Revisiting the 4-part harmonization problem with GAs: A Critical Review and proposals for improving.

Francisco Fernandez de Vega
University of Extremadura
Sta Teresa de Jornet 38
06800 Merida, Badajoz, Spain.
Email: fcofdez@unex.es

***Abstract*—The four-part harmonization problem is a well known problem that has been studied in the last three centuries by music scholars. The goal is to build up three different voices, melodies, based on a previously provided one, being it a soprano melody or a bass instead, so that a complete soprano, alto, tenor and bass (SATB) score is completed.**

The nature of the problem, combinatorial, has attracted interest for decades, and different artificial intelligence techniques have subsequently been applied, such as constraint programing or genetic algorithms. Although researchers employing the first have already stated that the problem is basically solved, and comparisons with GAs typically benefit the first, we think that a critical review of literature may provide useful information demonstrating that the problem is open for improvement, and that GAs still have an opportunity: tests presented in literature frequently employ examples of low difficulty, which provide misleading conclusions.

In this paper we present a review the literature and show that the samples employed to test every available technique are frequently oversimplified; moreover, we have analyzed many of the solutions provided, and have seen how they are not solutions at all, given the number of errors they embody.

Yet, we not only try to show drawbacks of previous approaches. We also try to understand difficulties for GAs when addressing the problem. We analyze the nature of the problem performing a number of tests, and try to see why the standard GA cannot cope with the problem. We propose new approaches that show how GAs could in the future be perfectly capable of addressing large and complex samples, providing solutions of much higher quality.

1. Introduction

Four-part harmony usually refers to music written for four human voices: soprano, alto, tenor and bass (SATB) in a homophonic shape using block chords (three or four sounds conforms a chord). J.S. Bach was one of the first to encode rules to properly build chorales using four-part harmony during the baroque period. Since then, many scholars have developed, changed and improved the rules along centuries. Although a number of techniques and theories could be employed to build different kind of four-part harmony today (such as jazz harmony, barbershop harmony...) classic rules and principles are the basis when practicing four-part harmony for every harmony student nowadays.

Yet, the four-part harmonization problem can also be considered from the combinatorial and search point of view: twelve different sounds (multiplied by the number of octaves available for every voice) can be selected for every single note. Thus, when the melody to be harmonized includes a large number of sounds, the search space may become intractable for an exhaustive search process.

The four-part harmonization problem has thus been a challenge for computer scientists, and different approaches have been employed when facing this problem: constraint programming, markov chains, genetic algorithms, etc. Some researchers have already claimed the advantages of constraint programming, even stating that the problem can be considered as solved. Moreover, some researchers have tried to show that GAs are not appropriate for this kind of problem. Nevertheless, we think that a careful study can provide the opposite conclusions and this paper tries to show why we consider the problem is still open to improvement.

Therefore, this paper thus tries both, to critically analyze results provided in the literature, and show where some of the difficulties for GAs to solve the problems lie, which may be useful to see new opportunities within the area. We provide first data demonstrating that the problems typically addressed in the literature are naive. Moreover, we show that solutions provided in the literature, are no solution at all. Then, we proceed revisiting the problem, their main features, difficulties, and ways for GA to successfully cope with the problem. Although we are still far from a final solution, the experimental results provide useful information for future improvements.

The rest of the paper is organized as follows: Section 2 reviews the literature, analyzing both techniques and solutions provided. Then, section 3 describes the different approaches we are interested in exploring, and some theoretical considerations that may be useful to understand the nature of the problem addressed. Section 4 describes the experiments developed and results obtained, and finally section 5 draw our conclusion.

978-1-5090-4601-0/17/$31.00

2. Is the 4-part harmonization problem already solved?

Although computer assisted music has been a topic present in computer science, and particularly in GAs, from the very beginning [9], naive approaches has been frequently described whose scope is quite limited. In the paper described above, for instance, Horner and Goldberg completely forget required information for properly building melodies, such as duration and octaves for every single note, bars required to organize notes, principles associated to phrases, climax within the melody, etc. The way of building melodies is referred by authors as "thematic bridging" and they argue that first, the method is characteristic of a well known composition approach: *phase or minimalistic* music; and second, it can be easily adopted by a GA. Unfortunately, this kind of statements are frequently found in the literature, and implicitly tries to show that the particular problem addressed, composition in this case, is already solved. Following this way of working, we could thus say the same using an even simpler method: a *random* algorithm can quite directly compose serial and random music, which is a well known approach developed during 20th century by Schoenberg and their students, when computers were not available yet. But this does not mean that random processes are better for music composition. Quite the opposite, we think that this kind of artificial oversimplification of the problem allows random methods to be able to *compose* something. Therefore, in our opinion, exactly the opposite approach must be pursued: classic harmony, although more difficult, should lead the research process, and algorithms must improve to cope with the complex nature of music representation.

