Bassline Composition for Alternative Rock using SP and SDS algorithms, a multi-objective approach

A.F. Zarta
Universidad de los Andes, Bogotá, Colombia

N. Velasco
Universidad de los Andes, Bogotá, Colombia

Esta investigación tiene como propósito solucionar el problema de encontrar un "buen" bajo para acompañar una melodía y una batería, siguiendo la estructura del genero Rock Alternativo. El método de solución consta de varias etapas las cuales incluyen: determinar la tonalidad de la canción utilizando un algoritmo de Matching que compara las notas en la melodía con las notas en cada tonalidad, ajustar acordes a la melodía construyendo un grafo de posibles progresiones y encontrando la ruta mas corta en este, improvisar bajos usando estos acordes como guía utilizando el algoritmo "Stochastic Diffusion Search". Las soluciones deben satisfacer una serie de restricciones mientras se maximiza la calidad del sonido medida usando dos criterios diferentes.

This research proposes a solution to the problem of finding a "good" bassline to accompany a melody and a drumbeat following the structure of the alternative rock genre. The solution method is made up of different steps which include: determining the key of the song by using a matching algorithm that compares the notes in the melody with the notes in each key, fitting chords to the melody by constructing a graph of possible progressions using candidate chords selected through a matching algorithm and finding the shortest path in it, improvising basslines using these chords as guidelines through the use of stochastic diffusion search algorithm. Solutions must satisfy constraints regarding their relationship with the drumbeat and the melody while maximizing the quality of the overall sound measured with two independent criteria.

1 INTRODUCTION

The problem of composing "good" basslines for songs is a task that is far from easy even for skilled musicians. Many aspects should be considered such as the quality of the note progression, the relationship with the melody and the drumbeat, the structure of the genre, among others. Within the alternative rock genre, bass is a key element in any song as it plays an important role in both harmonic and rhythmic sections. More importantly it acts as a bridge between the two sections, enabling the melody and rhythm to blend nicely.

Throughout history the ability of solving such problems has been believed to be a musician-only ability. However, due to the good results that optimization algorithms have showed in problems of a wide range of areas, the topic of musical composition through optimization algorithms has been of interest for researchers, especially during the latter years. This is very important, as these problems have usually been solved through an artistic approach while this research proposes a numeric approach. Nonetheless one can argue that music is intrinsically mathematic.

2 RELATED WORK

Early work in this field can be found in (Laine & Kuuskankare, 1994) in which musical pieces are composed by treating notes as a time series. This research does not take into consideration a specific genre structure, generating relatively bad solutions consequence of the unbounded solution space and differences between genres. With this in mind, the problem in this research is restricted so that solutions will follow the structure of the Alternative Rock genre.

Latter researches define the genre very clearly and a variety of genres have been studied such as Ethiopian Bagana in (Herremans , Weisser, Sorensen, & Conklin, 2015), Medieval Music in (Geem & Choi, 2007), Jazz and Blues in (Kunimatsu, Ishikawa,

Takata, & Joe, 2015) and fifth species counterpoint in (Herremans et al. 2013).

A great collection of solving techniques have been utilized such as Predictive models in (Herremans et al. 2014), in which a classification model is defined to determine if a song will be part of the top ten songs billboard, machine learning in (Dubnov, Assayag, Lartillot, & Bejerano, 2003) and models based on user feedback in both (Lichtenwalter, Zorina, & Chawla, 2008) and (Tojui & Iba, 2000), in which the algorithm generates solutions which are evaluated by the user in order to construct the objective function iteratively.

Although the topic of musical composition through optimization algorithms is common to all researches mentioned, the aim differs vastly form one research to another. In both (Lozano, Medaglia, & Velasco, 2009) and (Ren, Wang, Du, Liu, & Siddiqi, 2015) a chord progression is fitted to a particular melody, in contrast in (López-Ortega & López-Popa, 2012) and (Alfonseca, Cebrián, & Ortega, 2007) musical sections are composed without using any predefined section as basis. Other researches, such as (Dostál, 2005) focus on generation of rhythms and drum-like sequences rather than note-based music. Bassline[1] composition has only been taken into consideration in (Kunimatsu, Ishikawa, Takata, & Joe, 2015) in which chords are fitted into a melody and then a bassline is fitted into these chords generating satisfactory solutions.

3 PROBLEM DESCRIPTION

Similar to traditional Rock bands, Alternative Rock bands are usually made up of a drummer, a bass player, a singer, from one up to three guitarists and a in some occasions a keyboard player. The writing process regularly starts when some musician comes up with a section or riff[2] in his or her instrument and the other members of the band use it as basis for creating complimentary sounds. In this research we will be focusing in the case in which a drumbeat and a melody (produced by voice, guitar or keyboard) have already been defined and it is desired to generate a bass riff or "bassline" that can be effectively incorporated into the existing song. Taking this into consideration, the quality of a bassline will depend on the affinity of its groove[3] and the groove of the melody, the relationship between its notes and the notes of the melody that sound at the same time and the affinity between its groove and the drumbeat.

In order to solve the problem at hand a two-part algorithm was designed, in which chords will be fitted to the melody during the first part and basslines will be improvised using these chords as guidelines during the second part. Roughly, the solution algorithm is structured as follows:

Solution algorithm

Require: M, the melody of the song; D, the drumbeat of the song; CM, cost matrix of passing from one chord to another; NCPB, desired number of chords per bar; DT, desired degree of tension in the chord progression; MM, binary parameter that determines if the progression has a major or minor character; N, number of miners to use in SDS; p, probability of keeping exploring.

1.1 Use a matching algorithm to determine the key of the song.

1.2 Divide the melody M into fragments of equal length depending on the parameter NCPB.

1.3 Use a matching algorithm to define candidate chords that can be fitted to each fragment, creating a graph of possible chord progressions.

1.4 Change the CM depending on the values of DT and MM defined by the user.

1.5 Translate the chords to its diatonic function, by using the key found.

1.6. Use the modified CM to find the shortest path in the graph of possible chord progressions to obtain the best progression.

2.1 Using the key and the chord progression found, start SDS algorithm with N miners and set their statuses to "unpleased".

