MELODY HARMONIZATION IN EVOLUTIONARY MUSIC USING MULTIOBJECTIVE GENETIC ALGORITHMS

Freitas, A. R. R.
Universidade Federal de Ouro Preto (UFOP)
Brazil
alandefreitas@gmail.com

Guimarães, F. G.
Universidade Federal de Minas Gerais (UFMG)
Brazil
fredericoguimaraes@ufmg.br

ABSTRACT

This paper describes a multiobjective approach for melody harmonization in evolutionary music. There are numerous methods and a myriad of results to a process of harmonization of a given melody. Some implicit rules can be extracted from musical theory, but some harmonic aspects can only be defined by preferences of a composer. Thus, a multiobjective approach may be useful to allow an evolutionary process to find a set of solutions that represent a trade-off between the rules in different objective functions. In this paper, a multiobjective evolutionary algorithm defines chord changes with differing degrees of simplicity and dissonance. While presenting such an algorithm, we discuss how to embed musical cognizance in Genetic Algorithms in a meta-level. Experiments were held and compared to human judgment of the results. The findings suggest that it is possible to devise a fitness function which reflects human intentions for harmonies.

1. INTRODUCTION

Genetic Algorithms (GAs) have a wide range of applications in Science and Engineering, in complex problems for which a specific solution is difficult to find. The search capabilities of GAs have drawn the interest from many scientific communities, including even applications in art and music [1, 2, 3, 4]. There have been many studies involving evolutionary computation and art trying to understand the possible influence of bioinspired systems on art, technology and aesthetic appreciation [3]. The use of GAs for evolving and creating art and music is a field known as evolutionary art and music [2].

Many successful applications of GAs for music analysis and synthesis can be found in the literature [5]. One of the main uses of GAs in this context is the formation of melodic lines. An algorithm for the creation of jazz solos in real-time has been proposed [6, 7], showing how the development of automated composition for a particular genre of music can be approached with GAs that consider many features of musical tasks, such as audition and improvisation. Although there have been many interesting results, evolutionary music still faces many challenges [8] such as the evaluation of solutions.

The search for harmonies is another field of study altogether, which usually has the advantage of having fitness functions which are easier to specify. It has been shown that algorithms can create harmonies successfully [9]. Many of those works describe systems to generate four-part harmonies for a given melody (SATB harmonization) [10]. In these algorithms, a group of notes is usually generated for each original note in the melody and that is an easy problem when the chord changes are also given with the melody [11]. Analogously to methods for the generation of melodies, the development of harmonization methods focused on a particular style may be interesting [10].

When creating chord changes for pre-existing or earlier generated melodies, it is needed to define some factors that should reward or penalize an individual in its fitness computation. However, these factors are usually subjective and strongly rely on user preferences. The main idea in this paper is to present a multiobjective optimization approach for harmonization that is able to evolve harmonies while dealing with the trade-off between harmonies with or without tension [1] . It is important to base composition methods on the formal aspects developed for tonal music. However, respecting some set of basic rules does not guarantee that the result will be musically meaningful or interesting. Creativity and new ideas may even follow from the violation of some rules, but the problem is knowing which set of rules to violate and when.

The results show that the multiobjective algorithm is able to find a set of solutions that represent a compromise between rules in different objective functions, leading to at least a subset of solutions that people would consider musically interesting. We also present a novel harmony representation scheme which defines a chord for each measure, or a fraction of a measure, which may generate results that differ from those of the traditional approach for SATB harmonization, in which a group of notes is defined for each note of the melody. Besides this, five notes are allowed to exist in a chord and some notes can be left unused.

In this paper, we initially present a brief introduction to GAs and give a few examples of how some of its aspects can be decided in the musical context. We then give more specific definitions of musical terms relative to harmonies

[1] Tension is created in a chord by including extra dissonant notes which create the need for relaxation or release to the listener

Copyright: ©2011 Freitas, A. R. R. et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

and how our experiments are outlined. We discuss how a multiobjective fitness function can help users with different or unknown preferences. Finally, we analyze the results of the experiments and present our conclusions.

2. GENETIC ALGORITHMS

Genetic Algorithms (GAs) are an evolutionary inspired [12] method initially proposed to solve optimization problems [13]. It is a very useful technique for hard search problems in which there is a manner to evaluate solutions but there is no known algorithm that is able to determine the optimal solution in polynomial time.

Those systems can achieve successful results for the task of generating harmonies [14] but some difficulties arise. The first of them is that there is no guarantee that the optimal solution will be found, which is understandable when a program is expected to solve such problems. The second complication is that GAs may become very computationally expensive systems when much knowledge is added to the system. This sort of knowledge is indispensable for any system expected to produce good results.