One of the previous coauthors published another paper in 1995 [11], but the paper presents a strange conclusions with no methodology provided: Authors state that the GA has been able to solve 4-part harmonization problem -if the chord changes are provided by the user- coping with all of the available difficulties: secondary dominants, modulations, etc. Nevertheless, the two-pages paper doesn't provide any clue on how the genetic algorithm was configured, and only a narrative is provided explaining two versions of the problem, one of them which uses a melody and progression provided by the user as the input of the problem, and the other one, that simply evolves a 4-part harmony. Unfortunately, and given that authors never published extended version of this paper, a doubt remains for any interested reader about the validity of the conclusions and how long did the algorithm take to evolve the hand-written samples included in this strange paper, and ultimately if they were really generated by a GA. Thus, this paper cannot be considered as a source of information, nor a comparison token.

If we focus on the four-part harmonization problem, which has been frequently addressed by researchers. One of the first papers addressing the problem from the GA point of view was published by McIntyre in the nineties [1]. The authors state that the problem becomes tractable if *a melody and a key signature to work with* is provided. Basically a melody or a bass line must be provided as the input for the problem: the standard 4-part harmonization consist of properly building up the three remaining voices. What the author implicitly assumes is that the problem is much easier when the key signature is known before hand, because the size of search space is drastically reduced: instead of looking around 24 different keys (major a minor tonalities) each of them embodying 8 basic chords, one for each of the scale degrees, a single key is employed. Moreover, secondary dominants are not employed nor modulation processes (those in which the key changes along time) applied. In the paper, the author choose *C Major* key. Thousands of individuals and hundreds of generations were required for a simple GA to just harmonize 5 to 9 notes. The algorithm required between one and three hours to perform the task. Not suprisingly, researchers from other areas had thus an opportunity to criticize genetic algorithms when addressing so simple problems.

Another paper published in 1997, followed a different approach when trying to perform a real-time harmonization of a melody [8]. The idea for the approach was to be able to accompany musicians at live performance, so the algorithm must run fast enough to provide accompaniment without detectable delays. A neural network encoding rule-base harmony information approach is developed. Authors train the system using J.S. Bach chorales. Yet, authors recognize the problems associated to real-time and thus have to prune the number of rules considered. And maybe this, together with other simplifications hidden in the model, keeps the method to finding appropriate solutions. The sample show in the paper features important violations of the basic 4-part harmonization rules, such as incorrect chords taking part in progressions; from bars three to four, a standard V-I cadence is wrongly built: B-E-B-D chord, which don't belong to any of the degrees allowed in C Key, is employed before C-E-G-C, (first degree in C key). Again, oversimplification of the problem leads to incorrect results, this time using neural networks and rule based systems.

Again in 2000, an evolutionary approach was described by Wiggins et al. [7] to face musical composition. They perform a review of other approaches, some of them included above, and referred to the 4-part harmonization problem, stating that their aim is to avoid some of the previously described limitation: narrowing the scope of the search to a single key or providing chord progression to more easily find the distribution of notes along the voices. Nevertheless, modulation processes are avoided, and although some of the examples provided include secondary dominants, the resolution of some chords are incorrectly performed, For instance, the first chord in the example incorrectly duplicates fifth: E-G-C-G (first degree in C key). Authors state that the GA cannot solve the problem in 300 generations. In the conclusions, they also recognize that the amount of knowledge provided to the GA will be the key to properly solving the problem.

In [3] Pachet et al. presents a survey of musical harmonization with constraints. Although authors conclude that rule based systems with constraints are the best choice, and

that the problem is already solved, no comments on key changes, modulation processes, and secondary dominants are included in the examples provided. Authors refer to their own previous paper [4], to support their claim, but the constraint based approach is applied to even simpler problems in that previous paper.

In 2001, a comparison between GAs and Rule-based system is presented in [2]. Authors analyze the problem and state that nature of the problem makes it difficult for GAs based techniques to properly address it, given the general lack of specific knowledge included within the algorithm. They conclude that implicit knowledge is very important, and the rule-based system performs much better given the amount of knowledge it embodies. But again, quite simple examples are tested here, such as melodies with just 3-4 bars, and no secondary dominants are shown in the examples. Authors conclude that GA would need extra information for obtaining results of quality.