2.2 Each miner randomly finds a bassline that satisfies the constraints regarding the melody M and the drumbeat K.

2.3 Objective functions values are calculated for each solution and dominant solutions are included in the Pareto front of solutions.

2.4 **If** a solution is part of the Pareto front its respective miner is marked as pleased.

2.5 **If** a miner is unpleased he either continues searching the solution space with probability p or copies a solution of a pleased miner with probability $(1-p)$.

2.6 Return solutions in the Pareto front after convergence of the SDS algorithm.

End

[1] The term bassline refers to a musical line played by a bass

[2] The term riff refers to a musical phrase that is repeated throughout a song.

[3] The groove is the rhythmic feel of a section, given by the note values used in it.

4 STRUCTURE OF THE GENRE

It was previously stated that it is imperative to constraint the solution space to a particular genre. The genre at hand is Alternative Rock and to define its structure a database including songs of artists such as Muse, Kasabian, Red Hot Chili Peppers, Warpaint, among others, was constructed. The database consists of fifty songs and they are used to determine different metrics necessary for the algorithm.

4.1 Metrics

The following metrics serve as quantifiers of different properties of the genre and are used latter on to generate, evaluate and constraint solutions:

4.1.1 Frequency of passing from one chord to another

Table 1 shows the values of $(100 - f)$ of passing from a chord to another. In this case f stands for the frequency with which a chord passes to another given their diatonic functions[4].

Table 1. Cost of passing from chord to another given their diatonic functions

	I	ii	iii	VI	V	Vi	vii
I	100	92,3	96,2	69,5	55	87	100
ii	85,7	100	92	73,4	85,7	63,1	100
iii	85,3	79,7	100	66,7	85,3	83	100
IV	65	91,3	91,3	100	70	82,6	100
V	84,4	85	96,9	68,1	100	65,6	100
vi	80,1	94	90,9	64,5	70,4	100	100
vii	65,5	91,4	86,2	84,5	82,8	89,7	100

4.1.2 Frequency of note values in basslines

The frequency in which a certain note value appears in the basslines of the database of songs is shown in Table 2.

Table 2. Observed Frequency of Note Values

Whole	0%
Half	2,5%
Quarter	10%
Eight	55%
Sixteenth	30%
Thirty-Second	2,5%

4.1.3 Kick Drum / Bass Relationship

It was found that certain behaviors were very common in the songs regarding the relationship between the kick drum and the bassline. Representing the bassline (white) and the kick drum (gray) as squares, in which its length is proportional to the duration of the sound[5], the only patterns found are shown in Fig 1.

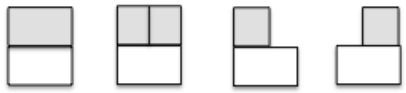

Figure 1. Allowed kick/bass patterns

In contrast, Fig 2 shows the patterns that never showed up.

Figure 2. Banned kick/bass patterns

4.1.4 Upper bound for musical intervals

The musical interval between each pair of notes that overlap[6] was determined. More importantly the proportion in which every interval appears within each song was found. This allows us to determine upper bounds for each interval. The upper bound is defined as the mean plus two standard deviations because it resulted in values consistent with the maximum proportions found. Table 3 shows explicitly the information mentioned.

[4] The diatonic function of a chord represents the degree of tension of the chord in relation to the key.

[5] A square can represent any note value, the important thing to notice is that if a square has more length than other, it also has a smaller note value.

[6] The pair is composed of one note from the melody and one note from the bassline. Overlap means that they sound at the same time.

Table 3. Upper bound for musical intervals

Unison	51,2%
Minor Second	3,9%
Major Second	26,9%
Minor Third	18,2%
Major Third	16,7%
Perfect Fourth	41,0%
Tritone	6,9%
Perfect Fifth	36,2%
Minor Sixth	23,5%
Major Sixth	41,3%
Minor Seventh	20,4%
Major Seventh	11,4%

4.1.4 Score of using a note in the bassline

Table 4 shows the score given to the notes when they are included in the bassline during a section that corresponds to a certain chord given its diatonic function.

Table 4. Score for the musical interval between a note and the root of the chord ringing given the diatonic function

	I	ii	iii	IV	V	vi	vii
Unison	5	5	2	5	5	5	0
Minor Second	0	-2	0	-2	-2	0	0
Major Second	0	-2	-2	0	2	1	0
Minor Third	-2	1,5	0	-2	-2	2	0
Major Third	2	-2	-2	3	0	-2	0
Perfect Fourth	0	-2	2	-2	1	2	0
Tritone	-2	-2	-2	1	-2	-2	0
Perfect Fifth	2	3	1	3	2	3,5	0
Minor Sixth	1	1,5	0	-2	-2	0	0
Major Sixth	0	-2	-2	1	2	-2	0
Minor Seventh	0	1	-2	0	0	0	0
Major Seventh	1	-2	-2	2	-2	-2	0

The values shown in Table 4 were defined based on the frequency of their appearance and a post calibration done with the help of last semester music students.

4.1.5 Score of using a note value in the bassline

The frequency in which a note value in the bass overlaps with a note value in the melody is used to define a score for each pair of note values that overlap as shown in Table 5.

Table 5. Score for pairs of note values (melody in rows and bass in columns)

	Half	Quarter	Eight	Sixteenth	Thirty-Second
Half	-1	1	1	-1	-1
Quarter	-1	1	1	0	-1
Eight	-1	2,6	5	1	-1
Sixteenth	-1	2,7	5	1	-1
ThirtySecond	-1	-1	1	-1	-1

As with the previous metric, the values shown in Table 5 were defined based on the frequency of their appearance and a post calibration done with the help of music students.

4.1.6 Time signature of the genre

By far the most common time signature in the genre is 4/4. For this reason all generated basslines use this time signature as default.

5 MUSICAL NOTATION/ SOLUTION CODIFICATION

To represent melody-like and drum music a two-dimensional array is used. The array for melody and bass includes the succession of notes played along with the note value of each note, as Fig 3 shows.

Note 1	Note 2	…	Last Note
NV (1)	NV (2)	…	NV (Last)

Figure 3. Melody and bassline representation

Notes are written using the CAGED[7] system and each note value is represented by an integer which indicates how many notes with a particular note value would fit into one bar (Table 6).

