Automatic Generation of Chord Progressions with an Artificial Immune System

María Navarro[1], Marcelo Caetano[2], Gilberto Bernardes[2], Leandro Nunes de Castro[3], and Juan Manuel Corchado[1]

[1] Department of Computer Science - University of Salamanca Plaza de la Merced s/n, 37008, Salamanca, Spain
maria90@usal.es; corchado@usal.es
[2] INESC TEC - Rua Doutor Roberto Frias 378, 4200-465 Porto, Portugal
mcaetano@inesctec.pt; gba@inesctec.pt
[3] Natural Computing Laboratory, Graduate Program in Computing and Electrical Engineering, Mackenzie Presbyterian University, São Paulo, SP, Brazil
lnunes@mackenzie.br

Abstract. Chord progressions are widely used in music. The automatic generation of chord progressions can be challenging because it depends on many factors, such as the musical context, personal preference, and aesthetic choices. In this work, we propose a penalty function that encodes musical rules to automatically generate chord progressions. Then we use an artificial immune system (AIS) to minimize the penalty function when proposing candidates for the next chord in a sequence. The AIS is capable of finding multiple optima in parallel, resulting in several different chords as appropriate candidates. We performed a listening test to evaluate the chords subjectively and validate the penalty function. We found that chords with a low penalty value were considered better candidates than chords with higher penalty values.
Key words: Artificial Immune Systems, Chord Progressions, Harmony, Consonance.

1 Introduction

Harmony plays a central role in Western tonal music. Simply put, harmony refers to the simultaneity of pitch (i.e., chords) and their progressions, known as chord progressions. Chord construction and chord progression are governed by implicit and explicit principles which are central in the study of harmony. Schönberg [1], Lerdahl [2], Riemann [3], and Schenker [4], among many others, have discussed these principles extensively and proposed rules to create optimal chord progressions according to the principles considered.

In music composition, creating chord progressions commonly requires knowledge usually acquired after years of music training. Not surprisingly, chord progressions have been a central topic in algorithmic composition given the challenging aspect of encoding the principles to generate a desired result. There have been several proposals to automatically generate chord progressions following different paradigms [5], including grammars, learning, biological principles, and rules

(*a priori* knowledge). CHORAL [6] is a system that harmonizes chorales in the style of Johann Sebastian Bach. The system is based on grammars and contains about 350 rules representing musical knowledge from multiple viewpoints of the chorale, such as the chord skeleton, the melodic lines of the individual parts, and Schenkerian voice leading. Steedman [7] presents a small number of rules to generate chord sequences using generative grammars. Eigenfeldt [8] proposes the generation of harmonic progressions by learning using case-based analysis of an existing material and employing a variable-order Markov model. Moroni [9] has developed a system called Vox Populi, based on evolutionary computation techniques for interactive algorithmic composition. In Vox Populi, a population of chords evolves through the application of a genetic algorithm to maximize a fitness criterion based on musically relevant factors. Anders [10] developed a computational model that creates chord progression following the rules that Schönberg proposed in his harmony treatise [1]. Paiement [11] adopts a probabilistic approach to model chord progressions. Fukumoto [12] generates chord progressions suited for user's feeling by using genetic algorithms.

In this work, we describe a method to automatically generate chord progressions with an artificial immune system (AIS). We propose a penalty function that encodes rules about chord construction as vertical constraints and chord progression as horizontal constraints. Then we use an AIS to find chords that minimize the penalty function and propose the next chord in a sequence as a minimum-penalty chord given the sequence as input. The AIS used is opt-aiNet [13], an algorithm inspired by the immune network theory for function optimization. Opt-aiNet is capable of finding multiple optima in parallel upon convergence, resulting in several chords as candidates for the next in the sequence. Thus we performed a listening test to evaluate the candidate chords and validate the penalty function.