Researchers have continue actively trying to improve GAs when facing this problem. In 2001, [6] published a new approach to completely generate four-part harmony from scratch. Solutions are found, but again "The chromosome also assumes a key of C-Major and does not contain any key modulations." [6]. This means that no modulations are allowed nor secondary dominants are included in the examples tested. The problem is simpler than the one we are most interested: given that no melody is provided as input, no restriction on how to evolve are imposed on the algorithm, that is free to generate any chord progression.

Finally, we may refer to a recent paper by Kaliakatsos et al in 2014 [5]. Authors state that traditional rules are somehow contradictory, so they need to apply probabilities when selecting rules to apply. Although the technique allows secondary dominants, the problem is somehow simplified: spinal chords, or anchors, are specified by user, so that the harmonization process is helped with this check points; moreover, authors are more interested in chord progressions than in voicing the parts.

Summarizing, we may thus say that although interesting approaches have been performed to solve the 4-part harmonization problems, and even when some claims can be found in the literature stating that the problem is solved, the truth is far from that. Oversimplification has been typically the way of facing the problem, and sometimes, no description of the algorithm can be found. Thus, we can say that the problem is still open, and improvements must be applied if we want to design a better algorithm. In any case, a common agreement on the need of extra knowledge for the GA has been reached. We will see if this knowledge can be obtained from the nature of the problem itself.

In the following sections, we describe an analysis of the problem from the GA point of view, and the path we followed to devise a possible improvements.

3. Methodology: A Divide and Rule based approach to the 4-part harmonization problem.

Before describing the different approaches we tried, let's first consider the nature of the 4-part harmonization problem. The classic method implies to include a number of rules that have been developed by music scholar along centuries, such as: avoiding parallel fifth and eights; applying a proper resolution of the third and seventh in dominant chords; etc. These rules must be used to build the fitness function. The number of rules to be applied may be large, some affecting inter-voicing relationships, other ones related to a single voice development; some to chord progression etc. This rules are well known and can be found in any harmony treatise [10]. Thus the evaluation of a candidate solution requires a large number of tests which takes time. Specifically, we have included these rules within the fitness function to be considered here: (i) avoid criss crossing voices; (ii) avoid augmented forth or major seventh in a single voice; (iv) avoid unison among male and female voices; (v) avoid bad third or seventh resolution in dominant chords; (vi) distances among voices smaller or equal than eighth; exception between bass and tenor (tenth); (vii) avoiding jump in the bass when first inversion chord is employed; (viii) avoid parallel fifth; (ix) avoid parallel eight; (x) avoid chords without the third; (xi) avoid not allowed duplication of note. Our fitness function simply compute the total number of errors found, although other possibilities, such as applying different penalties to every rule, could also be tried.

We could firstly consider the problem as partly separable, given that each of the rules measure something specific and different, although of course, changes to fix a problem in a rule may deteriorate the values for another one. Nevertheless, the application of GAs to this problem has been never considered from this point of view.

On the other hand, if we see that every possible mistake typically affects a given neighborhood (consecutive notes, consecutive chords...) one may think of splitting the melody to be harmonize into pieces; then we could evolve specific solutions to each of the pieces, and finally link all of the pieces in a single chain. There are several reasons aiming at following this approach: on the one hand the search space for every piece of music is much smaller. Moreover, the total size of the sum of all of the search spaces is much smaller than the total size when the whole melody is addressed from a single evolutionary process. The one-max problem is a good problem to compare with, as we will see below.

3.1. The One-max problem can provide some hints

In order to better understand the nature of the 4-part harmonization problem we will focus now on a well known GA benchmark, *the one max* problem. Given a series of bits of size n that are randomly generated, the genetic algorithm is applied to find the maximum: all of the bits set to 1. Although a simple problem, it has been frequently employed

for comparison purposes. We do not experiment with it here, but some considerations on this problem are useful for the one we are most interested in.

If we decide to work with a chromosome of 20 bits for the one-max problem, the size of the search space -total number of possible candidate solutions- is 2048. On the other hand if we split the chromosome in 10 different ones, each with size 2, the search space for each of the pieces is 4, and the addition of all these smaller search spaces is just 4x10=40. This means that splitting the problem into pieces, addressing them separately first and connecting the partial solutions then, would allow us to find the complete solution much quicker when compared with the traditional approach. This is only possible because of the nature of the problem, in which the *Divide and Rule* approach can be perfectly applied. Of course, using parallel models will also allow to reduce computing time, but the main advantage is in the way we reduce the size of the search space.