Table 6. Note Value Equivalencies

Whole	1
Half	2
Quarter	4
Eight	8
Sixteenth	16
Thirty-Second	32

[7] In the CAGED system the notes Do, Re, Mi, Fa, Sol, La and Si are represented by the letters C, D, E, F, G, A and B respectively.

As an example, the bassline of the verse of the song "Snow Hey Oh" by the Red Hot Chili Peppers is presented in the required format (Fig 4).

G#	C#	D#	E	D#	C#	B	D#	E	F#	B	A#
4	8	8	4	8	8	4	8	8	4	8	8

Figure 4. Bassline of the song "Snow Hey Oh"

This bassline is composed of four quarter-notes and eight eight-notes, which implies this bassline lasts for two bars.

This format for music representation was inspired by traditional music notation and aims to capture the essence of the pentagram but in a way that can be easily handled for computer programming. Pentagrams include the sequence of notes played as well as the duration of each note. However in the pentagram this representation is of visual character as each note is defined by its placing within the pentagram, not by a letter and the note value of each note is represented by a symbol rather than an integer. In this sense both notations are quite similar and translation from one to another can be done easily. Nonetheless pentagram presents information regarding the octave of the notes played, while the notation defined for this work does not.

The format used for the drumbeat is similar to the precedent. Since the drumbeat can be decomposed into its three main elements: the kick drum, the snare and the hi-hat, each percussion element is represented as a two-dimensional array, which includes if the percussion is ringing at a certain time or not and the duration of each sound or silence, as in Fig 5.

Percussion	Silence	Percussion	…	Silence
NV (Percussion)	NV (Silence)	NV (Percussion)	…	NV (Silence)

Figure 5. Drumbeat representation

Although it can be considered that no notes are played in the drumbeat the use of note value is still valid as it refers to the duration of the sound or silence that is playing. Same conventions as for melody and bass are used regarding the note value equivalencies.

As an example, the beat of the kick drum of the song "Girls and Boys" by The Subways is presented in Fig 6, in which "K" stands for kick drum and "S" for silence:

S	K	K	S	K	K	S	K
8	8	8	8	8	8	8	8

Figure 6. Kick drum beat of the song "Girls and Boys"

The beat is composed of eight eight-notes, making it a one-bar beat.

Drummers use a notation similar to the pentagram, in which each line represents a percussive instrument and every time one is playing a symbol is used to represent the duration of the sound. This is very similar to the notation previously stated and one can be easily translated into the other without loosing any information.

6 FEASIBILITY OF SOLUTIONS

The feasibility of a solution is associated to two main constraints: the drumbeat constraint and the melody constraint.

When analyzing the relationship between the bass and the drumbeat several conclusions were made. In first place, the hi-hat tends to serve as an indicator of the tempo of the song and thus it lacks a strong relationship with the bass. Secondly, the snare provides great movement to the drumbeat but it was not possible to determine common snare/bass patterns, as these tend to change drastically from song to song. However this is not the case for the kick drum, which is what was expected since both the bass and the kick drum are the low-end sections of any alternative rock song. In greater detail "the bass is a low frequency instrument, so when it hits a note simultaneously with the kick the syncopation is strong and provides a solid foundation to the rest of the musical structure" (Bassinplace, 2008).

Because of this, only the kick drum was taken into consideration when evaluating the relationship between the bassline and the drumbeat.

6.1 Kick Drum Constraint

It was previously stated that the kick drum/bass relationship is the most important as both are the low-end part of any song and thus its relationship has special properties.

By looking at Figs 1 and 2 it is possible to conclude that all kick notes are played when a bass note either starts or right before it finishes. Also all kick notes have a smaller or equal note value than the bass note they accompany. For a bassline to be feasible it must

only present patterns from the allowed pattern list shown in Fig 1.

6.2 Bass/Melody Intervals Constraint

In order to avoid the situation in which a bassline follows too closely the melody a constraint regarding the musical interval between each pair of notes[8] that overlap was defined. The interval is calculated using the melody note as the root, so if the melody plays a C note while the bass plays a D note, the interval is "major second" rather than "minor seventh".

For a bassline to be feasible the proportion of each musical interval must not exceed the defined values in Table 3.

7 SOLUTION EVALUATION

Recall from section 5 that a bassline is composed of a sequence of notes, each one of them with an associated note value. In order to evaluate the quality or "fitness" of a bassline one must evaluate both the notes and note values that made it up. In favor of doing so, two criteria were defined. In first place how good is the groove of the bass given the note values used in the melody and secondly, how good is the sequence of notes the bass plays given the chords fitted to the melody. Both metrics are defined in a way that better solutions will have greater scores, meaning this is a multi-objective maximization problem.

7.1 Quality of Notes in the Bassline

In order to quantify the quality of the notes used in the bassline a score was constructed based on the notes played and the diatonic function of the section they sound in. The bassline generation is guided by the chords fitted to the melody, because of this it is easy to determine the diatonic function of the chord that is ringing when a certain note is playing. For this, the musical interval between the root of the chord and the note that sound at the same time is considered.

Using the scores presented in Table 4 the final score can be computed, which is equal to the sum of all the scores of the notes that are playing depending on the moment they are played, divided by the number of notes in the bassline. This is done to correct the effect in the final score that basslines with many notes will have as a consequence of summing more terms.

This metric was inspired by the work of (Lozano, Medaglia, & Velasco, 2009), in which a score is given to a note depending on the musical interval between it and a note of the chord that sound at the same time, depending on the diatonic function of the chord and the key. By doing this we are guaranteeing that the notes in the bassline complement the melody nicely without exhaustively comparing each pair of notes but rather ensuring that the notes of the bassline will suit the chord progression appropriately.

7.2 Groove of the bassline

The note values used in the bassline dictate its groove. Even if the notes used complement the melody nicely, the bass and melody would not blend pleasantly without a groove that allows it. In order to measure the quality of the bass groove the note values of the bass and the melody are compared.

In a similar fashion to the evaluation of the quality of the notes, using the scores presented in Table 5, a final score is computed for each solution, which is equal to the sum of all the scores of the notes that overlap divided by the number of notes that overlap. This is also done to correct the effect in the final score that basslines with many notes will have as a consequence of summing more terms.