The main components of a GA are:

- A population of individuals (also chromosomes or genotypes) which represent possible solutions (phenotypes) for a problem. Those chromosomes must have some knowledge about the solution structure as the information in the chromosome maps to a phenotype which will undergo a process of natural selection. The chromosomes may represent musical information in the form of notes, frequencies or events.

- A selection process that judges the solutions according to an evaluation function (or fitness function). This operation selects which chromosomes shall have opportunity to reproduce and pass on part of its values (or genes) to the following generations. The computation of fitness for musical tasks are usually heuristic, interactive or based on rules [2]. Each of them have some assets and drawbacks.

- Genetic Operators to generate new potential solutions which will be latter tested. Those operators are structured considering information and premises about how the search for new solutions may take place. At this level, those operators must be able to combine previously selected information contained in current solutions in an effective way while having the potential to explore all possible solutions. Also, in musical systems, those operators can be guided for not generating solutions considered out of context.

Fundamental knowledge of the field must be given to the GA in order to distinguish between good and bad solutions, at least. Regarding the creation of harmonies, there must be enough knowledge embedded in the whole system to generate acceptable solutions in terms of pitches, durations and harmony itself. It is important to envisage how to embed this sort of information not only in the fitness function but also in other features such as the representation of solutions, since the algorithm can not generate any music solution not foreseen by its representation format.

Difficulties relative to the generation of harmonies using GAs and its subsequent implications are analyzed in the following sections.

3. EVOLVING HARMONIES

3.1 Representing a solution

It is important to have solution (or harmony) representations which are convenient to map fundamental aspects of Western music, namely pitch and chords. Still considering implicit rules of Western composition, the group of notes forming a triad of a specific chord has been the basis of the process of harmonization over the last centuries. The other characteristics of chords are usually described by the degree of the scale a given note is at.

Since tuning systems of equal temperament are the mainstream in Western music, in most cases, it is handy to use the relative shift between notes for the representation of music rather than the absolute values of pitches. Thus, the final result can be normalized to represent that solution in any specific key convenient to the user. Also in the final result, the octave in which each note will be performed may be conveniently decided by the user.

There are twelve notes (C, C#, D, D#, E, F, F#, G, G#, A, A#, B) and 7 degrees in a diatonic scale (Using C major as reference: C (1), D (2), E (3), F (4), G (5), A (6), B (7)). The degree of the remaining notes is defined in relation to the degree of its closest note (for instance, F# could be 5-, having a C major scale as reference). The interval between two consecutive notes is a semitone interval (for example, C and C#, or E and F). An interval of 2 semitones is a whole tone interval (for example, C and D, or E and F#). More information on intervals can be found in [15].

The representation must be meaningful to the specific domain, which is only the representation of harmonies in this paper. Thus, a given harmony is defined by a group of arrays, each of them defining a set of notes to be performed during that measure. This representation scheme considers a degree of granularity m of the chords in which a degree $m = 1$ means that one chord will be searched for each measure.

It is common to define a granularity for representation schemes in computer music [16] since the search space could be infinite otherwise. If the user wants more chords per measure, it is only necessary to increase the value of m. Therefore, m becomes a parameter of the algorithm or an aspect of the problem. Another solution possible for a variation in the granularity of the final result is to have a database with some chord progressions of different granularities that could be used among measures of the final result.

In this paper we propose a representation scheme for harmonies as the one in Table 1, where it is possible to perceive that the absolute frequencies of the notes are not represented. Those frequencies are defined in relation to the

Measure	1	2	...	n
5th Note	E	_	...	D
4th Note	C	D	...	F
3rd Note	G	A	...	D
2nd Note	E	F	...	B
1st Note	C	D	...	G

Table 1. Representation of a harmony

1st note, which is also the lowest pitched note. The octave of that note can be previously defined by the user in a convenient way for the composition. The use of 5 notes makes possible the use of a triad and two extra notes in the same chord. The 5th note is optional, as in Measure 2 of Table 1, because it could lead to many inevitable penalties during the evaluation of solutions in some specific contexts. Thus, a chord may have 4 or 5 notes.

This design avoids the generation of solutions with large vertical intervals because the 1st note is used as an anchor to decide the pitch of other notes. This codification makes the system more flexible than systems with only 4 notes (which would not be able to represent some dissonant chords) and avoids too large vertical intervals (because of the definition of the pitches in relation to each other), which would probably be treated by penalties in the fitness function if other representation approaches were used.

