Case-based Melody generation with MuzaCazUza

Paulo Ribeiro*, Francisco C. Pereira*, Miguel Ferrand[+], Amílcar Cardoso*
{pribeiro@student.dei.uc.pt, camara@dei.uc.pt, mferrand@music.ed.ac.uk, amilcar@dei.uc.pt}

***Centro de Informática e Sistemas da Universidade de Coimbra (CISUC)**
Polo II, Pinhal de Marrocos
3030 – Coimbra, Portugal
[+] Faculty of Music, University of Edinburgh
Alison House, 12 Nicolson Square EH8 9DF Edinburgh
Scotland

Abstract

MuzaCazUza is a system for interactive composition of musical ideas in two phases: first, the program composes a melody by the application of Case-based Reasoning (CBR) techniques, according to a given harmonic line and a set of pre-defined structure; then, a set of transformation operators are available to change the generated idea.

We believe analogy-based architectures (like CBR) are a fertile ground to model computational creativity and, more specifically, musical composition. Indeed, our everyday reasoning, be it more or less creative, has the constant presence of past experience and it is not uncommon in the Music field to notice recurrence of themes, rhythmic patterns, melodic cells, and other musical objects from previous compositions.

In this paper, we describe MuzaCazUza in some detail and discuss some practical and theoretical issues that will appear within context. We also show some preliminary results.

1. Introduction

This paper presents our most recent work on automatic music composition, more specifically on melody generation with Case-based Reasoning. Following previous work about this subject (omitted1), which focussed on the generation of whole piece structures, now we concentrate on generating melody having, as input, a harmonic line. This system, MuzaCazUza, is built according to the Case-based Reasoning paradigm, well known in the Artificial Intelligence research field. In this program, a case corresponds to a harmonic element (an underlying chord) and its realisation (the melody it harmonizes).

Differently from its predecessor, MuzaCazUza has an interactive environment that allows the insertion of operators (that can be applied to a phrase, a section, or the whole piece) or direct edition of notes. In our opinion, this feature becomes interesting in the sense that it underlies a composition based on transformations of musical structures, instead of focussing on low-level features, i.e., notes. MuzaCaZuZa uses its CBR mechanism to suggest melodies to start with, according to the harmonic line given as input, and has a set of operations available to explore. This environment allows a user with little musical knowledge to apply a set of abstract operators to the current data and design his/her musical concepts. In this sense, we believe this architecture can have an interesting role in the creative process of musical composition.

As in many other similar systems, (such as those referred in (Papadoupoulos and Wiggins, 99)) the question of evaluation is hard to handle. At the moment, we assume the most similar case, according to our metric, must have major priority, although we are conscious this is not at all an unquestionable rule in this field. Other tricky questions arise in MuzaCazUza such as the dependency to the case base, the choice of correct weights and similarity measures for comparison

of musical objects or the little focus on a major overall structural basis to melody generation. We will address these questions in more detail later in sections 3,4 and 5.

This paper starts with a short overview of the Case-based Reasoning paradigm (section 2), after which we will present MuzaCazUza's architecture, describing the aspects we believe to be the most interesting and valuable to the reader (section 3). In section 4, we describe and analyse some results. Finally, in section 5 we will dedicate to conclusions.

2. Case-based Reasoning

Case-based reasoning (CBR) uses past experience to solve new problems (Kolodner, 93) (Watson, 96). In a CBR system, experience is stored in the form of cases in a case library. Cases are episodic chunks of knowledge that are used in the resolution of a new situation. The solving process in CBR is based on the premise that identical problems have identical solutions. In other words, when a CBR system has a new problem to solve, it retrieves cases with similar problem descriptions and applies them to the present situation with the necessary (and possible) modifications. To this phase of changing the retrieved case in order to fit the present requirements, we call adaptation. Finally, the new case created by adaptation is evaluated and may eventually be stored in the case base.

Case representation is a crucial issue in a CBR system. It is the case representation that delimits the CBR steps. For example, cases can only be efficiently retrieved from the case library if important features in the problem description are represented. Cases can be represented by attribute/value pairs, hierarchies of concepts, textual description, causal models and other symbolic representation.

In the case retrieval phase, the CBR systems must decide which cases are the best candidates to solve the target problem. Naturally, there are several ways to accomplish this task, namely using an indexing scheme, where there is a set of indexes associated to each case that are used to retrieve the case; using a metric function like K-nearest neighbour to assess the similarity of the target problem and the case problem's description. Each of these similarity metrics returns an estimation of the similarity between the target and the query cases. According to the system's overall goal, the evaluated cases are ordered.

