SaxEx : a case-based reasoning system for generating expressive musical performances

Josep Lluís Arcos[1], Ramon López de Mántaras[1], and Xavier Serra[2]

[1] *IIIA, Artificial Intelligence Research Institute*
CSIC, Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia, Spain.
{arcos, mantaras}@iiia.csic.es, http://www.iiia.csic.es

[2] *IUA, Audiovisual Institute, Pompeu Fabra University*
La Rambla, 31 08002 Barcelona, Catalonia, Spain.
xserra@iua.upf.es, http://www.iua.upf.es

Abstract

We have studied the problem of generating expressive musical performances in the context of tenor saxophone interpretations. We have done several recordings of a tenor sax playing different Jazz ballads with different degrees of expressiveness including an inexpressive interpretation of each ballad. These recordings are analyzed, using SMS spectral modeling techniques, to extract information related to several expressive parameters. This set of parameters and the scores constitute the set of cases (examples) of a case-based system. From this set of cases, the system infers a set of possible expressive transformations for a given new phrase applying similarity criteria, based on background musical knowledge, between this new phrase and the set of cases. Finally, SaxEx applies the inferred expressive transformations to the new phrase using the synthesis capabilities of SMS.

1 Introduction

The problem with the automatic generation of expressive musical performances is that human performers use musical knowledge that is not explicitely noted in musical scores. Moreover, this knowledge is difficult to verbalize and therefore AI approaches based on declatarive knowledge representations have serious limitations. An alternative approach is that of directly using the knowledge implicit in examples from recordings of human performances.

Previous work has addressed this problem mainly by means of MIDI instruments with the unavoidable limitations regarding expressivity. Our goal is the generation of expressive musical performances in the context of instruments with rich and continuous expressive capabilities (like wind instruments).

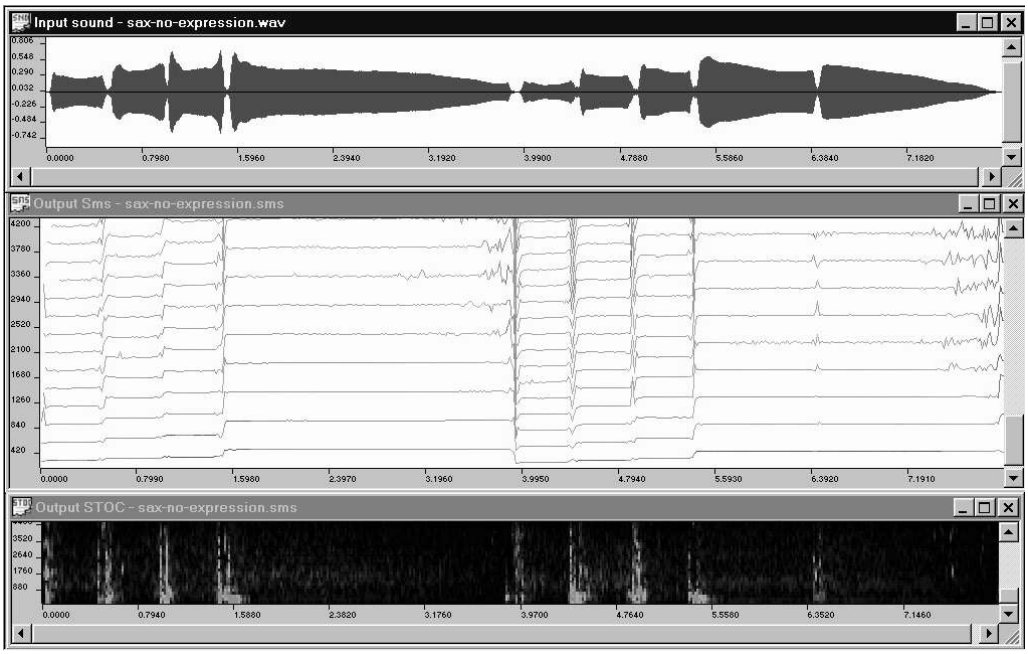


Figure 1: Snapshot of SMS analysis and synthesis graphical interface for the beginning of the 'Autumn Leaves' theme. The top window shows a graphical representation of the input sound file, the middle window shows the evolution of the partials' frequency, and the bottom window shows the spectral residual.

We have developed SaxEx, a case-based reasoning system for generating expressive performances of melodies based on examples of human performances. Case-based Reasoning (Aamodt and Plaza, 1994) (CBR) is a recent approach to problem solving and learning where new problems are solved using similar previously solved problems. The two basic mechanisms used by CBR are (i) the retrieval of solved problems (also called precedents or cases) using some similarity criteria and (ii) the adaptation of the solutions applied in the precedents to the new problem. Case-based reasoning techniques are appropriate on problems where many examples of solved problems can be obtained—like in our case where multiple examples can be easily obtained from recordings of human performances.