At the computer level, the notes can be represented by integer numbers from 1 to 12 and the codification does not have to define the classes of pitch of the notes as it does in Table 1, because a tuning system of equal temperament is used in the work. Therefore, the relation of intervals between the notes becomes more important than the absolute value of each note. The set of notes in each chord is represented by:

$$h_j = \langle \eta_{1j}, \eta_{2j}, \eta_{3j}, \eta_{4i}, \eta_{5j} \rangle,\ j = 1, \ldots, n \qquad (1)$$

with $\eta_{ij} \in \{0, \ldots, 12\}$ and 0 meaning no note.

3.2 Genetic Operators

The creation of new chromosomes throughout the generations is made by carefully designed genetic operators, which actually allow an evolutionary process to occur. The most common operators are mutation and crossover, though more complex operators can be designed for a specific field of problems. These operators may be blind or guided, which means they do not always occur in a completely random manner. Guided operators can help musical applications since completely random changes in the chromosomes are not convenient in some situations. For instance, if a group of genes represents a note, the mutation can be guided to respect some harmonic or melodic rules.

Some common musical operators are used in our implementation to make the evolutionary process more efficient. Apart from the crossover, all other operators occur in the mutation phase. The probability to get into the mutation phase is 20% and each mutation operator has another probability to happen. This value is defined relatively high because some of the possible mutations that do not affect the solutions very much have greater chance of being chosen. The probability of getting into the crossover phase is 90% and only a musical crossover can happen. All probabilities of applying those operators were arbitrarily chosen by the authors based on the experience acquired after many executions of the algorithm. The musical operators used for evolving harmonies are described below.

Musical Crossover The crossover implemented here does not work at a bit level, since it could generate many random solutions. Similarly to the usual crossover, a point is chosen between the parents and they share information considering the information to the left or to the right of this point. Nevertheless, the information of a measure is inseparable and the crossover point must be between 2 measures. The selection of only 1 crossover point is convenient since it does not break many relations between measures. Using more crossover points would be very disruptive. The relation between measures is important because many fitness restrictions depend on the association between measures. The probability of a crossover is 90%.

Pitch Mutation Changes the pitch of one of the notes in a measure. The maximum change in pitch possible is 1 tone because even though some large vertical intervals may be musically acceptable, many of them lead to no musical result. A larger interval can be reached by the effect of more than one operator or even many pitch mutations but not with a single pitch mutation. Intervals of at most one tone also force the algorithm to explore more dissonant chords since a note from the triad will never be mutated to another note of the triad. Note that very large leaps are not even foreseen by the representation scheme. The probability of using this operator is 30%. If there is no note in the randomly selected position (only possible in the 5th note of a measure), the operator is not applied.

Swap between the same measure To exchange the position of notes in the same measure. That operator creates inversions in the given chord. The probability of using this operator is 50%. If a position with no note is selected, a new position is selected.

Reinitialize the chord It reinitializes all notes of a measure using a triad that includes a melody note from the corresponding measure. That makes new chords with a high possibility of being acceptable. All triads have the same probability of being chosen and the probability of using this operator is 15%.

Copy Copy the information from a measure to another measure, creating the repetition of some chords in the harmony. The probability of using this operator is 5%.

Genetic Operators	Probability
Crossover phase	90%
1: Musical Crossover	100%
Mutation phase	20%
1: Pitch mutation	30%
2: Swap notes	50%
3: Reinitialize	15%
4: Measure Copy	5%

Table 2. Genetic Operators

Measure	1	2	3	4	5	6	7	8	
Notes									**Total**
C	**1**	0	0	**2**	**1**	0	**1**	**2**	**7**
C#	0	0	0	0	0	0	0	0	0
D	0	0	**1**	0	0	0	**1**	0	**2**
D#	0	0	0	0	0	0	0	0	0
E	0	0	0	**2**	**1**	0	**1**	0	**4**
F	0	0	0	0	0	**2**	0	0	**2**
F#	0	0	0	0	0	0	0	0	0
G	**3**	**2**	**1**	0	**1**	0	0	0	**7**
G#	0	0	0	0	0	0	0	0	0
A	**1**	0	**1**	0	0	**1**	0	0	**3**
A#	0	0	0	0	0	0	0	0	0
B	0	**1**	0	0	0	**1**	0	0	**2**

Table 3. Representing the problem

The probability of getting into the mutation phase was defined to 20% due to some mutations that do not alter expressively the fitness of a solution, such as the swapping of notes, which can only interfere in the condition of the position of the root note.

Table 2 summarizes the genetic operators used in this work with their respective probabilities of being applied.