Unfortunately, although this procedure could also save time for the 4-part harmonization problem, things are here not as easy: given that each piece of music would be evolved isolated, it may well happen that once pieces are connected, new violations of rules appear between the notes belonging to the connecting chords. Therefore the *Divide and Rule* could inspire our approach although will probably not be the solution required. On the other hand, the standard behavior in a GA is the following one: the speed of improvement of fitness quality decreases as the number of generations increase. There are several reasons behind this behavior, but we will focus on the way some of the genetic operators are applied when the max-one problem is being solved.

Lets now consider a partial solution with 19 bits set to 1, and a single bit with 0 in every individual in the population for the one-max problem. This means that the algorithm will not found the solution until a mutation happens in that specific location in at least one of the individuals in the population. With a standard 1% mutation rate, we would need 100 tries over a single specific bit to be sure it will change. But given that in our example there are 20 bits among which the mutation operator must select the bit to be changed, we would need 2000 mutation to be sure that final bit will be switched. Although this may be affordable given the short time required for GA loop in the one-max problem, and particularly the short time required to add up the number of ones in every chromosome, consider what happens instead when the fitness function is time consuming and the loop for a single generation takes a long time, which may be the case for the four-part harmonization problem. As we will show in the experimental stage, even a single violation of the rules, may not be fixed by a standard GA in a reasonable time. This is probably the reason why none of the approaches referred above were able to provide perfect solutions.

In the next section we will perform some test to confirm the nature of the problem we are facing, while also propose new mechanisms to properly cope with it.

3.2. The problem addressed

In order to test our proposal, we decided to use a melody which includes enough components to make the problem hard (real life problem) as our benchmark . We thus explicitly avoided the kind of extremely simple problems that have been traditionally employed in the literature. On the one hand, the size of the melody, 8 bars and 29 notes, is significantly longer than the typical ones employed in the above referred works. We must add that this is a real exercise proposed to harmony students in a Spanish music conservatory. Secondly, the melody includes altered notes that require secondary dominants to properly harmonize the voices. We don't address yet the modulation capabilities of the algorithm, that will be tested in future works. Fig 1 shows the referred melody. Although the key signature refers to F major, we may notice the note *Eb* in fifth bar, which does not belong to the key. Similarly we find in the seventh bar the *B* note, which is also out of F major key. This means that even if we provide the algorithm with the proper tonality to be employed, it must resort to secondary dominants to be able to find a correct harmonization.

3.3. The tests performed

Along the research process we tried different approaches for the genetic algorithm, all of them sharing some basic features. Firstly, the chromosome includes four voices, each of them 8 bars and 29 notes long. The soprano voice is set from the beginning, and the three remaining ones must evolve. The key signature is also set: F Major, but the progression to be applied must also evolve along the run. Therefore information about the chords to be applied for every note position is also included within the chromosome, but oppositely as some of the papers reviewed from the literature, we ask the algorithm to look for a good progression, instead of providing it from the beginning, which adds an extra layer of difficulty.

Given that different configurations of the algorithm were employed, the number of individuals and generations were established so that the total time employed for the run is the same for every run. Thus, the number of individuals employed were 10 or 20, depending on the run, and the total number of generation 30, 50 or 100, depending on the version of the algorithm employed. The idea is to allow all of them to evolve a solution using the same computing time: 6 hours.

The tests included the following configuration of the algorithm: (1) We tried the algorithm to evolve the 4-part harmony working with all the melody at once. We will refer to this approach as the *everything-at-once*; (2) we tried to evolve every bar independently, and after that we joined all of the results giving rise to the new population that is then evolved once again, but now working with the whole chromosome. We call this approach: *evolving-by-steps*; (3) instead of evolving every bar independently, we proceed in a sequential approach: we evolve the first bar; after a solution is found, we add the second bar to this solution,