Sound analysis and synthesis techniques based on spectrum models like Spectral Modeling Synthesis (SMS) (Serra, 1997) (Serra et al., 1997) are useful for the extraction of high level parameters from real sounds, their transformation and the synthesis of a modified version of the original. SaxEx uses SMS in order to extract basic information related to several expressive parameters such as dynamics, rubato, vibrato, and articulation. The SMS synthesis procedure allows SaxEx the generation of new expressive interpretations (new sound files).

SaxEx incorporates background musical knowledge based on Narmour's implication/realization model (Narmour, 1990) and Lerdahl and Jackendoff's generative theory of tonal music (GTTM) (Lerdahl and Jackendoff, 1993). These theories of musical perception and musical understanding are the basis of the computational model of musical knowledge of the system.

SaxEx is implemented in Noos (Arcos and Plaza, 1997) (Arcos and Plaza, 1996), a reflective object-centered representation language designed to support knowledge modeling of problem solving and learning.

1.1 SMS

SMS is a set of techniques for the analysis, transformation and synthesis of musical sounds. The goal of SMS is to get a general and musically meaningful sound representation, based on spectral analysis, from which we can manipulate musical parameters while maintaining the perceptual identity with the original sound when no transformations are made. Its particular approach to spectral analysis is based on decomposing a sound into sinusoids plus a spectral residual (Serra, 1997).

This process can be controlled by the user, or done automatically depending on the sound characteristics. The analysis procedure detects partials by studying the time-varying spectral characteristics of a sound and represents them with time-varying sinusoids. These partials are then subtracted from the original sound and the remaining residual can be approximated in the frequency domain. Figure 1 shows a snapshot of some of the graphical representations of sounds provided by the SMS graphical interface. Specifically, a window showing a graphical representation of the input sound file, a window showing the evolution of the partials' frequency, and a window showing the spectral residual.

From the sinusoidal plus residual representation we can extract high level attributes when the sound is a note or a monophonic phrase of an instrument. Attributes such as attack and release times, formant structure, vibrato, or average pitch and amplitude, can be obtained by the process described in (Serra et al., 1997). These attributes can be modified and added back to the spectral representation without any loss of sound quality.

This sound analysis and synthesis system is ideal as a preprocessor for Saxex, extracting high level musical parameters, and as a post-processor, adding the transformations specified by the case-based reasoning system to the original sound.

1.2 Case-Based Reasoning

Case-based Reasoning (Kolodner, 1993) (Aamodt and Plaza, 1994) (CBR) is a recent approach to problem solving and learning where a new problem is solved by finding a set of similar previously solved problems, called cases, and reusing them in the new problem situation. The CBR paradigm covers a family of methods that may be described in a common subtask decomposition: the *retrieve* task, the *reuse* task, the *revise* task, and the *retain* task. Different CBR methods differ in the way of achieving these four tasks.

The goal of the *retrieve* task is to recover a set of previously solved problems similar to the current problem. The retrieval task is usually performed using, in turn, three subtasks: *identify*, *search*, and *select* tasks. The *identify* subtask determines, using domain knowledge, the set of relevant aspects of the current problem. Then, using these relevant aspects as similarity criterion, the *search* subtask retrieves a set of precedent cases. Next, the goal of the *select* subtask is to rank the set of precedents using domain knowledge.

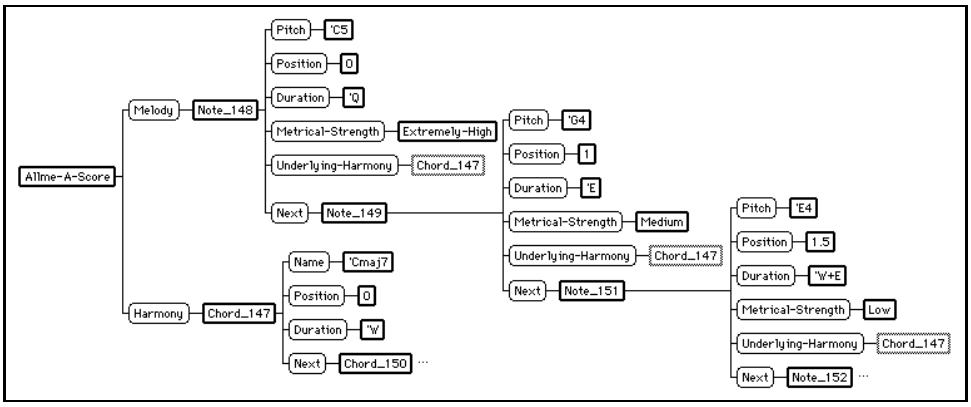


Figure 2: Browse of the score for the 'All of me' ballad represented in Noos. Features are represented as thin boxes, dots indicate not expanded terms, and gray boxes express references to existing terms.