3.3 Representing the problem

The problem (or the melody to be harmonized) is represented by a matrix in which a column represents a measure and twelve rows represent the possible notes in that given measure. Thus each element A_{ij} of the matrix represents the number of occurrences of the note i in the measure j. This representation scheme is exemplified in Table 3.

Summing up the values of all columns, as it is done in the column *total*, it is possible to have an idea of which scale is implicitly used in the melody. Otherwise, it is necessary to find in which measures the exceptions of a scale happen and which scales could be considered feasible to the problem.

In the example of Table 3, only notes of the C major scale are represented in column *total*, which means the implicit scale given by the melody is C major - or any relative scale, such as A minor. The scale used in a song as well as the present quantity of each note can be used later in the fitness function.

If the melody does not have enough notes to define a complete any of the predefined scales in column *total*, a scale that contains all notes from column *total* will be chosen. On the other hand, if the melody has so many notes that every scale lacks some note to match column *total*, some measures of the melody must be considered exceptions until an exception pattern of a scale for each chord matches column total. In those cases, the user can also choose the scales used by the algorithm to harmonize the melody.

Only diatonic scales were considered in this experiment. In spite of that, it is easy to notice that any scale could be considered by the method.

3.4 Multiobjective harmonization

Two evaluation functions are defined in order to deal with the trade-off between dissonance and simplicity of the harmonies. That is done by considering that harmonies containing chords with only a simple triad (1st, 3rd and 5th) or a simple triad plus a 7th in some cases are good harmonies regarding the simplicity criterion. In those harmonies, a 7th is not considered a desirable note unless it is in a context in which it results in an appropriate note on the following chord, later on. The maximization of a fitness function defined as simplicity function f_1 will represent the evaluation of this sort of harmonies.

A second evaluation function to be maximized, named dissonance function f_2, gives better fitness values to harmonies with dissonance on their chords, and measures with more notes than the triad are usually rewarded. Despite this consideration, notes besides the triad which do not fit the melody are penalized the same manner they would be in the first evaluation function.

With those functions, we define the harmonization stylistic contexts to be conciliated by the algorithm.

Perhaps, it would be possible to conciliate both objectives if we already knew the preferences of the user or if the designer becomes the user and use his own preferences to weight contradictory options. While some objectives are usually common to all harmonies, such as defining specific triads to guide the chord progression, some objectives are very contradictory according to the preferences of the user. For instance, the inclusion of dissonant notes will necessarily give different rewards and penalties to different functions.

With a multiobjective approach, the user can obtain a set of harmonies that fit that melody and represent a trade-off between the rules contained in the two objective functions. Then, the user can choose a harmony from the set of results and define the preferences while listening to the solutions.

As there are two fitness values for each solution now, we need to decide how to rank the solutions in relation to all functions. In order to do that, the solutions with the best values regarding both functions are grouped on a front which is numbered as front 1. The best solutions among the remaining ones are grouped on front 2 and so on. For solutions in the same front, the individuals are ranked accordingly to their groups and their distance from each other. This strategy is known as a Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [17] approach.

3.4.1 Evaluation of the solutions

In order to rate how good a solution is, it is necessary to define how knowledge about the subject will be built into the system. As it has been shown, much of this knowledge have already been implemented in the genetic operators but those are not able to distinguish which ones are good or bad solutions. The fitness of a solution is given depending on matters of note durations and vertical intervals in a harmony.

We need to determine some factors that may be desirable or penalized for solutions in different contexts. The relevant factors for evaluation of harmonies used in this work are described below.

- **Triads:** The triad has been the basic structure of a chord in Western Music over the last centuries. The absence of triads have a penalty of 40 in this work. If there is a 3rd but there is no 5th, the penalty reduces to 15 in the simplicity function f_1 and 5 in the dissonance function f_2. With a triad, there are two notes left in the representation scheme that can be used to create tension.

- **Dissonant Chords:** They may be desirable depending on the context. That is what mainly justify the use of 2 evaluation functions in this work. Notes out of the triad are considered dissonant and they are penalized in the first fitness function by 10 while rewarded by 10 in the dissonance function. A 7th is not penalized in the simplicity function if it leads to another note on the following compass with a semitone interval. Two dissonant notes also lead to a penalty of 20 in both functions if they are semitone distant.

- **Invalid Pitches and Chords:** Notes which do not belong to the scales implicitly defined by the melody are penalized by 30. A note is considered invalid in the simplicity function if (i) it does not belong to the main implicit scale of the song and (ii) it does not exist in the respective measure of the melody. A note is considered invalid in the dissonance function if (i) it does not belong to the main implicit scale of the song, (ii) it does not exist in the respective measure of the melody, (iii) it does not chromatically lead to another note in the following measure and (iv) there is a note semitone distant from it in the respective measure of the melody. The penalty for this restriction is 30 as it has to be greater than the reward for dissonant chords in the dissonance function.