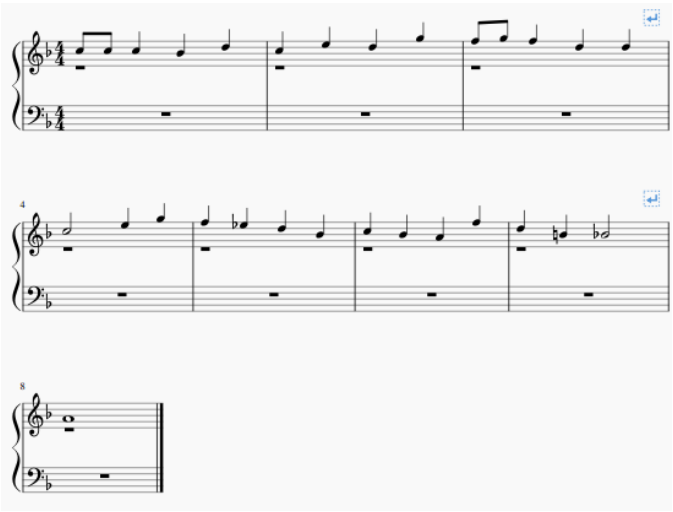

Figure 1. Melody to be harmonized

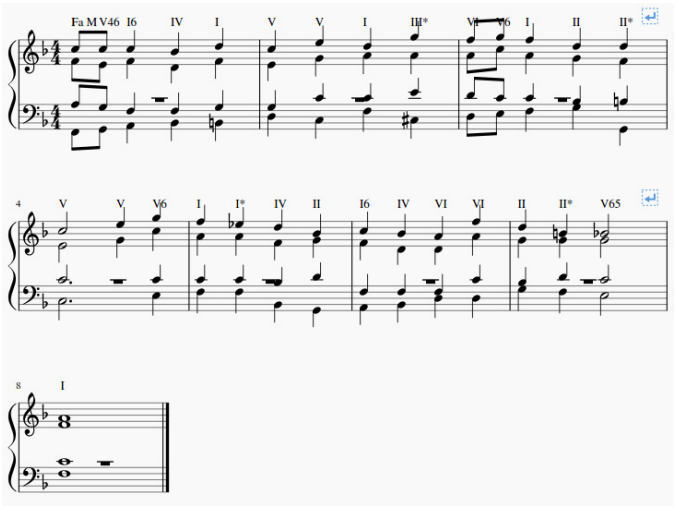

Figure 2. Melody harmonized by a harmony student.

and apply the evolutionary approach to both simultaneously; then the third bar is added, and so on and so for. This is the *incremental-approach*; This is the most time consuming version of the algorithm, given that it must perform as many evolutionary steps as bars includes the test. Thus, the smaller number of individuals and generations are provided, so that the whole experiment is run within the time limit: 6 hours; (4) finally, a test was perform to see the capability of the best of the approaches to fix a couple of problems on a harmonization already provided: one elaborated by a harmony student (see figure 2).

Given that the first step for any of the approaches is to find a suitable chord progression, we decided to apply first an extra evolutionary step that tries to find a good progression matching the melody. This is then applied without changes by all of the models described above. Although we could also consider the possibility of applying changes to the chord progression in the second step, we decided to leave it for future work. We must remark that the evolution of the chord progression is part of every run, and the time consumed to find a good progression is detracted from the total 6 hours available per run.

All of the algorithms were programmed in Lisp, due to its inherent advantages for managing symbolic information. We must say that the main part of the code was required for properly managing music information: music is quite a complex subject, and the western approach to encode it is inherently redundant. This means that a lot of code is required to properly managing a number of symbols that refer to the same physical magnitude: frequencies. For instance, three different notes such as *D, Cx, Ebb*, refers to a single concept. But again, this concept refers to a number of sound frequencies, depending on the octave applied to the note. Therefore, the code required for the evolutionary algorithm may just amount to a 5% of the total code required to properly managing music information.

4. Experiments and results

As described before, every version of the algorithm was an attempt to improve results quality. We will thus sequentially describe what was found with every approach. Although a single experiment was performed for every model, we think it allows us to understand the complexity of the problem and capabilities of the algorithm, which is the main goal here. In future work, a larger set of experiments and statistical information will be computed to better understand differences among models.

Given the large amount of rules that must be applied to every chromosome for the problem at hand, the fitness function is the one which takes longer within the algorithm. We employed the roulette wheel selection, and once the crossover is performed, a mutation was always applied which randomly changed a single chord, exchanging position of notes of the chord in different voices, and possibly the octave applied to each of the notes. Although other possibilities are available, we observed in several tests, that given the size of the melody, low mutation rates implied longer time to solutions. Of course new tests in the future would allow to better set each of the parameters, but results described below allows us to see the possibilities for each of the algorithms employed.

