; Slaney, M. Content-based music information retrieval: Current directions and future challenges. *Proc. IEEE* **2008**, *96*, 668–696. [CrossRef]
2. Shields, V. Separation of Additive Speech Signals by Digital Comb Filtering. Master's Thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, USA, 1970.
3. Moorer, J.A. On the transcription of musical sound by computer. *Comput. Music J.* **1977**, *1*, 32–38.
4. Piszczalski, M.; Galler, B.A. Automatic music transcription. *Comput. Music J.* **1977**, *1*, 24–31.
5. Emiya, V. Transcription Automatique de la Musique de Piano. Ph.D. Thesis, Télécom ParisTech, Paris, France, 2008.
6. Olson, H.F. *Music, Physics and Engineering*; Courier Corporation: Chelmsford, MA, USA, 1967; Volume 1769.
7. Lee, C.T.; Yang, Y.H.; Chen, H.H. Multipitch estimation of piano music by exemplar-based sparse representation. *IEEE Trans. Multimed.* **2012**, *14*, 608–618.
8. Yeh, C.; Roebel, A.; Rodet, X. Multiple fundamental frequency estimation and polyphony inference of polyphonic music signals. *IEEE Trans. Audio Speech Lang. Process.* **2009**, *18*, 1116–1126.
9. Goldberg, D.E.; Holland, J.H. Genetic Algorithms and Machine Learning. *Kluwer Academic Publishers-Plenum Publishers*; Kluwer Academic Publishers: Alphen aan den Rijn, The Netherlands, 1988.

10. Miller, J.F.; Harding, S.L. Cartesian genetic programming. In Proceedings of the 10th Annual Conference Companion on Genetic and Evolutionary Computation, Atlanta, GA, USA, 12–16 July 2008; pp. 2701–2726.
11. Harding, S.; Leitner, J.; Schmidhuber, J. Cartesian Genetic Programming for Image Processing. In *Genetic Programming Theory and Practice X*; Springer: New York, NY, USA, 2013; pp. 31–44.
12. Benetos, E.; Dixon, S.; Giannoulis, D.; Kirchhoff, H.; Klapuri, A. Automatic music transcription: challenges and future directions. *J. Intell. Inf. Syst.* **2013**, *41*, 407–434. [CrossRef]
13. Wang, D.; Brown, G.J. *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*; Wiley-IEEE Press: Hoboken, NJ, USA, 2006.
14. Christensen, M.G.; Jakobsson, A. Multi-pitch estimation. *Synth. Lect. Speech Audio Process.* **2009**, *5*, 1–160. [CrossRef]
15. Klapuri, A.P. Multiple fundamental frequency estimation based on harmonicity and spectral smoothness. *IEEE Trans. Speech Audio Process.* **2003**, *11*, 804–816. [CrossRef]
16. Bello, J.P.; Daudet, L.; Sandler, M.B. Automatic piano transcription using frequency and time-domain information. *IEEE Trans. Audio Speech Lang. Process.* **2006**, *14*, 2242–2251. [CrossRef]
17. Saito, S.; Kameoka, H.; Takahashi, K.; Nishimoto, T.; Sagayama, S. Specmurt analysis of polyphonic music signals. *IEEE Trans. Audio Speech Lang. Process.* **2008**, *16*, 639–650. [CrossRef]
18. Dressler, K. Pitch estimation by the pair-wise evaluation of spectral peaks. In Proceedings of the 42nd International Conference: Semantic Audio, Ilmenau, Germany, 22–24 July 2011.
19. Indefrey, H.; Hess, W.; Seeser, G. Design and evaluation of double-transform pitch determination algorithms with nonlinear distortion in the frequency domain-preliminary results. In Proceedings of the ICASSP '85. IEEE International Conference on Acoustics, Speech, and Signal Processing, Tampa, FL, USA, 26–29 April 1985; Volume 10, pp. 415–418.
20. Tolonen, T.; Karjalainen, M. A computationally efficient multipitch analysis model. *IEEE Trans. Speech Audio Process.* **2000**, *8*, 708–716. [CrossRef]
21. Kraft, S.; Zölzer, U. Polyphonic Pitch Detection by Iterative Analysis of the Autocorrelation Function. Proceedings of the 17th Int. Conference on Digital Audio Effects (DAFx-14), Erlangen, Germany, 1–5 September 2014; pp. 271–278.
