Paradigmatic Analysis using Genetic Programming

Carlos Grilo[1, 4]; Fernando Machado[2, 4]; Amilcar Cardoso[3, 4]
[1] Departamento de Engenharia Informática da Escola Superior de Tecnologia e Gestão de Leiria;
Morro do Lena, Alto Vieiro, 2401-951 – Leiria, Portugal
[2] Instituto Superior de Engenharia de Coimbra; Quinta da Nora 3031 – 601 Coimbra, Portugal
[3] Departamento de Engenharia Informática da Universidade de Coimbra;
[4] Centro de Informática e Sistemas da Universidade de Coimbra;
Polo II, Pinhal de Marrocos, 3030 – Coimbra, Portugal
grilo@dei.uc.pt; machado@dei.uc.pt; amilcar@dei.uc.pt

Abstract

Paradigmatic analysis consists in the segmentation of a musical piece through the identification of relations between different parts of the piece, and the classification of the identified segments into categories. In this paper we describe how a genetic programming system can be used to make the paradigmatic analysis of monophonic musical pieces, using a simple fitness function inspired in the Kolmogorov complexity estimation. We make use of automatically defined functions in order to represent segments. Relations are made explicit through the reuse of segments and the application of transformations to these segments.

1 Introduction

The segmentation of musical pieces and the clustering of different segments into meaningful categories are two key elements for the understanding of music. People do both these things through the identification of similarities and contrasts between the different parts of the musical piece they are listening, although they may do it non-consciously. Also, even if they consciously identify related segments, the criteria used to identify, compare and associate these segments may not be explicitly formulated. If 'explicitation' is not crucial to ordinary listeners, that is not the case in musical analysis, in which it is important that these criteria are objectively and explicitly defined in such a way that other analysts can better understand and criticize analyses already done.

Paradigmatic analysis (PA) is a formal approach to musical analysis, which intends to free the analysis process from subjective criteria. It comprises the segmentation of a musical piece through the identification of relations between different parts of the piece, and the classification of segments into categories according to the existing relations. This paper describes an ongoing research work in which we study how a genetic programming system can be used to identify the different segments of a monophonic musical piece and the relations among them. The relative quality of each segmentation is measured using a simple fitness function, inspired in the Kolmogorov complexity estimation, which guides the search through the space of possible segmentations. This work is part of a larger project named SICOM (Pereira *et al*. 1997), in which case-based reasoning is applied to musical composition. In SICOM each case is an analysis of a previously composed music that can be used, in conjunction with others, to create new musical pieces. This part of the project consists in the construction of one system able to fill this case base.

We start, in section 2, by describing PA and some previous work done in this area. A brief overview of GP is done in section 3. Our approach is then described in section 4 and some first results are presented in section 5. Finally, in section 6, we draw some overall conclusions and indicate the future steps in our research.

2 Paradigmatic Analysis

PA (Nattiez, 1975) is based on an analysis procedure proposed by Ruwet (Ruwet, 1972) in an attempt to systematize the musical analysis process. This procedure, which relies mainly on the notion of repetition, divides a musical piece into a set of segments, and classifies them into categories according to their similarity. One central idea of this type of analysis is that these tasks are done without considering the composer's intentions, nor the perceptions or interpretations of the listener/analyst, so that some scientific objectivity can be achieved.

The procedure consists in two main operations. The first one identifies segments that are exact repetitions of previous segments. This operation derives structures like, for instance, A + X + B + A + B, in which segments that are repeated through the piece are represented with the first letters of the alphabet, and segments that are not repeated are represented with the last ones. The second operation consists in the identification of relations other than repetition between the different segments. Two segments are related if one can be described as being a transformation of the other. This operation may modify the previously obtained structure in order to reflect these relations. The structure

A + X + B + A + B may be modified, for example, to A + A1 + B + A + B, or even to A + A1 + A2 + A + A2, where A1 and A2 are transformations of the A segment. These operations comprise an association of the identified segments into categories. If the structure derived is A + A1 + B + A + B + X, then segments A and A1 form one class, and segment B will form another one. Segment X is not associated to any class because no relation has been found between it and the other segments. Segments that don't fall in any class are called *rests*. After the segmentation of the whole music, the same procedure may then be applied to the resulting segments in order to identify smaller segments, and so on, until no further segmentation can be done. The final result can thus be represented as a tree in which longer segments are closer to the root.