This metric presents a way of determining if the note values in the bassline present a good configuration when compared with the note values of the melody. After all, these note values are responsible for how good the bass and melody blend.

8 THE SOLUTION ALGORITHM

The algorithm was designed to work best on particular riffs rather than on whole songs. This, because a vital input for the algorithm is the key of the melody and within the alternative rock genre it is not uncommon that the key varies throughout the song;

[8] The pair refers to one note from the bassline and one note from the melody.

and in this way the algorithm is more flexible being able to consider pieces of any length. To start the solution several inputs are necessary: a melody and a drumbeat in the format previously stated and some parameters that will be explained next.

8.1 Input Parameters

As it has already been established, flexibility within the algorithm is desired; because of this some parameters that change the way the solutions are constructed were included. These parameters must be defined by the user and are the following:

- *Key of the song* (Chosen among a list of candidates). Only major keys are accepted as valid input although this can be modified with the parameter *MajorMode*.

- *Number of chords per bar:* it takes the values of 1, 2 and 4 since higher values are unlikely to be found in the genre

- *MajorMode:* binary parameter that could be true if the selected key is going to be used or false if it is preferred to use the relative minor of that key instead.

- *DegreeOfTension:* 0 if a low degree of tension in the chord progression is desired, 1 if it is a high degree.

8.2 Determining the key of the song

The key of a song dictates which notes can and cannot be played within it, thereby determining it plays a vital role in reducing the solution space considering only chords and notes within the key of a song. It is important to mention that skilled musicians break this constraint on several occasions; however respecting it is a far more common practice. Because of this, the constraint is applied during the rest of the solution algorithm.

The key of a song is common to all the instruments in it, because of this, finding the key in which the melody most likely is, is equivalent to finding the key of the song. A matching algorithm is used for finding the possible keys that the melody is in. The matching algorithm works as follows:

1. An array containing the notes within the key is created for the twelve existing major keys. For example, the key of C major includes the notes C, D, E, F, G, A and B (This array is represented on Fig 7).

C	D	E	F	G	A	B

Figure 7. Array of the C major key

2. All the key's arrays are checked and a score is assigned to each key, which is equal to the sum of notes that appear both in the melody and in the array for that key. The maximum score for any key is seven as it is the number of notes that compose each one.

3. The key with maximum score is selected because it is the one that the melody is most likely to be in. If there are two or more keys with the same score, the user can select the one to be employed.

It is important to notice that only major keys are considered. This is due to the fact that all major keys have a relative minor, which is composed of the exact same notes. However this does not present a limitation since the major or minor character of the chord progression can be manipulated with the parameter *MajorMode*.

8.3 Breaking the melody into fragments

Taking the note values of the melody, a partial sum is computed from note i to note j in the following manner: $\frac{1}{NV(j)} + \frac{1}{NV(j-1)} + \cdots + \frac{1}{NV(i)}$. Given the fact that a 4/4 time signature is supposed, when this partial sum is equal to one a chord must be selected to accompany this fragment of the melody.

It is important to mention that the parameter *Number of chords per bar* works by multiplying the quotient $\frac{1}{NV(k)}$ for each note value in the melody. In order to show why this works, let's do an example.

Suppose we have the melody represented with Fig 8, which is a one-bar melody.

A	B	A	C	D	B	A	A
8	8	8	8	8	8	8	8

Figure 8. Generic melody

If one chord per bar is desired, the whole melody would be used for the matching algorithm since it is exactly one-bar long.

If two chords per bar are desired, the partial sum will be equal to one after the notes A, B, A and C are played, which can be easily seen by computing the sum: $\left(\frac{1}{8}\right) * 2 + \left(\frac{1}{8}\right) * 2 + \left(\frac{1}{8}\right) * 2 + \left(\frac{1}{8}\right) * 2 = 1$. This fragment would then be used for the matching algorithm. The same is true for the second half of the melody.

If four chords per bar are desired, the partial sum will be equal to one after the notes A, and B are played, which can be seen by computing the sum: $\left(\frac{1}{8}\right) * 4 + \left(\frac{1}{8}\right) * 4 = 1$. The same is true for the remaining three fragments.

8.4 Finding candidate chords

During this step the aim is to find the chords that fit particular segments (defined by the number of chords per bar) of the melody, as previously stated.

A matching algorithm is used in the same way as when defining the key of the melody but this time only for each fragment and returning chords, instead of keys.

Thirty-six arrays are created for each one of the major, minor and diminished chords in existence. Each one works the same as the key arrays, but this time the chords with the $maximun$ and the $[maximun - 1]$ scores are returned. This is done to have a bigger list of candidates to use.

8.5 Constructing the graph of chord progressions

For each melody fragment a set of candidate chords is determined using the matching, as previously explained.

Using these chords a graph is generated. Node 0 and U are dummy nodes representing the beginning and the end of progression. The nodes in between are the candidate chords found, arranged in a way that each "column" of the graph has the selected chords for each fragment. An arc (i, j) represents the possibility of passing from the chord i to j with an associated cost c_{ij} defined as in Table 1.

The next example aims to make this clearer.

Suppose we are constructing the graph for the song Warpaint by the band Warpaint. Fig 9 shows the melody of its verse.

E	G	B	E	E	E	B	E	B	D	A	D	D	D	D	C	D	E
8	8	8	16	8	16	8	8	8	8	8	8	16	8	16	8	8	8

Figure 9. Melody of the verse of the song "Warpaint"

This melody is a two-bar melody and one chord is desired per bar. For the first bar the matching algorithm returns the chords Em and G, which will be placed in the first "column" of the graph. In the other hand, the chords Am and D are returned for the second bar, which are placed in the second "column". The resulting graph is shown in Fig 10.

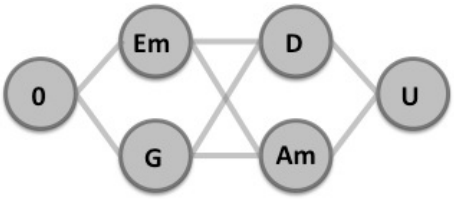

Figure 10. Progression graph of the song "Warpaint"

Finding the shortest path from node 0 to U allows us to find the optimal chord progression.