In the next section, we describe how the penalty function encodes the rules we use to automatically generate chord progressions as vertical and horizontal constraints. Section 4 explains how to encode the chords and the constraints, followed by how to find minima of the penalty function with the AIS. The last section shows some results of the system, a discussion about evaluation of these new chords, chord progressions obtained and future work proposed.

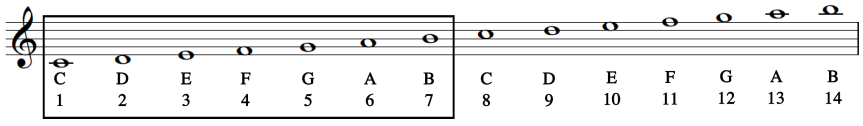

Fig. 1. Encoding diatonic scales. The figure shows the musical notation for the notes with letter names and the corresponding codification in the algorithm below each note. The C major scale is highlighted in the image.

2 Representing Individual Chords and Progressions

The aim of this work is to generate chord progressions as sequences of three-note chords. Each chord X is represented as a vector with three integers $X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$, where $[x_1, x_2, x_3]$ define each note in the chord. Fig. 1 shows the two octaves of the diatonic (C major) scale used to construct the chords, where each note is encoded as an integer. For example, C major gives $X = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$. Notice that the system can easily work with any other major or minor scale by simply adapting the representation. A sequence of chords is thus represented as $[X_1, X_2, \ldots, X_n]$, where the subscript is the position in the sequence and n is the current position.

3 Evaluating Chord Progressions with a Penalty Function

The aim of the penalty function P is to automatically evaluate a chord X_n given a sequence of p previous chords $[X_{n-1}, X_{n-2}, \ldots, X_{n-p}]$. For $p = 2$, the penalty function becomes $P\left(X_n/X_{n-1}, X_{n-2}\right)$, interpreted as penalty of chord X_n following $[X_{n-1}, X_{n-2}]$. The penalty function $P\left(X_n/X_{n-1}, X_{n-2}\right)$ shown in Eq. (1) encodes rules about chord construction as vertical constraints $V\left(X_n\right)$ and chord progression as horizontal constraints $H\left(X_n/X_{n-1}, X_{n-2}\right)$. In turn, $V\left(X_n\right)$ and $H\left(X_n/X_{n-1}, X_{n-2}\right)$ penalize chords by adding a different penalty value depending on the rules encoded. The final value of $P\left(X_n/X_{n-1}, X_{n-2}\right)$ is the sum of all penalties and thus represents how appropriate chord X_n is as a candidate to follow $[X_{n-1}, X_{n-2}]$. As such, the aim is to find chords that minimize the penalty function. Next we explain the vertical and horizontal constraints and how to assign penalties to chords using $V\left(X_n\right)$ and $H\left(X_n/X_{n-1}, X_{n-2}\right)$.

$$P\left(X_n/X_{n-1}, X_{n-2}\right) = V\left(X_n\right) + H\left(X_n/X_{n-1}, X_{n-2}\right) \tag{1}$$

3.1 Vertical Constraints

The vertical constraints $V\left(X_n\right)$ are related to chord construction, so they are only applied to notes that belong to the candidate chord X_n. The aim of the vertical constraints is to minimize both chord dissonance and balance the distance between the notes in the chord. The measure of consonance is based on the Pythagorean theory [14], which orders complementary intervals as listed below.

1. Unison and octave
2. Perfect fourth and perfect fifth
3. Major third and minor sixth
4. Minor third and major sixth
5. Minor second and major seventh
6. Augmented fourth and diminished fifth

The vertical constraints are the following

- V1: Maximize chord consonance
- V2: Favor triad chords to reinforce progression consonance