This procedure assumes the existence of some kind of mechanism with the capacity to identify repeated segments through the piece. Moreover, it states that, before the analysis is started, all the transformations that can be used to describe relations between segments must be defined, as well as the mechanisms that allow the identification of these transformations.

Although one of the main initial goals of PA was to free the analysis process from subjective criteria, it has been criticized because, in practice, both segmentation and consequent classification usually rely on the analysts' intuition (Cook, 1987). Another problem, reported in (Anagnostopoulou and Westerman, 1997), rises due to the existing bias towards considering the first segments appearing in the music as the *paradigms*, i.e. the most representative segments against which other segments are compared in order to form classes of segments. Sometimes these first segments are not so representative, which leads to difficulties both in the segmentation and classification. In (Anagnostopoulou and Westerman, 1997), the classification and the *paradigm* problems are addressed by using a self-organizing neural network, which groups already identified segments into clusters (categories) according to some features explicitly chosen by the analyst. The segmentation task is addressed in (Smaill *et al*. 1993), using a simple algorithm, based on the Ruwet's procedure, that strongly resembles our description above: first, the piece is scanned for exact repetitions and then, other relations are identified. In (Cambouropoulos, 1998) a model is proposed for segmentation based both on the identification of local discontinuities like big intervals, pauses, or longer notes (Cambouropoulos, 1996), and on the identification of similarities between different parts of one piece. Similarities between segments are identified using a brute-force pattern-matching algorithm, the *Sequential Pattern-Matching Algorithm* (Cambouropoulos, 1995), which identifies all possible matches existing in a musical piece. Then, the boundaries of all the identified segments and the identified local discontinuities are combined in order to establish if one particular point of the piece is, or not, a segment boundary.

3 Genetic Programming

Genetic programming (GP) is an evolutionary computation technique for the automatic generation of computer programs (Koza, 1992). The GP process may be concisely described as follows:

1. Randomly generate an initial population of individuals (i.e. programs);
2. Evaluate the current population using a fitness function that measures the quality of each individual;
3. Stochastically select the best individuals;
4. Apply genetic operators (e.g. recombination, mutation) to the selected individuals, thus generating a new population;
5. Substitute the old population by the new one, and then repeat the same process from step 2. The evolutionary process is stopped when a pre-specified halting condition is met.

From the description above, one can see that GP is quite similar to other evolutionary computation approaches. In fact, the same description could be used to describe a Genetic Algorithm. The main distinguishing features of GP are related with the representation of the individuals and with the fact that these individuals are computer programs. In GP the individuals are, typically, represented as trees. The internal nodes of the trees are functions belonging to a function set, and leafs are terminals belonging to a terminal set. The choice of appropriate function and terminal sets is, probably, one of the most important tasks in the development of a GP system, since it establishes the basic building blocks that will be available for the generation of a solution. As in GP the individuals are computer programs, one must execute them, in order to assign fitness. Fitness assignment is one of the key issues in GP, as in any evolutionary computation technique, since the evolutionary process is guided by it.

In order to generate a new population one applies genetic operators to the selected individuals. Usually, three types of operators are involved: reproduction, recombination and mutation. The reproduction operator merely makes a copy of a selected individual to the next population. The idea of the recombination operator is to generate new children through the exchange of genetic material of two parents. In GP the most popular recombination operator is the crossover operator introduced by (Koza 1992), which can be described as follows: randomly select two sub-trees (one from each parent), exchange the sub-trees. The idea of mutation is to randomly introduce new genetic material. In GP this is usually achieved by replacing a sub-tree by a randomly generated one. It is important to notice the roles of these operators: reproduction is merely a way of ensuring that some of the individuals are preserved; the

idea behind recombination is the exploitation of genetic characteristics that are already present in the population; finally, the role of mutation is to explore new regions of the search space.

As the number of generations increases the average fitness of the individuals tends to improve. Eventually, provided that enough time is given, the optimal solution will be found. One of the problems is that, in certain cases, there is no way to know when an optimal solution was found. Therefore, the algorithm is usually stopped when a pre-defined number of generations or fitness value is reached.