4.1. Progression first

As described above, the first step for each of the algorithms was to evolve a proper progression for the melody. Therefore, in this first step, the evolutionary process is only applied to the chord progression in the chromosome, and not to the whole 4-part harmonization problem. The specific fitness function in charge of analyzing the quality of the progression took into account the following components: (i) avoiding consecutive repetitions of the same degree; (ii) avoiding dominant chords without a seventh, followed or

TABLE 1. EVERYTHING-AT-ONCE APPROACH

Generations	Individuals	time (hours) to solution
100	20	4
150	20	5.5
200	20	7
300	20	10

TABLE 2. RUNNING EXPERIMENTS FOR 6 HOURS

Experiment	Individuals	Generations
Everything-at-once	20	150
By Steps	20	50
Incremental	10	30

proceeded by the same dominant with the seventh; (iii) avoiding harmonic syncopation; (iv) including perfect cadence at the end; (v) looking for good transitions, such as V-I, II-V, IV-V etc.

Given that this first step was exactly the same for all of the algorithms, no significant differences in the final results must be due to it. We were able to find solutions perfectly complying with the rules established, regardless of whether they would be preferred or not by a human musician. Solutions included correct employment of secondary dominants. In any case, new rules could be added to even improve the progressions. The conclusion is that the problem of looking for a good progression fitting the melody is not a fundamental one here.

4.2. The everything-at-once approach

Our first attempt to solve the problem simply tried to evolve the 4-part harmonization working with the whole chromosome, 8 bars, simultaneously. This first approach was the test to see the time required for evolving a number of generations and individuals in the population, and also test the quality of solutions found. Table 1 shows different configurations of the algorithm and time to completion. We thus decided to allow the other versions of the algorithm 6 hours to run. Therefore some test were performed to decide how many individuals and generation were appropriate for every experiment to take 6 hours. Table 2 shows the main parameter for the approach we describe here, as well as the remaining ones that will be described below. Some final adjustments were applied to the number of generations and individuals applied during the evolution of the progression for every experiment, so that the total computing time was the same.

The best of the results was obtained with the latest configuration: it took 6 hours to reach a best fitness value of 15, which is not a bad result given the size of the chromosome, although we would like to reach a perfect harmonization.

TABLE 3. EVOLVING-BY-STEPS APPROACH

Bar	Errors after evolution
1-2	8
3-4	6
5-6	6
7-8	3
1-8 linked	80

4.3. Evolving by steps: Divide and Rule?

The basic *Divide and Rule* approach was applied by splitting the whole chromosome in as many pieces as the number of bars in the melody. Being aware of the problem that may arise when joining into a single solution pieces that have been solved isolated, we decided to divide the chromosome into 4 pieces, each of them including 2 bars, so that the number of links is reduced to just 3, instead of 7 if we had employed 8 pieces.

After applying evolution to each of the sub-chromosome, we linked all of the partial solutions. We show in Table 3 the number of errors present within each of the bars after their partial evolution, and finally the total number of errors when solutions are linked.

Even when the addition of errors found in partial solutions amounts to 23, which is not smaller than the best value found with the *everything at once* approach, the real number of errors arise when the fragments are linked, and the whole chromosome analyzed: 80 errors. Therefore, we have confirmed that the simple *Divide and Rule* approach doesn't work properly in this problem, as anticipated. And the reason comes from what has been already analyzed: there is a connection among pieces that affect results, quite the opposite as with problems which are similar to the *one max* previously studied.

We thus decided to try an additional step: after obtaining the partial solutions, we applied an extra evolutionary step to the linked chromosome. Still, the improvement was poor: running the algorithm again for 50 generations only allowed to improve from the initial 80 errors to 77. In our latest attempt with the model, we run a new experiment to see what happens when every bar is evolved independently. In this run, after every bar was evolved and linked with the other ones, the fitness value computed was 39, and once the evolutionary algorithm applied again to the whole chromosome, we reached a fitness of just 20 which is not a bad result given the size of the chromosome and is near the value obtained by the previous approach. So, in our test, splitting the chromosome didn't provide better results than the traditional approach on the problem at hand. Yet, given that an improvement is found as the chunks employed are smaller, we could hypothesize, that maybe smaller pieces could be used, with just a couple of chords per chunk. In any case, 20 is still far from what we would like to have: a perfect solution for the problem.