Table 1. Table showing dissonances between n1 and n2

$d(x_1 - x_j)$	1	2	3	4	5	6	7	8	9	10	11	12
Interval	m2	M2	m3	M3	P4	dim5	P5	m6	M6	m7	M7	P8
Pythagorean Ratio	$\frac{16}{15}$	$\frac{9}{8}$	$\frac{6}{5}$	$\frac{5}{4}$	$\frac{4}{3}$	$\frac{7}{5}$	$\frac{3}{2}$	$\frac{8}{5}$	$\frac{5}{3}$	$\frac{7}{4}$	$\frac{15}{8}$	$\frac{2}{1}$
$k\left[d\left(x_1, x_i\right)\right]$	240	72	30	20	12	35	6	40	15	28	120	2

3.2 Enconding the Vertical Constraints in $V\left(X_n\right)$

$V\left(X_n\right)$ encodes the vertical constraints as shown in Eq. (2). This function consists of two parts, $V_1\left(X_n\right)$ encodes rule V1 and $V_2\left(X_n\right)$ encodes rule V2.

$$V\left(X_n\right) = V_1\left(X_n\right) + V_2\left(X_n\right) \tag{2}$$

$V_1\left(X_n\right)$ measures the level of consonance of each chord as

$$V_1\left(X\right) = ln\left[\sum_{i=2}^{N} k\left[d\left(x_1, x_i\right)\right]\right] \tag{3}$$

where N is the number of notes in the present chord (in this case $N = 3$, x_1 is the first note in the present chord, x_i is the i^{th} note in the present chord, $d\left(x_1, x_i\right)$ is the distance in half-steps, and $k\left[d\left(x_1, x_i\right)\right]$ is the dissonance between x_1 and x_i as proposed by Euler [15] to evaluate the consonance of intervals using Pythagorean theory. The dissonance k takes the product of the numerator and denominator of the rates shown in Table 1.

As an example, consider the chord of C major $X = \left[\begin{smallmatrix}1\\3\\5\end{smallmatrix}\right]$. The intervals considered are always obtained from the fundamental to the rest of the notes. In this case, the consonance measure is calculated as follows in Eq. (4).

$$\begin{aligned} V_1\left(X\right) &= ln(k(d(x_1 - x_2)) + k(d(x_1 - x_3))) = ln(k(d(1-3)) + k(d(1-5))) = \\ &= ln(k(d(2)) + k(d(4))) = ln(k(4) + k(7)) = ln(20 + 6) = ln(26) = 3.26 \end{aligned} \tag{4}$$

Likewise, $V_2\left(X_n\right)$ checks if the chord is part of a triad-chord (three-note chords) family [1]. $TRIAD\_SET$ is a set with the triad chords in major mode, where n is the number of elements of X_n contained in $TRIAD\_SET$.

$$V_2\left(X\right) = 2 * n(X \subseteq TRIAD\_SET) \tag{5}$$

For example, $V_2\left(\left[\begin{smallmatrix}1\\3\\5\end{smallmatrix}\right]\right) = 0$ because C major belongs to the triad-chord.

3.3 Horizontal Constraints

The horizontal constraints are related to chord progressions, so they compare the candidate chord X_n with previous chords $[X_{n-1}, X_{n-2}]$ taking into account functional (tonal) harmony and voice leading. Following the works of Schönberg [1] and Riemann [3], we considered the following horizontal constraints.

- H1: Reward pre-determined harmonic progression
- H2: Avoid chord repetition
- H3: Avoid constant use of *superstrong* progression
- H4: Minimize distance between voices
- H5: Resolve leading-note
- H6: Avoid parallel fifths and octaves

H1, H2 and H3 are related to harmonic functions or the role a chord has within the context of a specific key. In the context of this work, we reward pre-determined harmonic progressions based on the functional theory of Riemann [3]. Figure 2 illustrates the harmonic progressions according to Riemann's functional harmony. Thus the rules we adopt reward the progression Tonic-Subdominant-Dominant-Tonic (H1) and penalize the repetition of the same chord (H2). Finally, H3 was inspired by Schönberg's [1] harmonic treatise.