In (Koza, 1994) one important feature of GP was introduced, that allows a better exploitation of space regularities: the Automatically Defined Functions (ADFs). When ADFs are used, each program consists on a main function and a set of subroutines, the ADFs, which are simultaneously evolved. This feature is important to us, since through the use of ADFs we may reuse musical stuff in order to describe relations between segments located in different parts of the music.

4 Paradigmatic Analysis using Genetic Programming

In this section we explain our approach to PA, more specifically, to the segmentation and identification of relations between segments in monophonic musical pieces. GP is the mechanism used to do both these tasks. The goal is to find a program that generates a given musical piece and, which at the same time, is a description of that piece, i.e. shows what segments exist in the piece and which ones are related and how. One peculiarity of our approach, not very common between other works using GP, is that although the program result must be similar to the given music, we are not interested in that result by itself. What is really important here is the ordering and content of the various segments, and the functions applied to each of them, i.e. the relations among them.

Hence, each program in the population generates, as output, a string of notes. Moreover, program trees represent segmentations of the generated strings. Programs are composed of a main function tree describing relations between segments, and a set of ADFs that represent specific segments. The main function tree is built using a set of primitive functions that, beyond putting the different segments together, also describe melodic, rhythmic or other transformations that portray relations between segments. One of the most important functions used is *conc*, which receives, as arguments, two segments and gives, as a result, their concatenation. Besides putting segments together, it allows repetition to be clearly shown when one segment is used more than once. Examples of other functions are *up/down* transposition and *reverse*.

The leaves of the main function tree are ADFs without arguments. Each ADF consists only of one terminal symbol, a segment, represented as a pair [*l*, *r*], where *l* and *r* are, respectively, the left and right limits of the segment. When an ADF is called, it returns the string of notes from the original piece corresponding to the interval specified by [*l*, *r*]. This string will then be used in the main function tree so that it can be concatenated with other strings in a bottom-up fashion until the all piece is complete. Figure 1 shows a simple string of notes and one individual representing a segmentation of it.

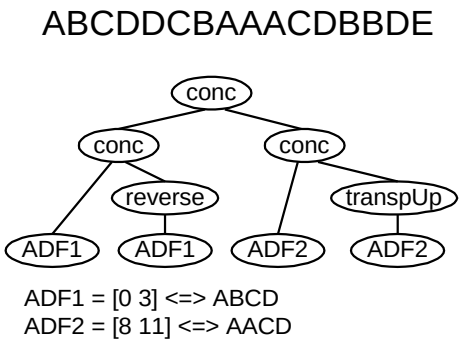

Figure 1: Example of one individual representing a segmentation of a simple music.

There are three main reasons why ADFs are used. First, ADFs allow a better exploitation of space regularities, as was referred in the previous section: in our case, we are interested that the most frequently occurring patterns are identified and kept so that the same individual can easily reuse them. Second, ADFs provide an elegant way of showing that two different parts of the piece are related. The third reason is that, if ADFs were not used, extra processing would be needed in order to detect if one particular segment was repeated throughout the piece since, it would have to be compared with all the other segments. This last aspect is important in the calculation of the fitness function.

The quality of each program is measured taking into account (i) the difference between the program output and the target piece, and (ii) the number of notes from the target piece really used to produce that output. Fitness is inversely related with these values. Hence, the goal is to minimize the value of the fitness function, which is described by the following weighted sum of two terms

$$F(x) = a \times d(m,t) + b \times \sum_{i=1}^{n} l(ADF_i)$$

where *x* is the individual to be evaluated, *a* and *b* are constants, *d*(*m*, *t*) is the difference between the program output *m* and the target music *t*, *n* is the number of ADFs of each individual (all individuals have the same number of ADFs) and

$$l(ADF_i) = \begin{cases} r_i - l_i + 1 & \text{if } ADF_i \text{ is used in the main function} \\ 0 & \text{if } ADF_i \text{ is not used in the main function} \end{cases}$$

represents the length of each segment, from de target piece, used in the main function tree.

The difference $d(m, t)$, between the program output and the target music is calculated with the Wagner-Ficher algorithm (Stephen, 1992), which measures the distance between two strings by the number of operations of insertion, deletion and substitution needed to transform one in the other. In the current version of the system, melodies are represented as strings of notes, each one described by its pitch and duration with the structure {{*Name of the note, Accident, Octave*}, *list of basic rhythmic figures*}, (Smaill e*t al.* 1993). The comparison between notes needs not to cover all the dimensions by which they are described. For instance, if only a rhythmic analysis is intended, then only notes duration must be considered. Also, other representations for melodies can be used as well: it can be used, for example, intervals between notes or contour (up, down, equal), depending on the goals of the analyst.