8.5 Finding the Shortest Path

In order to model this problem as a shortest path problem the costs of passing from any chord to another needed to be collected. As wide and broad as the alternative rock genre is, collecting this information for all the keys[9] presents a big roadblock since there are keys that are very popular and some that are very unpopular within the genre. For example, songs in the key of G major are really common

[9] The cost of passing from chord to another will be different for each key, since each chord plays a different role within each key.

while only a few songs are in the key of D# major. In order to solve this, the diatonic function of the chords was used instead.

The diatonic function is independent of the key in the sense that it functions equally for all the keys. Take the C major key as an example. Given this key it is possible to know the chords that are in it (C, d, e, F, G, a, bd) being **I** the diatonic function of the chord C, **ii** of d, **iii** of e and so on[10]. As an example, the chords in the key and their diatonic function are shown in Fig 11 for the keys of C and E major.

Key: C maj	C	d	e	F	G	a	bd
	I	ii	iii	IV	V	vi	vii

Key: E maj	E	f#	g#	A#	C	d	dd
	I	ii	iii	IV	V	vi	vii

Figure 11. Chords in the keys of C and E major and their diatonic function

Given the key and the progression graph it is possible to translate each chord into its diatonic function and then use the cost matrix presented in Table 1 to find the shortest path within the graph by using the Dijkstra algorithm.

In our example the translation will result in the graph shown in Figure 12, given that the song is in the key of G major.

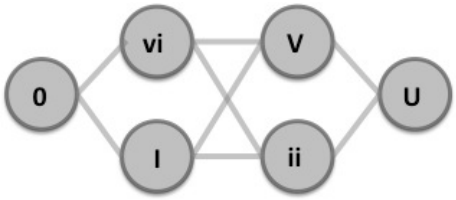

Figure 12. Progression graph of the song "Warpaint" using diatonic function

In this case the shortest path is $0 - vi - V - U$, which if translated back to chords means that during the first bar an E minor chord must be played and a D major chord during the second bar.

The graph represented in Fig 13 is an example of how the graph will look for the verse of the song Snow (Hey Oh) by the Red Hot Chili Peppers:

[10] In this context the notation used is lower case for minor chords and capital letters for major ones.

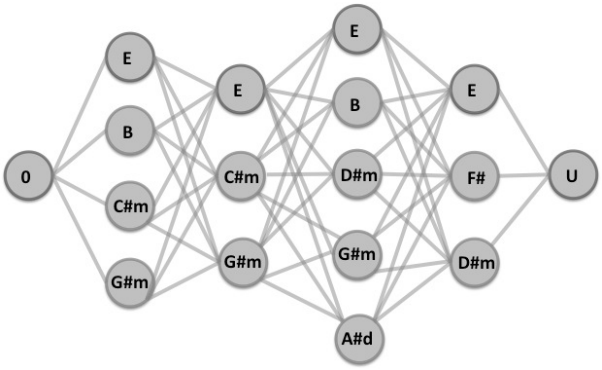

Figure 13. Progression graph of the song "Snow Hey Oh"

Given that the key of the song is B major and the appropriate number of chords per bar is two, when translating the chords to its diatonic function the shortest path is $0 - vi - IV - I - V - U$. This means that a G# minor chord must be played during the first half of the first bar, an E major chord during the second half of the first bar, a B major chord during the first half of the second bar and a F# minor chord during the second half of the second bar. It is important to notice that this chord progression shares the same root notes as the original bassline of the song (Figs 14 and 15).

G#m	E	B	F#m
2	2	2	2

Figure 14. Chords with duration for "Snow Hey Oh"

G#	C#	D#	E	D#	C#	B	D#	E	F#	B	A#
4	8	8	4	8	8	4	8	8	4	8	8

Figure 15. Bassline of the song "Snow Hey Oh"

8.6 Stochastic Diffusion Search for Bassline improvisation

Now that the key of the song has been defined and a chord progression has been fitted to the melody, it is time to compose the bassline.

The stochastic diffusion search algorithm works by selecting a number of "miners" who are responsible for exploring the solution space in search of good solutions. During the first iteration of the algorithm each miner randomly selects a solution. Solutions are then evaluated and the miner with the current best solution is marked as pleased. Unsatisfied miners will either search for a new random solution or copy the current best one given a probability of keeping exploring. The algorithm ends when all miners converge to a solution meaning they are all pleased.

This solution algorithm was chosen among many others because it combines exploration and intensification techniques in a way that few algorithms do. Also, it was used in (Majid Al-Rifaie & Majid Al-Rifaie, 2015) for generating music from plain text, which is a similar approach to the one in this work, and satisfactory solutions were generated.

The problem at hand tries to maximize two criteria, because of this, the traditional SDS algorithm had to be modified. Rather than being pleased when their solution is the current best one, miners will be pleased if their solution is included in the Pareto front.

The Pareto front is composed of all the solutions that are not dominated by any other solution[11]. The algorithm ends when all miners are pleased.

8.7 Generating Random Solutions

It was previously stated that in SDS miners search the solution space randomly. Because of this, an algorithm that generates random and feasible solutions needed to be developed.

Having the chord progression as well as the number of chords per bar an empty bass is created which contains zero notes and zero note values. Random note values are generated and added to the bassline until the whole, half or quarter (defined by the number of chords per bar) bar is completed. The values in Table 2 were used to define probabilities of generating a certain note value, which are equal to the observed frequencies presented in Table 2.

Then, a note is assigned to each note value by randomly selecting one of the three notes that made up the chord that is assigned during that time. This is done for the same number of bars of the melody.

Each randomly generated bassline must satisfy the constraint associated with the drumbeat. If a bassline does not satisfy this constraint it is eliminated and the process is repeated until one that does is found.

However these solutions rarely satisfy the constraint imposed by the musical intervals of the notes that overlap in the bass and melody. The solution is reconstructed until it satisfies both constrains.

For this, a position in the array of the bassline is randomly selected and a new note value is generated using the same probabilities as the initial solution.

If the new note value is equal to the old one, nothing happens.

If it is bigger, the new note value is used in the bassline and new note values are generated and added to the bassline until the bar is complete. Then for each new note value a new note is randomly selected from the list of notes that belong to the key of the song[12].

If it is smaller, the new note value is used in the bassline and notes of the same bar are randomly eliminated until the bar is complete.