H4, H5, and H6 are related to voice leading, namely, the horizontal progression of the individual voices. In this work, the voices are the notes in the chord. The guiding principle aims to minimize the distance between consecutive voices (H4). We further stress the importance of resolving the leading tone in H5 and avoiding parallel fifths and octaves in H5. The leading-note creates a temporary instability that requires melodic resolution to a stable tone [16]. For example, in C major, the seventh scale degree (B) has a strong melodic tendency towards the first degree (C) [17]. We chose to additionally penalize parallel fifths and octaves because, according to traditional Western musical practice, they result in weak relative motion between chords.

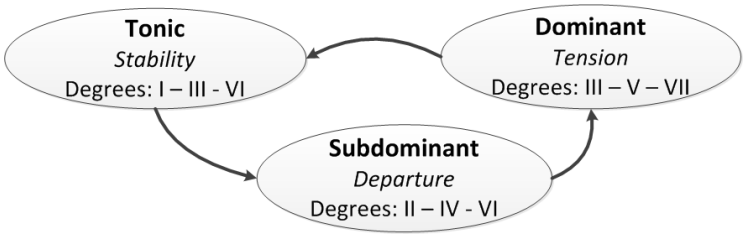

Fig. 2. Relationship between the three main harmonic functions in a tonality. Each ellipse gives information about the name and the meaning of each function. Additionally the degrees corresponding to each function are shown. It is noteworthy that some chords have different function depending on its previous chord in the progression.

Table 2. Table showing the values of $H_1\left(X_n/X_{n-1}, X_{n-2}\right)$. First row reflects the function of the previous chord, whereas first column represents the function of the evaluating chord X_n.

	Tonic	Subdominant	Dominant
Tonic	2	0	5
Subdominant	5	2	0
Dominant	0	5	2

3.4 Encoding the Horizontal Constraints in $H\left(X_n/X_{n-1}, X_{n-2}\right)$

Eq. (6) shows that $H\left(X_n/X_{n-1}, X_{n-2}\right)$ encodes the horizontal constraints in six terms, each corresponding to one horizontal constraint H.

$$\begin{aligned} H\left(X_n/X_{n-1}, X_{n-2}\right) = H_1\left(X_n/X_{n-1}, X_{n-2}\right) + H_2\left(X_n/X_{n-1}, X_{n-2}\right) + \\ + H_3\left(X_n/X_{n-1}\right) + H_4\left(X_n/X_{n-1}\right) + H_5\left(X_n/X_{n-1}\right) + H_6\left(X_n/X_{n-1}\right) \end{aligned} \tag{6}$$

Notice that H_1 and H_2 depend on both X_{n-1}, X_{n-2} whereas the others depend only on X_{n-1}. $H_1\left(X_n/X_{n-1}, X_{n-2}\right)$ measures how the progression is adapted to the predefined harmonic function structure given by Tonic-Subdominant-Dominant-Tonic. In this case, we need X_{n-1} and X_{n-2}, the two last chords of the sequence, given as the input. The weights proposed are shown in Table 2.

For example, consider $X_{n-1} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$(tonic) as input and $X_n = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$ (subdominant) proposed by the system. X_n obtains a penalty value of 0, because tonic is followed by subdominant.

Eq. (7) shows that chord repetitions receive a penalty of 2 for each consecutive repetition of the chord. In particular, only the previous two chords of the progression are considered to apply Eq. (7).

$$\begin{aligned} H_2\left(X_n/X_{n-1}, X_{n-2}\right) = 2 \quad \text{if } X = Y \\ H_2\left(X_n/X_{n-1}, X_{n-2}\right) = 0 \quad \text{otherwise} \end{aligned} \tag{7}$$

In the previous example, the input $X_{n-1} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$ followed by the same chord $X_n = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$ is penalized with two points.