The second term sums the length of the segments used in the main function. Its important to note that even if one ADF is used more than once in the main function, the length of the segment it stands for is only summed once. This allows that those individuals that are able to produce the target music using fewer notes are considered the best ones.

Our idea can be viewed as a kind of Kolmogorov complexity estimation, in which the goal is to find the smallest program that produces a given string (Conte *et al*. 1997). Here, we are not really looking for the smallest program. Instead, we want to find the program that produces the target music, or a very close one, maximizing the reuse of musical material. For example, consider the very simple string of notes CCCCGGGGCCCC, and the two individuals with the same ADFs, in Figure 2, describing it.

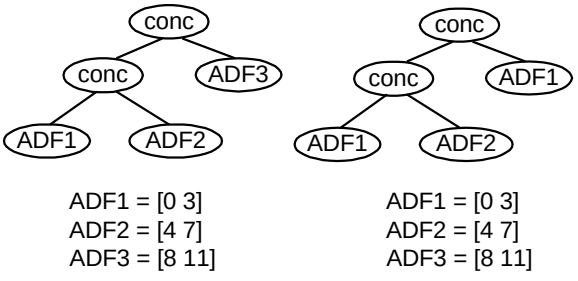

Figure 2: Two different individuals with the same ADFs describing the same music

The individual on the right side is, according to the fitness function, better than the one on the left because it uses fewer notes from the target music or, saying it another way, it does a better reuse of the existing segments.

This fitness function does not use musical knowledge to measure the quality of each individual. We do not claim that this function is sufficient, by itself, to lead the system to the best segmentations, or that the use of other methods cannot enhance the quality of some segmentations. On the contrary, in some situations it may happen that an individual with a fitness value slightly worst than other one can represent a better segmentation from the point of view of a human analyst. Another problem that may happen, as we will see in the next section, is that sometimes the boundaries of the identified segments may not correspond exactly to the boundaries of the segments that a human analyst would identify. In these situations, it is evident that methods based on the detection of local discontinuities like the one in (Cambouropoulos, 1996) could be of great help. However, our experimental findings indicate that this fitness function is able to lead to very acceptable segmentations. These segmentations, when not perfect, can be viewed as first versions, which can be refined latter by other methods. Alternatively, we could incorporate in the fitness function one term that would measure the quality of segment boundaries: those corresponding to bigger discontinuities would contribute to a better fitness value.

5 Experimentation

In this section we show the results obtained after some experiments made with the troubadour songs «Be m'anperdout…» (Figure 3) and «Maria muoter reinû maît» (Figure 4), used by Ruwet (Ruwet, 1972) to exemplify the PA procedure.

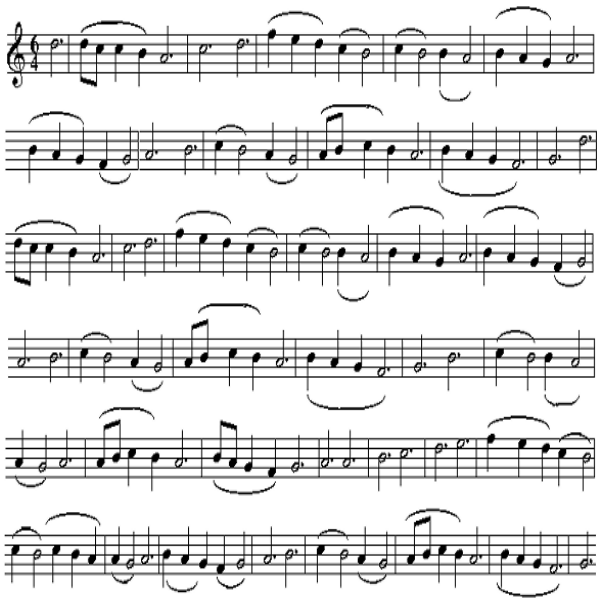

Figure 3: «Be m'anperdout»