- **Avoid unisons:** It is not desirable to have unisons spending notes that could be used to form new chords. Unisons of the root note may be normal while unisons of the 3rd are not usually accepted. Most unisons are penalized by 5, unisons of 3rds are penalized by 10 and unisons with the root note are not penalized.

- **Sevenths:** Sevenths that lead to another note in the following chord are rewarded by 10 in the dissonance function. They are rewarded by 10 in the simplicity function only if they lead to a 3rd in the following chord.

- **Tonic position:** Having some reward to chords with its root note in the 1st note, which is meant to be the lower note, may be interesting. Otherwise, there would be no difference of fitness between inverted chords and most chords would tend to be inverted. The value of the reward is 10 in the simplicity function and 3 in the dissonance function.

Condition	Simplicity Function	Dissonance Function
1: Absence of a triad	-40	-40
1.1: Absence of 5ths	-15	-5
2: Dissonant note	-10	+10
2.1: Meaningful 7th	0	+10
2.2: Semitone dissonance	-20	-20
3: Unisons	-5	-5
3.1: Unison of 3rd	-10	-10
3.2: Root note unison	0	0
4: Invalid note[a]	-30	-30
5: Meaningful sevenths[b]	+10	+10
6: Root note position	+10	+3

[a] The conditions for an invalid note are different in each function.

[b] The conditions for a meaningful 7th are different in each function.

Table 4. Parameters of the fitness evaluation

Other factors that could be used to influence the fitness of the harmonies in different contexts are (i) the distance of vertical intervals between the notes, (ii) the omission of specific notes, (iii) the range of the notes and (iv) stylistic predefined chord progressions. The latter would give implicit rewards to some genres of music as those sorts of rules cannot be considered in general even when considering only Western music.

Table 4 shows a brief description of the conditions of evaluation with their associated influence on the fitness function.

4. EXPERIMENTS AND RESULTS

Experiments with a population of 100 randomly initialized individuals were held with 100 executions of the algorithm. Random values between 1 and 12 are given to each note, apart from note 5 of each measure, which receives values from 0 to 12, where 0 means there is no note. The representation is made in the same way as in Table 1.

A known melody was used as the problem to give a better idea of the capabilities of the algorithm. This melody is represented in Figure 1.

To notice how the operators are influencing the execution of the GA, Figure 2 shows the influence of each condition of the fitness function in the whole population after 100 executions of the algorithm. Each group of bars shows the average influence of the restrictions for 20 generations.

The restrictions are numbered in the x axis in the same manner as they are numbered in Table 4. The values of influence consider the average influence in all individuals on

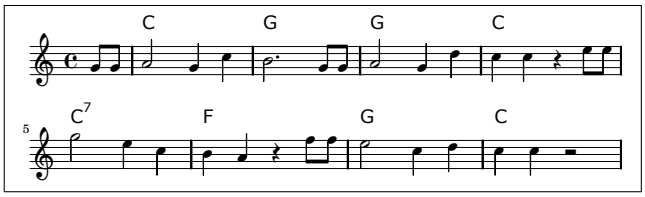

Figure 1. *Happy Birthday to You* and its harmony.

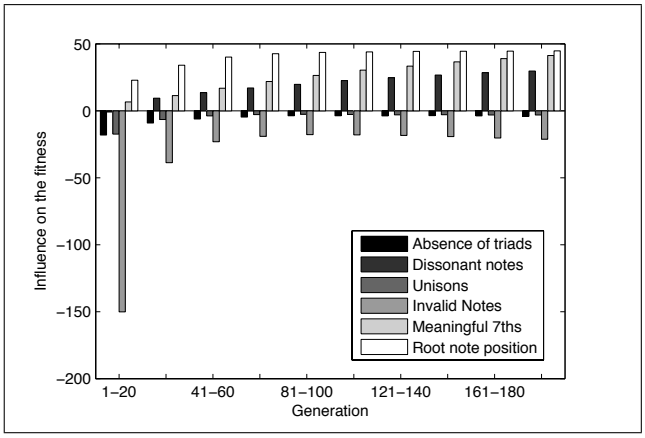

Figure 2. Fitness profile through the generations.

the respective generations, considering functions 1 and 2. It is clear that the presence of invalid notes has the greatest weight in the evaluation of the solutions. This happens mainly because of the nature of the problem and the penalties for invalid notes never get very low because two conflicting concepts of invalid notes are defined in the fitness functions. The graph was created considering 200 initial generations of this experiment.