Inspired by Schönberg's treatise [1], the *superstrong* progression occurs when the root of the second chord is a second step up or down. This kind of progression is not frequently used, but this does not mean the progression should be avoided. Thus Eq. (8) penalizes the *superstrong* progression with 0.5, where X_n is the present chord and X_{n-1} is the previous chord of the input progression.

$$\begin{aligned} H_3\left(X_n/X_{n-1}\right) = 0.5 \quad \text{if } X_n \cap X_{n-1} = \emptyset \\ H_3\left(X_n/X_{n-1}\right) = 0 \quad \text{otherwise} \end{aligned} \tag{8}$$

For example, considering input $X_{n-1} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$, a *superstrong* cadence appears if the next chord has not any common notes with it, i. e., $X_n = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$.

$H_4\left(X_n/X_{n-1}\right)$ is the distance between the X_n array and the previous chord X_{n-1} in the progression. This is related to rule H4, and measures the "voice leading" of the progression.

We were looking for a way to penalize larger intervals against shorter intervals when comparing one chord option with another in the same input progression. For this reason, we used an exponential function. In order to smooth the value obtained, a logarithmic function was added, as we can see in Eq. (9).

$$H_4\left(X_n/X_{n-1}\right) = ln\left(\sum_{i=1}^{p} 4^{|\left(x_{ij} - x_{i(j-1)}\right)|}\right) \tag{9}$$

where i refers to the note of a given chord, j refers the element of the chord sequence, and p is the maximum number of voices (in this particular case, $p = 3$).

For example, the input $X_{n-1} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$ followed by $X_n = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$ has the following distance measure:

$$\begin{gathered} H_4(X_n/X_{n-1}) = ln(4^{|x_{12}-x_{11}|} + 4^{|x_{22}-y_{21}|} + 4^{|x_{32}-y_{31}|} = \\ ln(4^{|1-2|} + 4^{|3-4|} + 4^{|5-6|}) = ln(4+4+4) = ln(12) \end{gathered} \tag{10}$$

$H_5\left(X_n/X_{n-1}\right)$ is a melody penalty that considers the parallel fifths and octaves (rule H5). If a parallel fifth or octave appears, the sequence is given a penalty value of 3 as shown in Eq. (11).

$$\begin{gathered} H_5\left(X_n/X_{n-1}\right) = 3 \quad \text{if } |x_{ij} - x_{(i-1)j}| + 1 = 5 \wedge |x_{i(j-1)} - x_{(i-1)(j-1)}| + 1 = 5 \\ H_5\left(X_n/X_{n-1}\right) = 3 \quad \text{if } |x_{ij} - x_{(i-1)(j-1)}| + 1 = 8 \wedge |x_{i(j-1)} - y_{(i-1)(j-1)}| + 1 = 8 \\ H_5\left(X_n/X_{n-1}\right) = 0 \quad \text{otherwise} \end{gathered} \tag{11}$$

There is a paralell fifth between $X_{n-1} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$ and $X_n = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$, because $|x_{32} - x_{22}| + 1 = |1 - 5| + 1 = 4 + 1 = 5$ and $x_{31} - x_{21} + 1 = |2 - 6| + 1 = 4 + 1 = 5$. Thus X_n receives a penalty of 3.

Finally, the last element in Eq. (6), $H_6\left(X_n/X_{n-1}\right)$, is a melody penalty that takes into account the leading-tone resolution (H6). If the leading-note resolution rule is not accomplished, the penalty is of 2.5 as shown in Eq. (12).

$$\begin{gathered} H_6\left(X_n/X_{n-1}\right) = 2.5 \quad \text{if } hf\left(X_n\right) = Tonic \wedge x_{i(j-1)} = 7 \wedge x_{ij} \neq 8 \\ H_6\left(X_n/X_{n-1}\right) = 0 \quad \text{otherwise} \end{gathered} \tag{12}$$

where x_{ij} represents the i^{th} note of the j^{th} chord in the sequence. The term $hf\left(X_n\right)$ calculates the harmonic function that the chord X_n represents in the moment j. $hf\left(X\right)$ can take three values: *Tonic*, *Subdominant* and *Dominant*.

Note that the penalty values proposed in functions H_5 and H_6 have been obtained empirically.