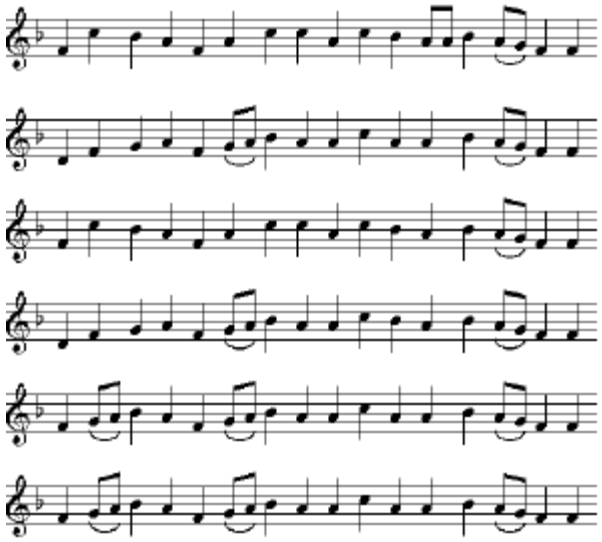

Figure 4: «Maria muoter reinû maît»

We made 30 runs for both pieces and in each run we used populations of 3000 individuals evolving during 100 generations. Standard reproduction, crossover and mutation operators were used with rates, respectively, of 10%, 60% and 30%. Each individual was composed by 6 ADFs and only the *conc* function was used in the main function since only relations of repetition or "almost repetitions" exists in these pieces. The termination criterion is the number of generations since it is impossible to know, in advance, what is the fitness value corresponding to the best analysis.

After some preliminary experiments, the values of the fitness function weights were established as follows: 5 to the first term and 1 to the second. This is a sensible choice since if the proportion between the first and second terms is too large, the system tends not to reuse segments; if it is too small it tends not to approximate program results to the original music. We also define a limit to the error *e*, equal to 5% of the number of notes of the given music, so that individuals with an error greater than this limit cannot be considered as acceptable segmentations.

In 28 of the 30 runs made with the piece «Be m'anperdout» we obtained individuals with the same structure of the one in Figure 5, producing a string of notes equal or almost equal to the target piece (1 to 3 notes of error), using only 99 or 100 notes of 142. All these individuals have the same structure of the segmentation made by Ruwet: A + A + X. The difference is that, although he considers the last segment as a *rest* X, he subsequently subdivides it into three other ones. The two last ones, together, correspond, with some differences, to segment A without the first 5 notes. The error between the music produced by some of these individuals and the target music is due to some misplaced segments boundaries. In these cases, as we have already mentioned, it would be helpful to use local search methods so that these situations could be corrected or avoided. The difference between the output of the other 2 individuals and the given music is greater than *e* (7 notes to this piece), and so, they are not considered.

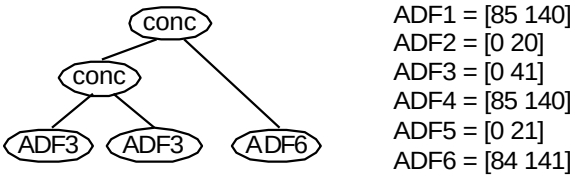

Figure 5: Example of a segmentation of «Be m'anperdout»

Figure 6 shows two analyses made by the system to the piece «Maria muoter reinû maît». In 16 of the runs we obtained individuals equivalent to the first one, which are the best ones, corresponding to the first level segmentation made by Ruwet, with a fitness value of 69. In other 11 runs we obtained individuals like the second one with fitness of 88. These individuals are not so good as the first ones because the segmentation is less detailed. Allowing a higher number of generations could, possibly, solve this problem. The other 3 individuals have the same structure of the first individual in Figure 6 but the difference between the output and the given music is greater than *e* (5 notes to this piece).