Case adaptation is known as the most complicated task of CBR, mainly because it is essentially domain dependent. In general terms, the adaptation module of such a system applies some modification operators to shorten the distance between the problem and the solution found. These operators can be production rules, formulas, heuristics, or specific procedures.

In summary, CBR provides a fast reasoning mechanism, especially suited for domains where there is no causal model and where there are no a priori "correct answers". As referred in (omitted2) and as we will describe below, an interesting domain such as Musical Composition can be explored with this paradigm.

3. MuzaCazUza

In the following diagram, we show the general architecture of MuzaCazUza. We will now describe each component briefly.

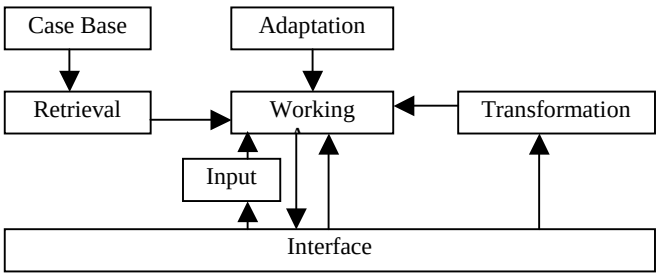

Figure 1 - Architecture of MuzaCazUza

3.1. Input

As a starting point for the melody generation we need to provide MuzaCazUza with a set of degrees and modulations over which the program will accomplish the melody creation. This input is passed to the program in the form of a text file, containing the harmonic structure of the piece, coded with a simple notation the program understands. An example of input is:

[(tM)I,V,vi,IV;I,ii7,V/V;(mV)I,IV,vi,V;(rIV)I,V,V7,I][1]

3.2. Case Base

As with every application that uses Case-Based Reasoning, MuzaCazUza has its power supply in the Case Base (CB). This CB is built upon a simple MS Access database table with twenty fields and has, at the moment, six pieces of music from the Baroque period. As argued before, the case representation is determinant to the behaviour of the system, since it can greatly affect the way cases are retrieved, so for this issue we chose to define each degree (*chord*) and its involving *rhythm* and *melody* to be a case, with a set of atributes. It seems to be the most appropriate representation, since the input is a set of degrees and the output is a melody and rhythm applied to each of those degrees. In other words, when adding a music to the CB, each time a degree changes a new case is stored. Every case stores relevant information about it and its relation with the surrounding degrees. Indispensable fields in a case are: the pitch and duration of the notes that build up the melody, the duration of the degree in measures, the degree itself and also the previous and the next degrees, the amplitude and the direction between the first and the last notes in the case, the beat and division of the measure, and flags like phrase ending and piece ending.

3.3. Retrieval

The retrieval section of the program is perhaps the most important part of MuzaCazUza. Here is the place where the program first comes up with melodic and rhythmic suggestions to the accompaning harmony.

By running through the entire case base, MuzaCazUza scores each and every case in it, selecting one of the best. This score is based on matching rules that are applied by comparing the degrees in the input with the cases in the case base and also with the cases already chosen for the final piece. A case score is itself a weighted sum of the various scores that are assigned to some of its fields. This again is done comparing cases. The program uses a set of formulae to determine how similar these cases are and, as this similarity grows, so does its score. To compare two degrees, we apply a method based on Schöenberg's *chart of the regions* (Schöenberg, 54). Position of measures, melodic and rhtyhmic sequences are compared according to their immediate involving structure (e.g. measure 5 of music A, which has a total of 15 measures is more similar to measure 10 of music B, with a total of 30 measures, than with measure 5, of the same music); other atributes, like *tempo* or *melody direction*, are compared with a all-or-nothing formula, i.e., total similarity if values are equal, none otherwise. We think this approach allows a wider scope of results and a better fine-tuning of what similarities we want to favour.

3.4. Transformation

After every retrieval phase, we end up with a musical piece. However, depending on the input and chosen weights, this piece may have no interesting phrasic structure at all. Furthermore, there are always some improvements we can make over the melody to adjust it to our personal tastes. Whether its use is for improvements, minor adjustements or just for experimentation, MuzaCazUza's operators take the creative process a little further. These operators are divided into two categories: structure and transformation. There are four structure operators: *rep* (repeats both melody and rhythm of a given case or set of cases onto a given target), *repm* (repeats only the melodic pattern), *repr* (repeats only the rhythm pattern) and *transp* (tranposes a case or a set of cases into a target by a given interval). There are also seven transformation operators: *mirrorh* (plays a case or set of cases backward, also known as retrogade), *mirrorvl*, *mirrorvm*, *mirrorvh* (plays an inversion of a case or set of cases centered on the lowest, middle or highest note respectively), linear8, linear16 (links the most relevant notes of a case or set of cases using either eighteenths or sixteenths) and *random* (applies one random operator to a random case or set of cases).