Given a set of ordered precedent cases, the *reuse* task constructs a solution for the current problem adapting the solutions taken in precedent cases. The ranking over cases is interpreted as preference criterion. An usual policy is to consider only the maximal precedent determined by the *select* subtask.

When the solution generated by the *reuse* task is not correct, an opportunity for learning arises. The revision phase involves detecting the errors of the current solution and modifying the solution using repair techniques. This phase, that is not present in all CBR methods, takes the result from applying the solution in the real world (or by asking a teacher).

Finally, the new solved problem is incorporated into the system by the *retain* task in order to help the resolution of future problems. This task involves selecting which information of the case retain and how to integrate the new case in the memory structure.

In Section 2.2 we will see these tasks in the light of the SaxEx system.

1.3 Noos

Noos is a reflective object-centered representation language designed to support knowledge modeling of problem solving and learning. The Noos language has been implemented using Common Lisp and currently is running on several platforms. The main development platform is the Macintosh (using MCL), providing a window-based graphical interface.

Modeling a problem in Noos requires the specification of three different types of knowledge: *domain knowledge*, *problem solving knowledge*, and *metalevel knowledge*.

Domain knowledge specifies a set of *concepts*, a set of *relations* among concepts, and problem data that are relevant for an application. Concepts and relations define the *domain ontology* of an application. For instance, the domain ontology of SaxEx is composed by concepts such as notes, chords, implication/realization structures, and expressive parameters. Problem data, described using the domain ontology, define specific situations (specific problems) that have to be solved. For instance, specific inexpressive musical phrases to

be transformed into expressive ones. Noos is based on *feature terms* (Plaza, 1995). Feature terms are record-like data structures embodying a collection of features. Figure 2 shows the representation of a score in Noos that is described in Section 2.1.

Noos has been used to implement several applications such as CHROMA(Armengol and Plaza, 1994), a system for recommending a plan for the purification of proteins from tissues and cultures, SPIN, a sponge identification system for a class of marine sponge species, and SHAM, a knowledge-based system for harmonizing catalan folk songs.

Problem solving knowledge specifies the set of tasks to be solved in an application. For instance, the main task of SaxEx is to infer a sequence of expressive transformations for a given musical phrase. *Methods* model the ways to solve tasks. Methods can be elementary or can be decomposed into subtasks. These new (sub)tasks may be achieved by other methods. A method defines an execution order of subtasks and an specific combination of the results of the subtasks in order to solve the task it performs. For a given task there may be multiple alternative methods that may be capable of solving the task in different situations. This recursive decomposition of task into subtasks by means of a method is called the task/method decomposition.

Metalevel (or reflective) knowledge is knowledge *about* domain knowledge and problem solving knowledge. Intuitively, metalevel knowledge can be used to model criteria for preferring some methods over other methods for a task in a specific situation.

The metalevel of Noos incorporates *preferences* to model decision making about sets of alternatives present in domain knowledge and problem solving knowledge. For instance, preference knowledge can be used to model criteria for ranking some precedent cases over other precedent cases for a task in a specific situation.

Once a problem is solved, Noos automatically memorizes (stores and indexes) that problem. The collection of problems that a system has solved is called the *Episodic memory* of Noos. The problems solved by Noos are accessible and retrievable. This introspection capability of Noos is the basic building block for integrating learning, and specifically case-based reasoning, into Noos.

Noos also incorporates *perspectives* (Arcos and López de Mántaras, 1997), a mechanism to describe declarative biases for case retrieval in structured and complex representations of cases. Perspectives provide a flexible and dynamical way of retrieval in the episodic memory and are used by SaxEx for making decisions about the relevant aspects of a problem.

2 Saxex

An input for SaxEx is a musical phrase described by means of its musical score (a MIDI file) and a sound. The score contains the melodic and the harmonic information of the musical phrase. The sound contains the recording of an inexpressive interpretation of the musical phrase played by a musician. The output of the system is a new sound file, obtained by transformations of the original sound, containing an expressive performance of the same phrase. Solving a problem in SaxEx involves three phases: the *analysis* phase, the *reasoning* phase, and the *synthesis* phase (see Figure 3).