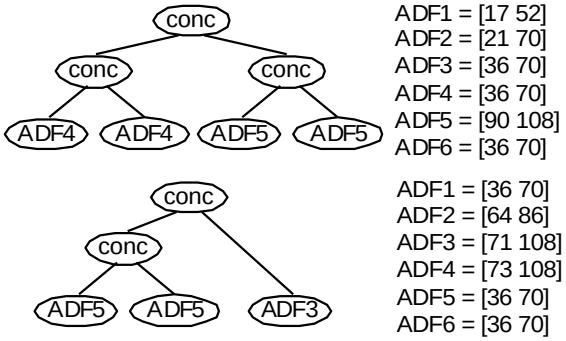

Figure 6: Examples of two segmentations of «Maria muoter reinû maît»

It can be seen in the first individual of Figure 6 that neither ADF4 nor ADF5 corresponds to the first occurrence of the segment in the original piece. Although this may be strange at a first glance, it is actually an advantage, since it is a way of solving the *paradigm* problem (see section 2), as the system tends to choose the more representative segments in order to minimize the error. Another important question concerning this example is that, although it is a good segmentation, the music produced by this individual is different from the original music in 3 notes. In this case, this is due to the fact that, in the original music, the first occurrence of the A segment (here represented by ADF4) is not exactly equal to its second occurrence. Ideally, the individuals should describe one of the segments as a transformation from the other one.

However, to attain this, we would have to include a special function in the function set which we could only describe by "insert a note in position ten and then divide the rhythm of note eleven and twelve by two". Instead of dealing with a complex function set with very specific functions like this one, which would drastically reduce the performance of the system, it is preferable to allow some error on the output, given that it is not greater than the error limit *e*; if the error happens to be systematically too large, no good analysis is possible and the function set must be revised.

The pieces here discussed, like many others, have a hierarchical structure. This means that they are composed of bigger related segments that are also decomposable in smaller segments. The results of the experiments that we have already done lead us to conclude that the system has a strong tendency to first identify the longest segments existing in one piece, i.e., to do the higher-level segmentation. The question of how to continue a first segmentation, in order to identify smaller segments, still is an open problem in our work. One obvious possibility consists in the application of the same process of segmentation to the segments already identified. However, this approach can, in some situations, be fruitless because the existing segments may not be composed of related sub-segments. As an example, suppose that one piece has the structure A + B + A and that A = *a* + *b* and B = *c* + *b*. If we try to analyse the two segments independently we will be unable to identify segments *a*, *b* and *c*, because *a* is not related to *b*, which is not related to *c*. A human analyst would have no problems with this approach because he/she can always take a look to other segments while working on a specific segment, in order to identify common sub-segments. Based on this idea, we sketched an approach that consists in the application of the segmentation process again to the entire piece, but now imposing the following restrictions: only segments that fall within the previously identified segments can be generated; individuals that use more notes than the ones used in the previous segmentation (with a higher second term in the fitness function), or that use a equal or fewer number of segments, are strongly penalized. We applied this approach to the piece «Be m'anperdout…» using, as previously identified segments, the ones corresponding to ADF3 ([0, 41]) and ADF6 ([84, 141]) in Figure 5. Also, individuals with less than 4 segments and using more than 100 notes from the original music were strongly penalized. In Figure 7 we show two of the best individuals that resulted from these experiments. The first one has an error of 4 notes and uses 66 notes from the original music, which corresponds to a fitness value of 86. The second one has no error and uses 78 notes from the original music, thus having a fitness value of 78. Although both individuals are similar to the segmentation done by Ruwet, the first one is closer to it. This is one of the cases where the best individual, from the point of view of a human analyst, has a worst fitness value. In this case, this happens due to the errors between the music produced by the first individual and the original one. One way of solving this problem is, once more, to include in the function set a function that better reflects the transformation between the two segments (those corresponding to the first and last occurrences of ADF2 of the first individual of Figure 7), so that less or no errors exist. Another approach is to ignore the error between the two pieces when this error is smaller than the limit *e*.

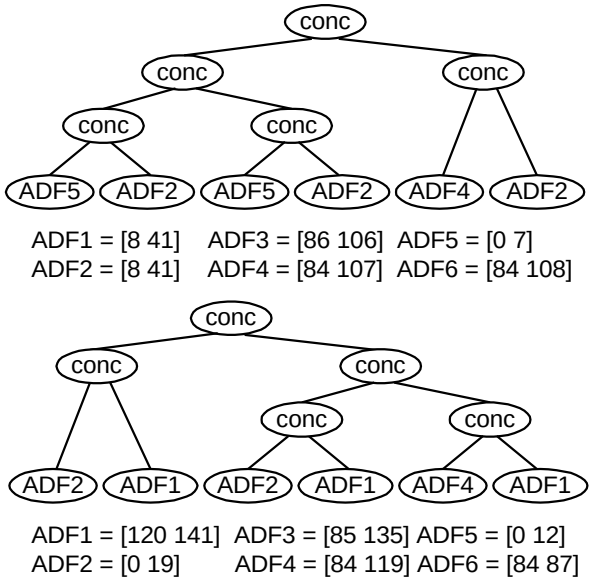

Figure 7: Two further segmentations of «Be m'anperdout…»