4 Chord Progressions as Minima of the Penalty Function

Given a sequence of two previous chords as reference, it is possible to associate a penalty value to any three-note chord representable by a vector of three integers between 1 and 16. For instance, the chord $X_n = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$ preceded by $X_{n-2,n-1} \begin{bmatrix} 2 & 1 \\ 3 & 5 \\ 7 & 8 \end{bmatrix}$ has a penalty of 12.79. Thus $P\left(X_n / X_{n-1}\right)$ can be used to propose the next chord in a sequence using the penalty value as measure. According to the vertical and horizontal constraints encoded in the penalty function, chords with low penalty values should be better candidates to follow a given sequence of two chords than chords with higher penalty values. Thus the problem of automatically proposing the next chord in a given chord sequence $[X_1, X_2, \ldots, X_n]$ becomes simply finding the chords that correspond to the minima of the penalty function with $[X_1, X_2, \ldots, X_n]$ as input. The method can be iteratively applied to add a new chord to the sequence at each iteration, such that the automatic generation of chord progressions becomes the search for the minima of the penalty function.

Finding chords with low penalty values can be a difficult problem. An exhaustive (or brute-force) search would require testing every possible three-note chord representable to determine which has the lowest penalty value. Thus we use an optimization method to search for a minimum of the penalty function at each iteration. Similarly to the approach adopted in Vox Populi [9], the generation of chord progressions can be considered as a search problem in which the constraints must be followed so as to explore the conceptual space of possible solutions. However, the penalty function can potentially have several minima, each corresponding to a different chord. These minima will have different penalty values associated, such that the global minimum of the penalty function might not be the best candidate in a particular musical context. Therefore, we use opt-aiNet [18] to minimize the penalty function due to its ability to find several minima in parallel.

This algorithm, inspired in natural immune systems [19], assumes a randomly initialized set of immune cells or antibodies (in our particular case, each antibody is a three-note chord) in the network. Their affinity is determined using a distance metric, in this case, the penalty function. Some high affinity antibodies are selected and reproduced based on their affinity: the higher the affinity, the higher the number of clones and vice-versa. The clones generated suffer a mutation inversely proportional to their affinity. Those antibodies whose affinity is less than a given threshold are eliminated from the network. Finally, a number of newly generated antibodies are incorporated into the network.

As an example, we created a short sequence of two chords in C major as input, and we had the AIS generate multiple options for the third chord in the sequence. The input chords sequence is [1 3 5]([60 64 67] in MIDI mode). After convergence, 42 chords were presented as candidates to follow the input sequence. Each of these chords corresponds to a minimum of the penalty function, but they all have different penalty values associated. In our implementation, we have observed that the AIS typically results in over 30 chords with penalty values ranging from 8 to 22. It is impractical to listen to so many chords every time we

want to add a new chord to a given sequence. Thus we decided to investigate if the penalty values can be used to further inform us about the quality of the chords to follow the sequence.

5 Evaluation

We conducted a preliminary study into the relationship between the penalty function and the subjective evaluation of chords in a sequence. The evaluation aims to investigate if chords with low penalty values are judged more appropriate to follow a given chord sequence than chords with higher penalty values.

We created a short sequence of two chords in C major as input, and we had the AIS to generate multiple options for the third chord in the sequence. The input chords sequence is [60 64 67, 62 67 71] After convergence, 35 chords were presented as candidates to follow the input sequence, with penalty values ranging from 8.32 to 22.26. We ranked the chords by penalty values and selected 17 chords to represent the whole range of values between the minimum and the maximum. We asked participants to listen to the three-chord progressions and evaluate how well the third chord follows the first and second using the following scale: very good (1 point), good (0.75 points), fair (0.5 points), bad (0.25 points), or very bad (0 points). The listening test can be found here: `http://goo.gl/forms/flTb4PbZY6`.