22. Klapuri, A. Multipitch analysis of polyphonic music and speech signals using an auditory model. *IEEE Trans. Audio Speech Lang. Process.* **2008**, *16*, 255–266. [CrossRef]
23. Zhang, W.; Chen, Z.; Yin, F. Multi-Pitch Estimation of Polyphonic Music Based on Pseudo Two-Dimensional Spectrum. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2020**, *28*, 2095–2108. [CrossRef]
24. Goto, M. A predominant-F0 estimation method for polyphonic musical audio signals. In Proceedings of the 18th International Congress on Acoustics, Kyoto, Japan, 4–9 April 2004; pp. 1085–1088.
25. Cemgil, A.T.; Kappen, H.J.; Barber, D. A generative model for music transcription. *IEEE Trans. Audio Speech Lang. Process.* **2006**, *14*, 679–694. [CrossRef]
26. Kameoka, H.; Nishimoto, T.; Sagayama, S. A multipitch analyzer based on harmonic temporal structured clustering. *IEEE Trans. Audio Speech Lang. Process.* **2007**, *15*, 982–994. [CrossRef]
27. Emiya, V.; Badeau, R.; David, B. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Trans. Audio Speech Lang. Process.* **2009**, *18*, 1643–1654. [CrossRef]
28. Duan, Z.; Pardo, B.; Zhang, C. Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions. *IEEE Trans. Audio Speech Lang. Process.* **2010**, *18*, 2121–2133. [CrossRef]
29. Peeling, P.H.; Godsill, S.J. Multiple pitch estimation using non-homogeneous Poisson processes. *IEEE J. Sel. Top. Signal Process.* **2011**, *5*, 1133–1143. [CrossRef]
30. Koretz, A.; Tabrikian, J. Maximum a posteriori probability multiple-pitch tracking using the harmonic model. *IEEE Trans. Audio Speech Lang. Process.* **2011**, *19*, 2210–2221. [CrossRef]
31. Yoshii, K.; Goto, M. A nonparametric Bayesian multipitch analyzer based on infinite latent harmonic allocation. *IEEE Trans. Audio Speech Lang. Process.* **2011**, *20*, 717–730. [CrossRef]
32. Vincent, E.; Bertin, N.; Badeau, R. Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Trans. Audio Speech Lang. Process.* **2009**, *18*, 528–537. [CrossRef]
33. Benetos, E.; Dixon, S. A shift-invariant latent variable model for automatic music transcription. *Comput. Music J.* **2012**, *36*, 81–94. [CrossRef]
34. Benetos, E.; Cherla, S.; Weyde, T. An Efficient Shift-Invariant Model for Polyphonic Music Transcription. In Proceedings of the MML 2013: 6th International Workshop on Machine Learning and Music, ECML/PKDD, Prague, Czech Republic, 23 September 2013.
35. O'Hanlon, K.; Nagano, H.; Plumbley, M.D. Structured sparsity for automatic music transcription. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 25–30 March 2012; pp. 441–444.
36. Keriven, N.; O'Hanlon, K.; Plumbley, M.D. Structured sparsity using backwards elimination for automatic music transcription. In Proceedings of the 2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP), Southampton, UK, 22–25 September 2013; pp. 1–6.
37. Fuentes, B.; Badeau, R.; Richard, G. Harmonic adaptive latent component analysis of audio and application to music transcription. *IEEE Trans. Audio Speech Lang. Process.* **2013**, *21*, 1854–1866. [CrossRef]

38. Marolt, M. A connectionist approach to automatic transcription of polyphonic piano music. *IEEE Trans. Multimed.* **2004**, *6*, 439–449. [CrossRef]
39. Sigtia, S.; Benetos, E.; Dixon, S. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2016**, *24*, 927–939. [CrossRef]
40. Böck, S.; Schedl, M. Polyphonic piano note transcription with recurrent neural networks. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 25–30 March 2012; pp. 121–124.
41. Garcia, G. *A Genetic Search Technique for Polyphonic Pitch Detection*; ICMC: Geneva, Switzerland, 2001.
42. Lu, D. Automatic Music Transcription Using Genetic Algorithms and Electronic Synthesis. Bachelor's Thesis, University of Rochester, Rochester, NY, USA, 2006.