Figure 3 shows one solution taken from the Pareto front with a good value in the simplicity function $(50, 61)$. The generated harmony is very simple and has much similarity with the original harmony of the song represented in Figure 1, which the algorithm has no explicit access to. The only differences would be the minor chords, which are inexistent in the original song (Dm is the relative minor chord for F).

The possibility of getting harmonies similar to the original one is an evidence that the algorithm is resulting in musical solutions. However, that does not mean that a solution which does not look like the original one is not musical. Such harmonies may be found and copying or finding the original harmony of a melody is not the goal of this work. That is why "ground-truth" tests, based on the original harmony, do not apply to this algorithm as our intention is to create feasible harmonies as diverse and creative as possible.

Figure 4 shows one solution taken from the Pareto front with a good value in the dissonance function $(-40, 101)$. It is possible to notice that the algorithm could manage to generate more complex harmonies even for a simple melody.

It is also interesting to note that this solution has a D, which does not belong to the implicit scale (Dm does). However, this D is in a measure in which there is no F,

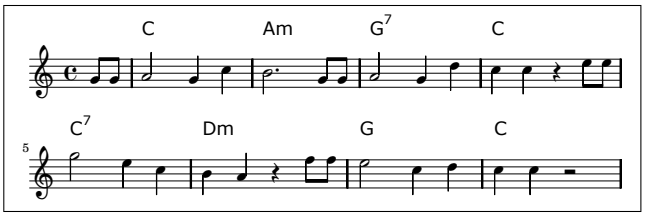

Figure 3. Solution 1.

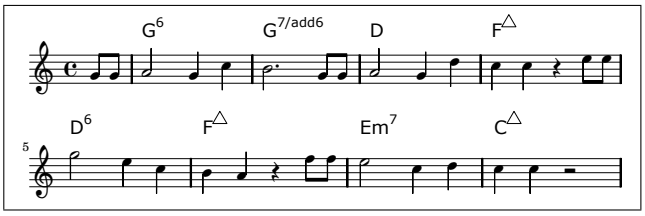

Figure 4. Solution 2.

which would be the 3rd of a Dm. Thus, the note F# in the D chord of the harmony is not penalized in the dissonance function because it chromatically leads to the following chord ($F^{\triangle}$).

Also according to the determined rules, the algorithm can be considered successful since all individuals in the Pareto front of the last generations examined broke very few minor restrictions in the simplicity function or had positive values of fitness for the dissonance function.

An example of the evolution of the Pareto fronts in a single execution of the algorithm is represented of Figure 6. The lines represent the points dominated by the best solutions in a given generation while the dots represent the fitness of all solutions evaluated by the algorithm. In all executions of the algorithm, the extreme fitness values found as an answer for the simplicity function were $(-310, 251)$ and $(120, 121)$ while the extreme values found in the dissonance function were $(70, 21)$ and $(-60, 251)$. The final Pareto fronts have an average of 21.08 individuals, being the average fitness value of these individuals $(4.36, 178.14)$.

The effectivity of the algorithm and its operators have to be measured accordingly to the quality of the set of best solutions. In order to do that, the hypervolume of the Pareto fronts can be used [18]. With this strategy, the area, volume or hypervolume dominated by the best set of solutions is used to measure the quality of the population in that respective generation. With the average hypervolume of all generations, it is possible to define the efficiency in the convergence of the algorithm.

The hypervolumes were measured with a reference point $(-1200, -1200)$ for the dominated area. Figure 5 represents data about the hypervolumes generated throughout 200 generations (considering 100 executions). The axis x is in logarithmic scale.

The algorithm evaluated 20,000 solutions in 200 generations and the average hypervolume of the final Pareto front is 1.8197×10^6 (standard deviation 3.1243×10^4) while the initial Pareto front formed from 100 random solutions has an average hypervolume of 9.3421×10^5. A Pareto

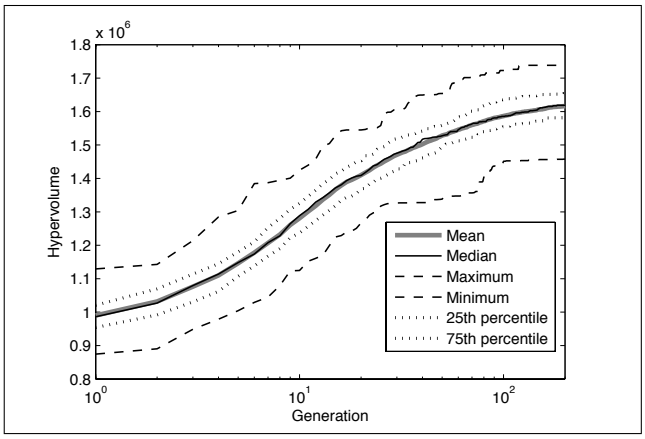

Figure 5. Evolution of the hypervolume.