In the transformation module, every note can be edited or deleted.

---

[1] The tM means that the piece is supposed to be in a major mode; mV means modulation to the fifth, while rIV means, in this context, return to the fourth degree, which in relation to the dominant corresponds to the tonic.

3.5. Adaptation

This section does a final passage on the melody, converting every note out of tone to the nearest tonal note. So this adaptation does not focus music theory issues (like the role of the original degree in the case), instead we apply a really pragmatic solution.

3.6. Interface

The interface lets the user interact with the program. This simple frame shows the transformation section, where the user can apply operators to the melody part created by the retrieval module:

3.7. Working Area

This is what MuzaCazUza uses to communicate between the different modules of the program, and where it stores its current state. In the figure 2, we can see the working area where the user can apply the operators, select weights for the case retrieval and configure the midi options.

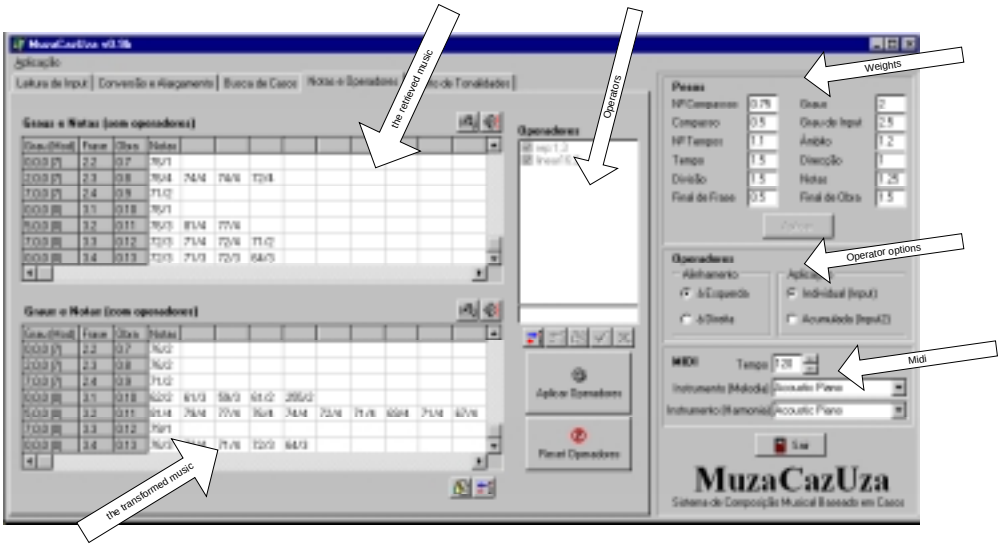

Figure 2 - the interface of the operator, weights and midi working area

4. Results

The results obtained with MuzaCazUza vary greatly but some of them are worth mentioning. For example, 24 variations of a small motif from *Jesu, Joy of Man's Desiring* by *Bach* were done on a four minute piece. Each variation was the application of an operator over the previous one. Although this seems very interesting, scientifically, we cannot take many conclusions so we made a weight impact study test. In this test, we took 11 experiments, each of them with 30 runs. With these experiments, we hoped to evaluate only the weight impact, not the aesthetical side of the results. The table we now show represents the weight configuration for each experiment:

Experiment	*Nmb. Measures*	*Measure*	*Nmb. Beats*	*Beats / Measure*	*Beat Value*	*Degree*	*Input Degree*	*Amplitude*	*Amplitude Direction*	*Notes*	*Phrase Final*	*Piece Final*
1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	5	5
3	1	1	1	1	1	5	1	1	1	1	5	5
4	1	1	1	1	1	5	5	1	1	1	5	5
5	1	1	1	1	1	5	5	1	5	1	5	5
6	1	1	1	1	1	5	5	5	5	1	5	5
7	0	0	0	0	0	1	0	1	0	0	0	0
8	0	0	0	0	0	1	0	1	1	0	0	0
9	0	0	0	0	0	1	1	1	1	0	0	0
10	1	1	1	1	1	0	0	0	0	0	0	0
11	1	1	0	0.5	10	5	3	3	0	3	2	0

Table 1 - Testing MuzaCaZuZa

And the input file contained the following progression:

[(tM)I,IV,V,I]

This is rather a small input, so what we obtained were essentially segment of "wannabe" melodies. It's important to say that the small size of the case base somehow delivers great recurrence to pieces of music. To comment a little bit over the results, we'll start to say that the first experiment was the base for the other experiments, as we attributed the same weight to every item. With the second test being the same but with increased weight for the finals, the resulting measures insistently "froze" in the same idea, because there are few finals in the case base (only six piece finals).