Although these experiments resulted in a few interesting individuals, all with the same structure of the ones in Figure 7 (*a* + *b* + *a* + *b* + *c* + *b*), we think that this approach for further segmentations is still suffering from some deficiencies, namely, the fact that it does not consider the structure outlined in the first segmentation (in this case, A + A + X) as already established. This means that the system spends part of the time rediscovering the previously outlined structure. Sketching a suitable procedure for further segmentations will be one of our main tasks in the near future. In the next section we present some ideas that, we hope, will enhance de performance of the system.

6 Conclusions and Future Work

In this paper we have presented an approach to PA, using GP. The way programs are represented, combined with the fitness function, which is better for individuals that produce pieces closer to the original one and that make a better reuse of musical stuff, allows the system to choose the most representative segments. The first results obtained indicate that GP using ADFs may be well suited to the task, due to its capacity to explore the search space regularities. The power that GP has already demonstrated in other very complex areas also leads us

to believe that these results can be substantially improved.

Our future work will comprise several tasks whose main purpose is to enhance the performance of the existent system. These include: outline of a better procedure so that first segmentations can be further detailed; testing of complementary segmentation methods, like the ones based on local discontinuities, so that segmentations and respective evaluation can be more accurate; use multi-population evolution with exchange of genetic material between populations; co-evolution of a GP algorithm which evolves simple tree programs that are executed against several strings of segments evolved by a Genetic Algorithm. Finally, we want to extend the system so that non-monophonic pieces can be analysed, which would also imply a change in SICOM's capabilities.

Acknowledgments

This work was partially supported by POSI - Programa Operacional Sociedade de Informação of Portuguese Fundação para a Ciência e Tecnologia and European Union FEDER, and by Escola Superior de Tecnologia e Gestão de Leiria.

References

C. Anagnostopoulou, G. Westermann. Classification in Music: A Computational Model for Paradigmatic Analysis. In *Proceedings of the International Computer Music Conference*: 125-128, Thessalonik, Greece, 1997.

E. Cambouropoulos, A. Smaill. A Computational Model for the Discovery of Parallel Melodic Passages. Proceedimgs of the XI Coloquio di Informatica Musicale, Bologna, Italy, 1995.

E. Cambouropoulos. A Formal Theory for the Discovery of Local Boundaries in a Melodic Surface. *Proceedings of the III Journées d' Informatique Musicale*, Caen, France, 1996.

E. Cambouropoulos. Musical Parallelism and Melodic Segmentation. In *Proceedings of the XII Colloquium of Musical Informatics*, Gorizia, Italy, 1998.

M. Conte, G. Trautteur, I. de Falco, A. Della Cioppa, E. Tarantino. Genetic Programming Estimates of Kolmogorov Complexity. In *Proceedings of the Seventh International Conference on Genetic Algorithms*: 743-750, Morgan Kaufmann, 1997.

N. Cook. *A Guide to Musical Analysis*. Oxford University Press, 1987.

J. R. Koza. *Genetic Programming*. Cambridge, MA: The MIT Press, 1992.

J. R. Koza. *Genetic Programming II*. Cambridge, MA: The MIT Press, 1994.

J. J. Nattiez. *Fondements d'une Semiologie de la Musique*. Union Generale d'Editions, 1975.

F. C. Pereira, C. Grilo, Luis Macedo, and Amílcar Cardoso. Composing Music with Case Based Reasoning. Second Conference on Computational Models of Creative Cognition. Dublin, Ireland, 1997.

N. Ruwet. *Langage, musique, poésie*. Editions du Seuil, Paris, 1972.

A. Smaill, G. A. Wiggins, and M. Harris. Hierarchical Music Representation to Composition and Analysis. Computer and the Humanities, 27: 7-27, 1993.

G. G. Stephen. String Search. Technical Report, School of Electronic Engineering Science, University College of North Wales, 1992.