It is important to notice that if a change in the solution results in it not respecting the drum constraint, the change is not made.

By doing this, in a few iterations the initial solution is reconstructed into a solution that satisfies both constrains and thus is feasible.

SDS runs until all miners are pleased which means they converge to a Pareto front.

9 RESULTS

The algorithm was tested on twenty of the songs in the database. In order to show the results obtained a particular and exhaustive example will be presented, followed by the comparison of different solutions with the original bassline for five songs.

The algorithm was applied to the song “Californication” by the Red Hot Chili Peppers, using the vocal line of the first verse of the song as melody. The song is in the key of C major and the chord progression generated (with one chord per bar) consists of Am and F in loop during the six bars of the fragment. Figs 16 to 20 show the Pareto front found for different number of miners and probabilities of keeping exploring:

[11] This means that there is not any bassline that has a greater value in both criteria.

[12] In the same manner that chords belong to keys, notes do so. Actually, the chords in a key exist within it as a result of the notes that made them up.

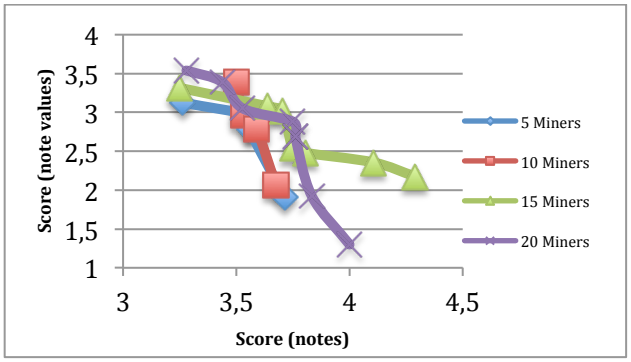

Figure 16. Pareto front for probability = 0,1

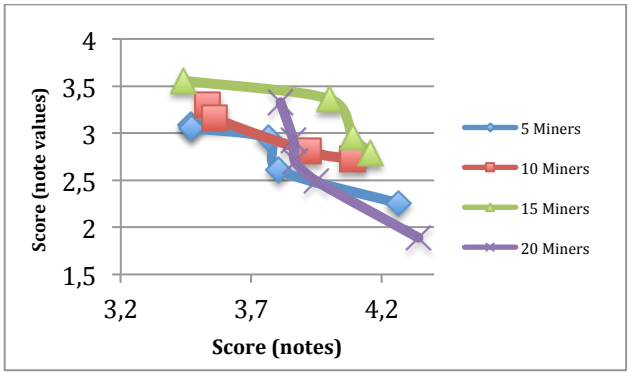

Figure 17. Pareto front for probability = 0,3

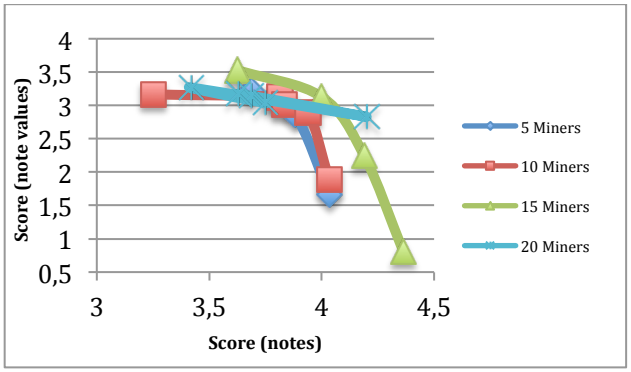

Figure 18. Pareto front for probability = 0,5

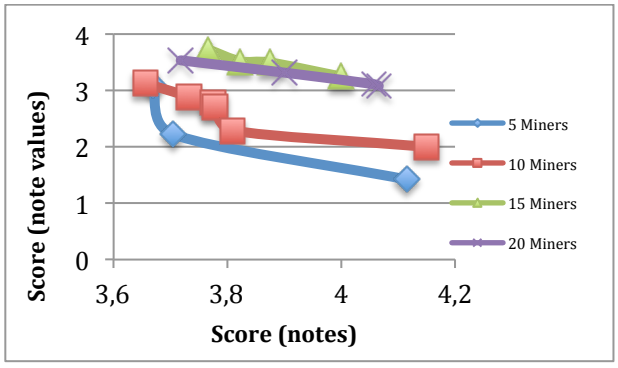

Figure 19. Pareto front for probability = 0,7

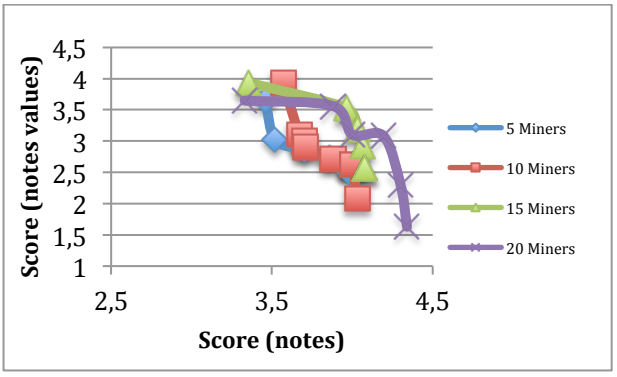

Figure 20. Pareto front for probability = 0,9

This exercise is done in pro of calibrating the algorithm by defining an appropriate number of miners and an appropriate probability, which will result in better solutions.

It is important to consider that SDS is a non-deterministic algorithm and thus it generates different solutions and different Pareto fronts every time it runs. Even though it can vary, the behavior of the fronts shown in this example is common to almost all scenarios for this song.

Analyzing the Pareto fronts generated it was found that in general, dominant fronts produce better solutions over dominated fronts. The front generated by using twenty miners is similar or worst than the one using fifteen, because of this fifteen is a good number of miners for this particular song. Fig 21 shows the fronts using fifteen miners and different probabilities.