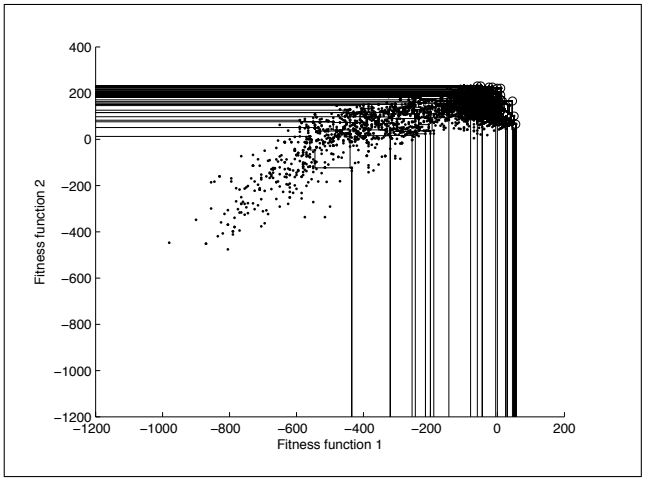

Figure 6. Evolution of the pareto fronts.

front produced from 20,000 randomly generated solutions has an average hypervolume of 1.1833×10^6 (standard deviation 3.0844×10^4).

In order to give evidence of the reliability of the fitness functions, a survey was also done with 12 participants. They were asked to define the quality of 12 harmonies taken from the Pareto fronts of different generations.

The following idea was used to select the harmonies evaluated by the participants. The average hypervolume in the first generation is $g_0 = 9.3421 \times 10^5$ and in the last generation is $g_{200} = 1.8197 \times 10^6$. Therefore, $\Delta = g_{200} - g_0 = 8.8554 \times 10^5$. The generations with average hypervolume values closest to $g_0 + (i-1)\frac{\Delta}{n-1}$ were used as samples in addition to generations 0 and 200, being n the number of samples and i the desired sample. This method was used because the evolution of the hypervolumes happens in logarithmic scale, as it can be seen in Figure 5. By using this method with $n = 4$, generations 0, 6, 16 and 200 (or 0, $10^{0.7782}$, $10^{1.2041}$, $10^{2.3010}$) were chosen. Three harmonies were taken from each of those generations, two of them from the extremes of the Pareto fronts, with good function 1 and 2 values, and the other one from its middle.

The participants were non-musicians and all scores given by them were normalized to a scale from 0 to 10. All harmonies were presented in a shuffled order. They were

Generation	Harm.1	Harm.2	Harm.3	Average
0	2.2	1.8	5.0	3.0
6 ($10^{0.7782}$)	4.3	5.1	3.6	4.3
16 ($10^{1.2041}$)	5.1	5.7	3.0	4.6
200 ($10^{2.3010}$)	8.7	7.7	7.2	7.9

Table 5. Survey about the quality of the solutions

asked to sort the harmonies by order of preference to define a score from 1 to 12 for those harmonies. The scores were defined by the rank of the respective harmony. The average results are shown in Table 5. Harmonies 1 and 3 are the best ones regarding functions 1 and 2, respectively. Harmony 2 represents a trade-off harmony.

The results indicate a small tendency of preference for the harmonies with good values in the simplicity function (Harmonies 1). However, it does not necessarily mean that any of the functions have the ability to represent the harmonization process more effectively than the other, because those numbers might be only showing preferences of the participants. In fact, this experiment can give only evidence of a well defined fitness function and not prove it as it could depend on the specific musical context.

Larger human scores for individuals in latter generations are a good sign that the fitness functions are in fact representing good harmonies. The gap between the average score in the last generation and other generations is also interesting. Harmonies which disrespected only few restrictions imposed by the fitness functions (generation 16) had a very low score (only 84% higher than completely random solutions) while the final solutions had significantly higher scores (320% higher than the initial solutions). Thus, solutions breaking few restrictions were considered almost as bad as solutions breaking many restrictions or even random solutions.

Either way, the higher the average human results are, the higher the fitness function values returned by the algorithm were. That gives some evidence that the codification of the restrictions described in this work represents at least in part successful harmonies while the system is able to work with different preferences of users. After all, these are the goals of the proposed algorithm.