The third and fourth tests started to give the melody line a considerable improvement as the degrees start to enter the weight scene. The degree and input degree weights are very important in the retrieval sequence.

The fifth and sixth tests introduced some notable fluency but led up to unexpected finals. With the seventh and eighth test, we obtained nice melodic pieces but with no rhythm at all, since the rhythm-linked weights were all zero. The ninth test has given better rhythmless melodies with the same unexpected finals.

The tenth test solved the lack of rhythm, but maintained the abrupt finals. Finally, the eleventh test, with some random mixed weights showed up with the best results, with nice and somehow balanced melodic phrases.

As a second test, we ran the same input, but with an ABA structure, where the first A section repeats itself over the second A section. Over the B section, modulated to the dominant, was applied a linear16 operator to add some ornaments to the melody. Every output was also processed by the adaptation module. Therefore, the input was:

Sequence=[I, IV, V, I, V/V; (mV)I, IV, V, I; (rIV)I, IV, V,I] Operators: rep:1,3; linear16:2

The results seemed to show the same properties as the first tests with the "frozen" phrases and uninteresting themes. However the interest increased with the fifth and sixth tests, and with the eighth, the melody started to have some nice notes leading to the tenth test where a glimpse of a genius really awakened. Rapid passages due to the linear16 operator and a nice phrasing, led to the best pieces among all the tests, perhaps because the degree and degree input got a zero weight. Interestingly, the last test wasn't near as creative as the previous one, definitely because the weights aren't well adjusted leading to visible limitations on the results produced.

It seems that much of the weaknesses of MuzaCazUza lie on its small case base. Maybe a wider case base or simply a module for the introduction of new pieces into it could produce nicer, richer and varied melodies. Also, the introduction of generated pieces to the case base would result in some sort of recycling, which would also be of great interest.

Another remarkable observation is that, if adaptation is not used, the dissonant sound sometimes obtained is interesting, even more if we take into account that the cases used contain tonal music from the Baroque era. And it is a fact that not every piece of music created by MuzaCazUza can sound close to Baroque, but with adjustements on the weights and on the input we can obtain some known motifs we immediately associate with this period.

5. Conclusions

There are some visible limitations in the results. Perhaps one of the major problems is the small size of the case base which results in some repeated phrases during retrieval. Although there are some ways to make these phrases unrecognizable, it would be a major improvement to have an easier way to add cases to the case base. Another aspect that deserves some attention is the fact that, although the case base only contains music from the Baroque period, there are some results closer, at the most, to the contemporary period. This is due to the lack of cases in situations where a really suitable case cannot be found, resulting in strong dissonances, and also to the use of different weights in the retrieval process and the use of operators that can truly alter the meaning and sounding of a melody line. These and other factors contribute to an increasingly surprise factor in the final results.

We think this program is a very appealing and funny way to obtain and develop musical ideas, although its interface is still far from easy to use. Nevertheless, the operator philosophy we applied brings an easy way to create.

From a more generic perspective, we are sure MuzaCazUza, and more specifically its underlying architecture, is a promising ground of research in Computational Creativity and Analogy. Our next steps in this path will lean towards automatic generation of cases, since as we could see this is determinant for the results, and study ways of automatic evaluation arround the involved subjects.

References

(Kolodner, 93) Kolodner, J., *Case-Based Reasoning*. Morgan Kaufman Publ., San Mateo, CA. 1993.
(Papadopoulos and Wiggins, 99) Papadopoulos, G. and Wiggins, G. AI Methods for algorithmic composition: A Survey, a Critical View and Future Prospects. In *AISB Symposium on Musical Creativity*, Wiggins, G. (ed.) Edinburgh, UK.1999
(Shöenberg, 54) Schoenberg, A., *Structural Functions Of Harmony*, 1954
(Watson,96) Watson, I. Applying Case-Based Reasoning: Techniques for Enterprise Systems, 1997