We expected chords with low penalty values to be considered better candidates to follow the input sequence than chords with higher penalty values. In total, 24 people took the test, among which 8 declared no musical training, 7 considered themselves amateurs, and 9 professional musicians. In figure 3 we see the subjective evaluation as a function of the penalty value for each third chord that they heard.

The aim of the evaluation is twofold, we would like to validate the penalty function and investigate if there is a threshold value that can be used as decision boundary for chord quality. The validation of the penalty function involves investigating whether chords with low penalty are considered better candidates to follow a given sequence than chords with higher penalty values. We also want to automatically determine the quality of a chord as candidate to follow a sequence. Ideally, we want to be able to use the penalty value as decision boundary. Thus chords with penalty value lower than a certain threshold would correspond to good candidates.

Figure 3 shows the result of the listening test as the subjective evaluation of the penalty function. In figure 3, we see the mean and standard deviation of the subjective rating for each candidate chord plotted against the corresponding penalty value. The horizontal lines labeled *very bad*, *bad*, *fair*, *good* and *very good* can be used as reference to interpret the figure. The chords with lower penalty value were indeed rated better chords than those with lower penalty values. Using the line labeled *fair* as decision boundary, we can associate penalty values lower than 0.5 with chords that were rated positively.

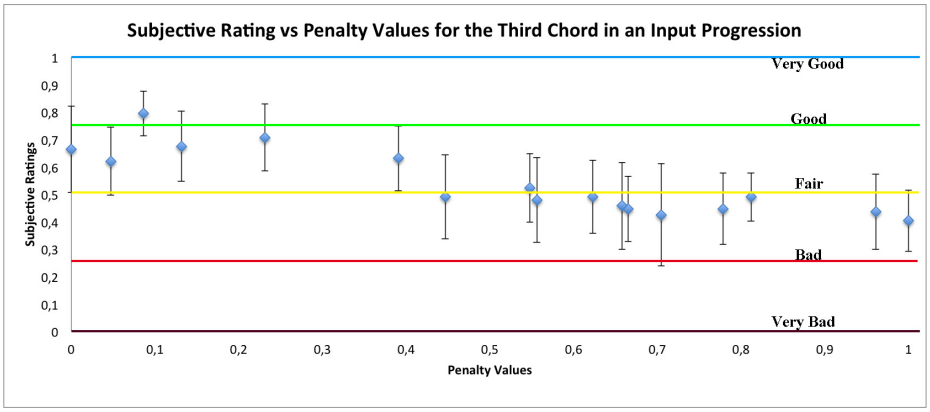

Fig. 3. Plot of the subjective evaluation as a function of the penalty value. Each point represents the evaluation of the third chord created by the system, considering the chord progression given as the input. The horizontal axis is the normalized penalty value and the vertical axis is the mean of the scores the listeners gave. The standard deviation is shown for each point by using vertical bars. The lines show the values corresponding to the semantic labels.

6 Discussion

The main goal of the evaluation is to validate the penalty function proposed to to create a compositional aid application. The penalties associated with vertical and horizontal constraints are selected empirically, testing the system to balance all the rules. The penalty values seem to be inversely proportional to how well people judged the chords proposed by the AIS following a given sequence. This indicates that we might be able to determine suitable chords using the penalty function. However, the threshold depends on the penalties associated with the constraints and changing these would probably change the threshold values.

More importantly, the listening test asked participants to evaluate how well a given chord proposed by the AIS follows two others in a sequence. This question does not specify whether to consider vertical or horizontal aspects. In other words, a chord proposed by the system might be rated *very good* because it is very pleasant independently of the two previous chords heard. Some participants reported using different criteria to rate how well the chords follow the sequence. This question seems important to pursue in future work.

7 Conclusions

In this work, we proposed a penalty function using rules about chord construction and chord progression. These constraints apply rules from tonal Western music and functional harmony as vertical and horizontal constraints. Then we apply an artificial immune system (AIS) to automatically generate the next

chord in a sequence taking two previous chords as input. The AIS is capable of finding multiple optima in parallel, resulting in different chords as appropriate candidates.