5. CONCLUSION AND FUTURE WORK

The experiments with two functions to evaluate the solutions made clear important aspects of this sort of problem. The implicit knowledge of the system have great influence on the results. The way solutions which are good in relation to only one evaluation function differ from other solutions can demonstrate it very clearly. The multiobjective approach enables the algorithm to ignore particular preferences of the user and generate a set of feasible solutions.

The algorithm converged to many interesting solutions such as the one in Figure 4. This solution is the result of the flexibility built into the system which leads to another trade-off, one between diversity and obedience to certain rules. The last option is not usually useful to composers looking for new ideas. Yet systems with creativity can give

sets with many feasible solutions instead of giving always the same solution excessively based on rules.

Given the main idea of the method, readers can hopefully use their imagination to formulate their own harmonization rules and conciliate different musical contexts.

Methods that would allow the GA to have an idea of how the whole music is meant to be and get any sort of external influence would also probably lead to better contextualized approaches. Currently, most conditions of evaluation of a solution consider only the following chord, at most. It would be interesting to imagine systems that could analyze the measures as a whole before applying genetic operators.

Rule-based models can be more efficient in some specific cases of harmonization, specially when composing in a particular style. Nevertheless, evolutionary systems have clear advantages of flexibility and possibilities of creative new solutions. The multiobjective approach clearly demonstrates how these possibilities of new creative solutions can be utilized. In future work, a better definition of which aspects are preferences and which aspects are rules could lead to a creative evolutionary approach that could also implicitly include rule-based features.

Acknowledgments

This work was supported by the National Council for Research and Development (CNPq) and Coordination for the Improvement of Higher Level Personnel (CAPES, Brazil).

6. REFERENCES

[1] D. W. Corne and P. J. Bentley, Eds., *Creative Evolutionary Systems*, ser. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, 2001.

[2] E. R. Miranda and J. A. Biles, *Evolutionary computer music*. Springer Verlag, 2007.

[3] J. J. Romero and P. Machado, *The art of artificial evolution: A handbook on evolutionary art and music*, ser. Natural Computing Series. Springer, 2007.

[4] S. Todd and W. Latham, *Evolutionary art and computers*. Orlando, FL, USA: Academic Press, 1992.

[5] A. R. Brown, "Opportunities for evolutionary music composition," in *Proceedings of the Australasian Computer Music Conference*, Melbourne, 2002, pp. 27–34.

[6] J. A. Biles, "GenJam: A genetic algorithm for generating jazz solos," in *Proceedings of the International Computer Music Conference*. Citeseer, 1994, pp. 131–131.

[7] ——, "GenJam: Evolution of a jazz improviser," in *Creative Evolutionary Systems*, D. W. Corne and P. J. Bentley, Eds. Morgan Kaufmann, 2001, pp. 165–188.

[8] J. McCormack, "Open problems in evolutionary music and art," in *Lecture Notes in Computer Science, Applications on Evolutionary Computing, EvoWorkshops 2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC*, vol. 3449. Springer, 2005, pp. 428–436.

[9] G. Papadopoulos and G. Wiggins, "Ai methods for algorithmic composition: A survey, a critical view and future prospects," in *AISB Symposium on Musical Creativity*. Citeseer, 1999, pp. 110–117.

[10] R. McIntyre, "Bach in a box: The evolution of four part baroque harmony using the genetic algorithm," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. IEEE, 1994, pp. 852–857.

[11] A. Horner and L. Ayers, "Harmonisation of musical progression with genetic algorithms," in *ICMC Proceedings 1995*, 1995, pp. 483–484.

[12] C. Darwin, *The origin of species*. Signet Classic, 2003.

[13] C. Reeves, "Genetic algorithms," *Handbook of Metaheuristics*, pp. 109–139, 2010.

[14] S. Phon-Amnuaisuk, A. Tuson, and G. Wiggins, "Evolving musical harmonisation," in *Artificial neural nets and genetic algorithms: proceedings of the international conference in Portorož, Slovenia, 1999*. Springer Verlag Wien, 1999, p. 229.

[15] M. Kennedy and J. Bourne, *The concise Oxford dictionary of music*. Oxford University Press, USA, 2004.

[16] J. A. Biles, "Autonomous GenJam: eliminating the fitness bottleneck by eliminating fitness," in *GECCO-2001 Workshop on Non-routine Design with Evolutionary Systems*, 2001. [Online]. Available: http://sydney.edu.au/engineering/it/~josiah/gecco_workshop_biles.pdf

[17] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Parallel Problem Solving from Nature PPSN VI*. Springer, 2000, pp. 849–858.

[18] C. Fonseca, J. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," in *Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, vol. 216, 2005.