43. MIDI Manufacturers Association. *The Complete MIDI 1.0 Detailed Specification*; The MIDI Manufacturers Association: Los Angeles, CA, USA, 1996.
44. Reis, G.; Fonseca, N.; Ferndandez, F. Genetic algorithm approach to polyphonic music transcription. In Proceedings of the 2007 IEEE International Symposium on Intelligent Signal Processing, Alcala de Henares, Spain, 3–5 October 2007; pp. 1–6.
45. Reis, G.; De Vega, F.F.; Ferreira, A. Automatic transcription of polyphonic piano music using genetic algorithms, adaptive spectral envelope modeling, and dynamic noise level estimation. *IEEE Trans. Audio Speech Lang. Process.* **2012**, *20*, 2313–2328. [CrossRef]
46. Inácio, T.; Miragaia, R.; Reis, G.; Grilo, C.; Fernandéz, F. Cartesian genetic programming applied to pitch estimation of piano notes. In Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI), Athens, Greece, 6–9 December 2016; pp. 1–7.
47. Miller, J.F.; Thomson, P. Cartesian genetic programming. In *Genetic Programming*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 121–132.
48. Miragaia, R.; Reis, G.; Fernandéz, F.; Inácio, T.; Grilo, C. CGP4Matlab-A Cartesian Genetic Programming MATLAB Toolbox for Audio and Image Processing. In *International Conference on the Applications of Evolutionary Computation*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 455–471.
49. Miragaia, R.; Reis, G.; de Vega, F.F.; Chávez, F. Multi Pitch Estimation of Piano Music using Cartesian Genetic Programming with Spectral Harmonic Mask. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, Australia, 1–4 December 2020; pp. 1800–1807.
50. Martins, L.G. A Computational Framework for Sound Segregation Music Signals. Ph.D. Thesis, University of Porto, Porto, Portugal, 2008.
51. Rao, K.S.; Prasanna, S.R.M.; Yegnanarayana, B. Determination of Instants of Significant Excitation in Speech Using Hilbert Envelope and Group Delay Function. *IEEE Signal Process. Lett.* **2007**, *14*, 762–765. [CrossRef]
52. Koza, J.R. *Genetic Programming: on the Programming of Computers by Means of Natural Selection*; MIT Press: Cambridge, MA, USA, 1992; Volume 1.
53. Koza, J.R. *Genetic Programming II: Automatic Discovery of Reusable Subprograms*; MIT Press: Cambridge, MA, USA, 1994.
54. Miller, J.F. Gecco 2013 tutorial: Cartesian genetic programming. In Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, Amsterdam, The Netherlands, 6–10 July 2013; pp. 715–740.
55. Shorten, C.; Khoshgoftaar, T.M. A survey on image data augmentation for deep learning. *J. Big Data* **2019**, *6*, 60. [CrossRef]
56. Emiya, V.; Bertin, N.; David, B.; Badeau, R. Maps-A Piano Database for Multipitch Estimation and Automatic Transcription of Music. Research Report, 2010. Available online: https://hal.inria.fr/inria-00544155 (accessed on 21 March 2021).
57. Noll, A.M.; Schroeder, M.R. Real Time Cepstrum Analyzer. U.S. Patent 3,566,035, 23 February 1971.
58. Hansen, N.; Arnold, D.V.; Auger, A. Evolution Strategies. In *Springer Handbook of Computational Intelligence*; Kacprzyk, J., Pedrycz, W., Eds.; Springer: Berlin/Heidelberg, Germany, 2015; pp. 871–898. [CrossRef]
59. Goldman, B.W.; Punch, W.F. Analysis of cartesian genetic programming's evolutionary mechanisms. *IEEE Trans. Evol. Comput.* **2015**, *19*, 359–373. [CrossRef]
60. Miragaia, R.; Reis, G.; de Vega, F.F.; Chávez, F. Evolving a Multi-Classifier System with Cartesian Genetic Programming for Multi-Pitch Estimation of Polyphonic Piano Music. In Proceedings of the 36th ACM/SIGAPP Symposium Om Applied Computing, Gwangju, Korea, 22–26 March 2021.
61. Kehling, C.; Abeßer, J.; Dittmar, C.; Schuller, G. *Automatic Tablature Transcription of Electric Guitar Recordings by Estimation of Score-and Instrument-Related Parameters*; DAFx: Erlangen, Germany, 2014; pp. 219–226.