TABLE 4. EVOLVING-BY-STEPS: COMPARING SIZES OF PIECES

Number of Bars/piece	Number of pieces	Best fitness
2	4	77
1	8	20

4.4. The incremental approach

We build up this new approach using the principles learned: dividing chromosomes in as many pieces as bars. But now, instead of linking all the pieces together and apply an extra evolution stage, we decided to proceed incrementally: the algorithm follows a number of steps, as many steps as pieces we decide to split the chromosome in. In the first step, the first number of bars selected from the chromosome are evolved. Once this step is concluded, the second piece -as it is- is added to all of the individuals of the previous population, so that we have the same number of individuals, but now with an extra number of bars; and then, these population is evolved again. The process repeats adding the third piece from the melody, and evolution is applied, and so on and so for until the whole chromosome is finally evolved together.

We designed the algorithm to last the same number of hours as in the previous experiments, employing 30 generations and 10 individuals per evolutionary stage. These parameters were selected so that the total time for the 8 evolutionary stages lasted 6 hours.

Table 5 shows the results obtained. The first thing we notice is that the final value reached, 65, is much worse than the result obtained with the previous approach. On the other hand, we may notice that every time a bar is added to the previously evolved piece of music, fitness value grows a lot, as we already expected. But we thought the evolutionary process would be able to properly improve the new bar added in conjunction with the previously evolved part. If we check the table, we see that during the three first bars things are kept under control, with small increases in the fitness value when the linking of bars is performed, together with improvements after evolution with final fitness values that are not far from those obtained in the previous step. Nevertheless, once we reach bar #4, we cannot reach fitness values under 10. If we look to the final step, when bar #8 is added and the last evolutionary process is applied to the whole chromosome, the improvement goes from 79 to just 65, which is worse than the improvement attained with the simple *evolving-by-steps* approach. It seems that given that the time for the whole experiment is the same as in the previous cases, and given that the size of the chromosome grows every step, the number of generations available for the final steps are not enough to find good solutions.

Although results reported for each of the approach correspond to a single experiment, similar results were obtained for another series of experiments.

TABLE 5. EVOLVING-INCREMENTALLY: COMPARING SIZES OF PIECES

Bars	C1	C2	C3	C4	C5	C6	C7	C8
Initial Best Fit.	5	20	26	45	44	74	65	79
Final Best Fit.	2	6	10	22	18	32	38	65

4.5. Trying to fix a single problem

So we have seen how all the approaches tested confirm the difficulty of the problem we face. We thus decided to run an additional test: trying to simply apply evolution to fix the couple of errors present in a 4-part harmony solution developed by a music student (see figure 2). A careful analysis shows some violations of the harmony rules. For instance the second chord in the second bar includes the third of the chord (E) in the soprano voice, that indirectly resolves into tonic, but in the tenor voice (C), instead of resolving in the same voice. The problem comes from the progression selected by student: V-VI. Something different should be tried, given that the melody is the input and shouldn't be changed. Anyway, it allows us to test the capability of the GA to fix a single problem. Thus we allowed the algorithm -the incremental version- to change any of the voices when trying to provide a perfect harmonization. After 6 hours of evolution, the algorithm was not able to provide a solution with errors fixed.

Thus, this latest test shows us the problem already described for *the one max problem*: A large amount of generations must be computed to have an opportunity for the mutation operator to switch a chord. Things are even worser in our problem: while for the one max only two possibilities are available for a position of the chromosome, 1 or 0, here a number of combinations of notes for a single chord are possible; for instance the V7 chord in F major includes 4 notes: C-E-G-Bb. Given that 4 notes are available, 24 different permutations are available. If we also take into account *octaves*, things are much worse. Therefore, it is hard for mutation operators to find the desired combination quickly.

4.6. Longer runs

Although certainly we would agree with some authors about the need for specific knowledge to help GAs to be able to find solutions quicker, we wanted to know if some specific barrier keeps the standard GA to find better solutions by itself.

Thus, despite the difficulties found above, we decided to launch some final tests with longer runs (24 hours) for the two methods that provided worse results, to see if GAs had found an upper limit in their possibilities, or they still can progress if longer evolution times are allowed. Table 6 shows the results obtained for a couple of experiments. If we compare the value obtained for the incremental approach, with that obtained with a shorter run of only 6 hours, we notice an important improvement: we go from 65 (in 6 hours) to 40 in 24h, a long time to get a value which is still

TABLE 6. DIVIDE AND RULE 24 HOURS EVOLUTION: 50 INDIVIDULS, 100 GENERATIONS BY STEP.

Bars	1	2	3	4	5	6	7	8	Final Best
Best Fit.	1	4	2	0	0	0	0	0	10

TABLE 7. PROVIDING LONGER TIME TO EVOLVE THE 4-PART HARMONY

Method	Best Fitness
Incremental	40
Divide and Rule	10

far from our goal, but we have confirmed that the GA still have room for improve results if enough time is provided.