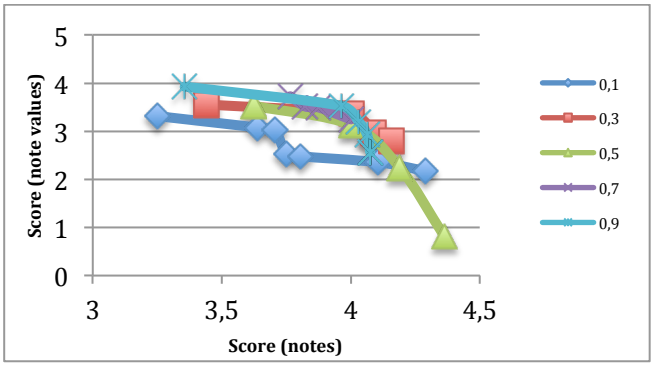

Figure 21. Pareto front for number of miners = 15

Comparing the fronts in Fig 21 it can be seen that similar fronts are generated for probabilities greater than 0,3. It is concluded that a good probability of keeping exploring for this particular song is 0,5 since it generates good solutions in a fraction of the time than with greater probabilities.

Using the Pareto front of fifteen miners and probability equal to 0,5 the solutions are saved. Out of four basslines considered, two blend nicely with the melody and the drumbeat according to the subjective opinion of the authors and consultants who are currently studying music.

Figures 22 to 25 present the Pareto front[13] vs the objective values of the original bassline of the song for five different songs, by Arctic Monkeys, Warpaint, The Subways and Red Hot Chili Peppers (last two), respectively.

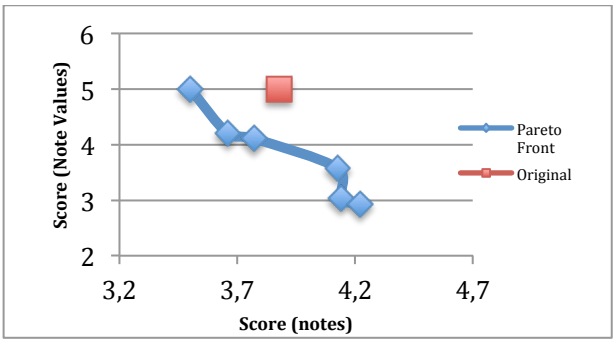

Figure 22. PF[14] vs OB[15] for the song "Black Treacle"

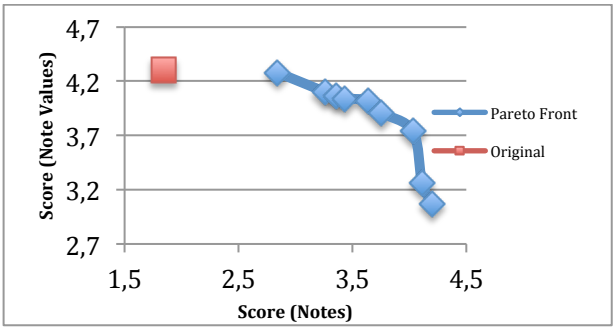

Figure 23. PF vs OB for the song "Elephants"

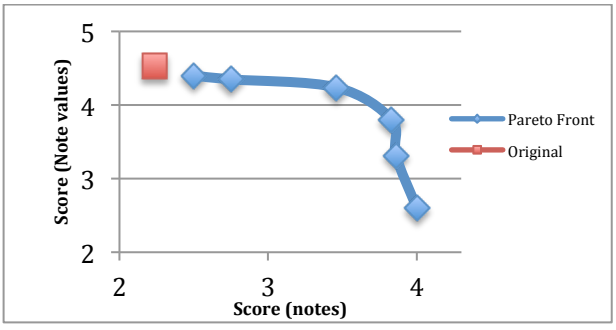

Figure 24. PF vs OB for the song "Girls and Boys"

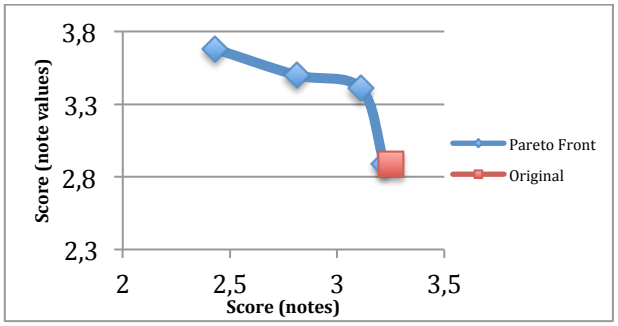

Figure 22. PF vs OB for the song "The Zephyr Song"

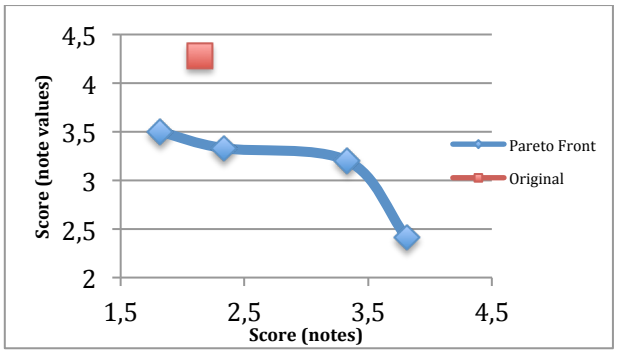

Figure 25. PF vs OB for the song "This Velvet Glove"

In general terms, the original bassline of the songs either will be part of the Pareto front or dominate it. This was expected since the aim of this work is not to generate similar or better solutions than the original bassline but rather "good" basslines.

One could think that the solutions of interest are located in the middle of the front, as extreme solutions have a really good value for one criterion but perform poorly in the other. However, it was determined that this is not the case, as in many cases the original solution is an extreme solution of the Pareto front. This is evidence of the fact that at least one of the solutions in Pareto front, regardless of their position within it, blend nicely with the melody and the drumbeat.

The algorithm was tested for several other songs and it was determined that if a correct number of miners and a correct probability are selected at least one of the solutions in the Pareto front will complement the melody and the drumbeat pleasingly according to the subjective opinion of the authors and last semester music students.

Figures 26 to 28 present the average number of objective function evaluations, average number of iterations and average time elapsed for thirty runs of the algorithm using different numbers of miners and probabilities for the song "Californication".