We performed a listening test to evaluate the chords subjectively and validate the penalty function. We found that chords with a low penalty value were considered better candidates than chords with higher penalty values. We determined that there is threshold value in the penalty function associated with the subjective evaluation. Chords with associated penalty value above this threshold value were considered as good chords to follow a given chord progression, whereas chords below this value were rated negatively

Future work can integrate the system into an application to assist users in composing chord progressions. In addition, we can develop a Markov Model to decide the harmonic function of the chord that we can propose, so that the system will be capable of automatically composing a chord progression given only one initial chord. The chords can be encoded using MIDI notation to enable the integration of several musical systems without the need to rewrite the code.

Acknowledgements

This work has been partially supported by the Spanish Government through the project iHAS (grant TIN2012-36586-C01/C02/C03), the Media Arts and Technologies project (MAT), NORTE-07-0124-FEDER-000061, financed by the North Portugal Regional Operational Programme (ON.2 - O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by national funds, through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT), and the Mackenzie University, Mackpesquisa, CNPq, Capes (Proc. n. 9315/13-6) and FAPESP.

References

1. Schoenberg, A.: The Musical Idea and the Logic, Technique and Art of its Presentation. Indiana University Press (2006)
2. Lerdahl, F.: Tonal pitch space. Music Perception (1988) 315–349
3. Agmon, E.: Functional harmony revisited: A prototype-theoretic approach. Music Theory Spectrum **17** (1995) 196–214
4. Stock, J.: The application of schenkerian analysis to ethnomusicology: Problems and possibilities. Music Analysis (1993) 215–240
5. Papadopoulos, G., Wiggins, G.: AI methods for algorithmic composition: A survey, a critical view and future prospects. In: AISB Symposium on Musical Creativity, Edinburgh, UK (1999) 110–117
6. Ebciouglu, K.: An expert system for harmonizing chorales in the style of js bach. The Journal of Logic Programming **8** (1990) 145–185
7. Steedman, M.J.: A generative grammar for jazz chord sequences. Music Perception **2** (1984) 52–77

8. Eigenfeldt, A., Pasquier, P.: Realtime generation of harmonic orogressions using controlled markov selection. In: Proc. of 1st Int. Conf. on Computational Creativity. (2010) 16–25
9. Moroni, A., Manzolli, J., Von Zuben, F., Gudwin, R.: Vox populi: An interactive evolutionary system for algorithmic music composition. Leonardo Music Journal **10** (2000) 49–54
10. Anders, T., Miranda, E.R.: A computational model that generalises schoenberg's guidelines for favourable chord progressions. In: 6th Sound and Music Computing Conference, Porto, Portugal (2009)
11. Paiement, J.F., Eck, D., Bengio, S.: A probabilistic model for chord progressions. In: Proceedings of International Conference on Music Information Retrieval. (2005) 312–319
12. Fukumoto, M.: Creation of music chord progression suited for user's feelings based on interactive genetic algorithm. In: Advanced Applied Informatics (IIAIAAI), 2014 IIAI 3rd International Conference on, IEEE (2014) 757–762
13. De Castro, L.N., Timmis, J.: Artificial Immune Systems: A new Computational Intelligence Approach. Springer (2002)
14. Crocker, R.L.: Pythagorean mathematics and music. Journal of Aesthetics and Art Criticism (1964) 325–335
15. Knobloch, E.: Euler transgressing limits: the infinite and music theory. Quaderns d'història de l'enginyeria **9** (2008)
16. Babbitt, M.: The structure and function of musical theory: I. In: College Music Symposium, JSTOR (1965) 49–60
17. Benward, B., Saker, M.: Music in Theory and Practice. London, England: McGraw-Hill (2003)
18. de Castro, L.N., Timmis, J.: An artificial immune network for multimodal function optimization. In: Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on. Volume 1., IEEE (2002) 699–704
19. Murphy, K.: Janeway's Immunobiology. Garland Science (2011)