On the other hand, if we look to results obtained in the second experiment (*divide and rule* table 6), a quite impressive result was obtained: just 10 errors in the exercise. Moreover, 4 of the bars, when independently evolved, obtained a 0 score, which means that the evolutionary algorithm as it is, without any problem specific information added, was capable of finding a perfect 4-part harmony for some of the bars.

Summarizing, we have confirmed that the GA have not reached yet the upper limit of their capabilities for addressing this problem for any of the methods implemented. The quality of results obtained, 10 error over 29 chords, are of similar quality to the best previously described in the literature for smaller problems. Yet, while quite simpler and non-realistic problems are frequently employed in the literature to study the capability of other algorithms, a hard real-life problem has been considered here with a large chromosome size, 29 notes to be harmonize, that includes altered notes and requires secondary dominants to solve the problem.

Two of the approaches presented, *everything-at-once* and *divide-and-rule*, have provided the best results, although still not perfect ones. Nevertheless, we think that a refinement of the standard genetic operators, as well as adding specific information from the problem, will allow the GA to perfectly cope with real-life hard 4-part harmonic and be competitive with other techniques available.

5. Conclusions.

This paper tries to shed some light on the 4-part harmonization problem, specifically when addressed by means of Genetic Algorithms. We have seen that previous approaches typically employed simple and unrealistic problems. Therefore some of the conclusions reached, must be put into question, such as claims about the capabilities of some algorithms to perfectly solve the problem.

We have analyzed the difficulties of GAs for finding solutions, considering possibilities for splitting the chromosome into pieces and trying to evolve partial solutions. After applying different versions of the algorithm, we have seen that although partial evolution of pieces may help, the basic *Divide and Rule* does not work for the problem, and an extra evolutionary step must be applied after linking partial solutions together. Nevertheless, perfect solutions has not been found yet for the hard benchmark problem employed, a real 4-harmony exercise for harmony students, although promising results were reached.

A specific test has been applied trying to see if the evolutionary process can fix a single violation of harmony rules. The negative results help us to see that the standard mutation operator does not work properly here. Something different must be tried in future work, such as adding information on error positions that may help the mutation operator, and therefore the whole GA to find solutions.

Finally, a couple of long runs were performed, to see if different versions of the GA can still progress towards better solutions. We have found that after 24 hours the *Divide and rule* approach, of course followed by an extra evolution step, have found a solution with just 10 errors, which is quite promising.

Acknowledgments

This work has been supported by Spanish Ministry of Economy, Project UEX:EPHEMEC (TIN2014-56494-C4-2-P); Junta de Extremadura, and FEDER, project GR15068.

References

[1] McIntyre, R. A. (1994, June). 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 (pp. 852-857). IEEE.

[2] Phon-Amnuaisuk, S., and Wiggins, G. (1999, April). The four-part harmonisation problem: a comparison between genetic algorithms and a rule-based system. In Proceedings of the AISB?99 Symposium on Musical Creativity (pp. 28-34). London: AISB.

[3] Pachet, F., and Roy, P. (2001). Musical harmonization with constraints: A survey. Constraints, 6(1), 7-19.

[4] Pachet, F., and Roy, P. (1995). Mixing constraints and objects: A case study in automatic harmonization. In Proceedings of TOOLS Europe (Vol. 95, pp. 119-126).

[5] Kaliakatsos-Papakostas, M., and Cambouropoulos, E. (2014). Probabilistic harmonization with fixed intermediate chord constraints. In ICMC.

[6] Donnelly, P., and Sheppard, J. (2011, April). Evolving four-part harmony using genetic algorithms. In European Conference on the Applications of Evolutionary Computation (pp. 273-282). Springer Berlin Heidelberg.

[7] Wiggins, G., Papadopoulos, G., Phon-Amnuaisuk, S., and Tuson, A. (1998). Evolutionary methods for musical composition. Dai Research Paper.

[8] Spangler, R. R., Goodman, R. M., and Hawkins, J. (1998). Bach in a box-real-time harmony. Advances in Neural Information Processing Systems, 957-963.

[9] Horner, A., and Goldberg, D. E. (1991). Genetic algorithms and computer-assisted music composition. Urbana, 51(61801), 437-441.

[10] Piston, W. (1948). Harmony. Norton.

[11] Horner, A., and Ayers, L. (1995). Harmonization of musical progressions with genetic algorithms.