[13] After proper calibration
[14] Pareto Front
[15] Original Bassline

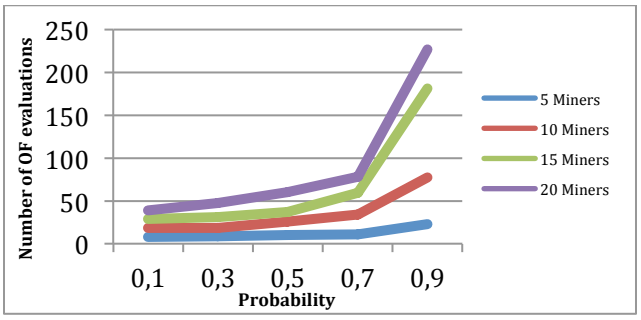

Figure 26. Number of objective function evaluations

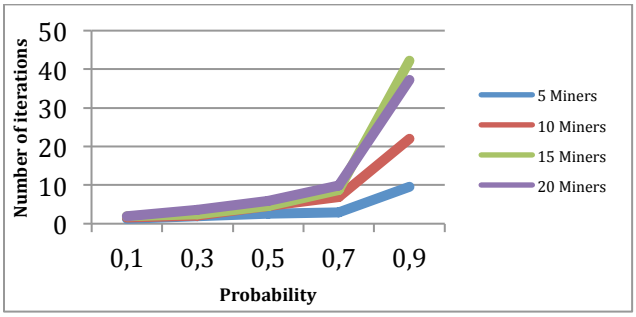

Figure 27. Number of iterations

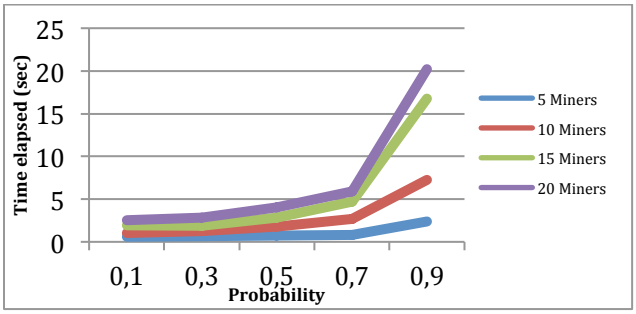

Figure 28. Time elapsed in seconds

As it can be seen in Figs 26 to 28, in general, the average number of objective function evaluations, the average number of iterations and the average time elapsed increase when there is an increment in the number of miners and probability. This was expected because a greater probability forces the miners to keep exploring instead of copying a solution that belongs to the Pareto front and more miners allow greater exploration but slower convergence.

10 CONCLUSIONS

The area of music composition through optimization techniques is an area that has developed greatly during new millennium. Researches tend to focus on chord fitting for melodies or generation of melodies and rhythms. However, bassline composition is still an underdeveloped topic in this area, which has been studied in very few investigations. This work pretends to expand the state of art of this topic.

The aim of this research was to solve the problem of finding a good bassline for a given melody and a given drumbeat, which was satisfactorily achieved through the use of a flexible algorithm that combines different metaheuristics to solve a multi-objective optimization problem, which guaranties that at least one good bassline will be found if calibrated appropriately. Nonetheless, there were cases when at least one good solution was found but the other solutions did not blend well, especially with the melody. This indicates that the algorithm still can be improved by refining the solution method and the metrics used for quantifying the quality of the basslines, which can be tackled in a future research.

It is important to mention that the quality of the solutions was determined by the opinion of the authors and the opinion of last-semester music students who served as consultants.

One limitation of the algorithm is that it does not take into consideration the octave of the notes that are being played either in the melody or the bassline. When transcribing a solution to musical notation it was up to the authors to decide the octaves of the notes in the bassline. Deciding the octave of the notes given the bassline is an interesting problem that can be studied in future investigations.

11 BIBLIOGRAFÍA

Alfonseca, M., Cebrián, M., & Ortega, A. (2007). *A Simple Genetic Algorithm for Music Generation by means of Algorithmic Information Theory.* IEEE Congress on Evolutionary Computation .

Bassinplace. (1 de December de 2008). The kick drums relation to the bass? TalkBass Forums.

Dostál, M. (2005). *GENETIC ALGORITHMS AS A MODEL OF MUSICAL CREATIVITY – ON GENERATING OF A HUMAN-LIKE RHYTHMIC ACCOMPANIMENT.* olomouc: Palacký University .

Dubnov, S., Assayag, G., Lartillot, O., & Bejerano, G. (2003). *Using Machine-Learning Methods for Musical Style Modeling.* IEEE Computer Society.

Geem, Z., & Choi, J.-Y. (2007). *Music Composition Using Harmony Search Algorithm.* Springer.

Herremans , D., Weisser, Sorensen, K., & Conklin. (2015). *Generating structured music for bagana using quality metrics based on Markov models.* ELSEVIER.

Herremans, D., & Sorensen, K. (2013). *Composing fifth species counterpoint music with a variable neighborhood search algorithm.* Antwerp: ELSEVIER.

Herremans, D., Martens, D., & Sorensen, K. (2014). *Dance Hit Song Prediction.* Antwerp: ANT/OR.

Kunimatsu, K., Ishikawa, Y., Takata, M., & Joe, K. (2015). *A Music Composition Model with Genetic Programming –A Case Study of Chord Progression and Bassline-.* Nara.

Laine, P., & Kuuskankare, M. (1994). *Genetic Algorithms in Musical Style oriented Generation.* Helsinki: Sibelius Acadmy Computer Music Studio.

Lichtenwalter, R., Zorina, K., & Chawla, N. (2008). *Applying Learning Algorithms to Music Generation.*

López-Ortega, O., & López-Popa, S. I. (2012). *Fractals, fuzzy logic and expert systems to assist in the construction of musical pieces* . Hidalgo: ELSEVIER.

Lozano, L., Medaglia, A., & Velasco, N. (2009). *Generation of Pop-Rock Chord Sequences Using Genetic Algorithms and Variable Neighborhood Search.* Bogotá: Springer.

Majid Al-Rifaie, A., & Majid Al-Rifaie, M. (2015). *Generative Music with Stochastic Diffusion Search.* London: SpringerLink.

Ren, T., Wang, Y.-f., Du, D., Liu, M.-m., & Siddiqi, A. (2015). *The guitar chord-generating algorithm based on complex network.* Shenyang: ELSEVIER.

Tojui, N., & Iba, H. (2000). *Music Composition with Interactive Evolutionary Computation.* Tokyo.