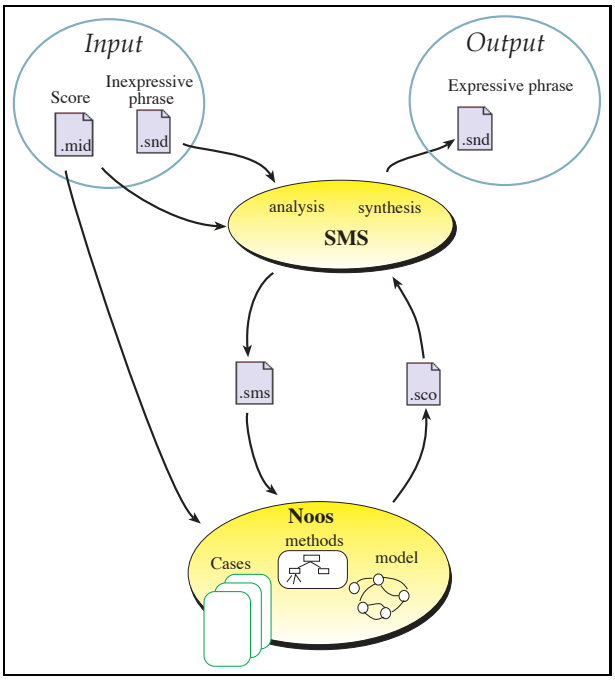


Figure 3: General view of SaxEx blocks.

Analysis and synthesis phases are implemented using SMS sound analysis and synthesis techniques. The reasoning phase is performed using case-based techniques and implemented in Noos and is the main focus of this paper.

SaxEx has been developed specifying two different types of knowledge: (1) modeling the concepts and structures relevant for representing musical knowledge, and (2) developing a problem solving method for inferring a sequence of expressive transformations for a given musical phrase.

2.1 Modeling musical knowledge

Problems solved by SaxEx are represented as complex structured cases (see Figure 4) embodying three different kinds of musical knowledge: (1) concepts related to the score of the phrase such as notes and chords, (2) concepts related to background musical theories such as implication/realization structures and GTTM's time-span reduction nodes, and (3) concepts related to the performance of musical phrases. A Problem to be solved is represented by the score.

A score (see Figure 2) is represented by a `melody`, embodying a sequence of notes, and a `harmony`, embodying a sequence of chords. Each `note` holds in turn a set of features such as the `pitch` of the note (`C5`, `G4`, etc), its `position` with respect to the beginning of the phrase, its `duration`, a reference to its `underlying-harmony`, and a reference to the `next` note of the phrase. Moreover, a note holds the `metrical-strength` feature, inferred using GTTM theory, expressing the note's relative metrical importance into the phrase. Chords hold also a set of features such as the `name` of the chord (`Cmaj7`, `E7`, etc), their

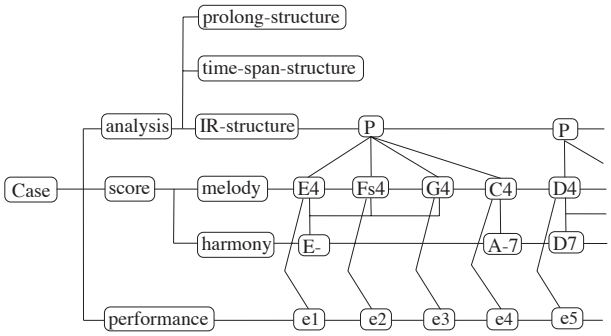


Figure 4: Overall structure of a SaxEx case.

`position`, their `duration`, and a reference to the `next` chord.

The musical analysis structure embodies analysis structures of the phrase built using the background musical knowledge. Narmour's implication/realization model (IR) proposes a theory of cognition of melodies based on eight basic structures. These structures characterize patterns of melodic implications that constitute the basic units of the listener perception. Other parameters such as metric, duration, and rhythmic patterns emphasize or inhibit the perception of these melodic implications. The use of the IR model provides a musical analysis based on the structure of the melodic surface.

On the other hand, Lerdahl and Jackendoff's generative theory of tonal music (GTTM) offers a complementary approach to understanding melodies based on a hierarchical structure of musical cognition. GTTM proposes four types of hierarchical structures associated with a piece. This structural approach provides the system with a complementary view for determining relevant aspects of melodies.

Our goal in using both, IR and GTTM models, is to take advantage of combining the IR analysis of melodic surface with the GTTM structural analysis of the melody. These are two complementary views of melodies that influence the execution of a performance.

Examples of GTTM analysis structures are the `prolongational-reduction` structure embodying a hierarchical structure describing tension-relaxation relationships among groups of notes, the `time-span-reduction` structure embodying a hierarchical structure describing the relative structural importance of notes within the heard rhythmic units of a phrase, and the `process-structure` embodying a sequence of implication/reduction (IR) Narmour's structures.

A performance is represented as a sequence of events. There is an `event` for each note within the phrase embodying knowledge about expressive parameters applied to that note. Specifically, an `event` holds knowledge about expressive parameters of notes such as `dynamics`, `rubato`, `vibrato` level, `articulation`, and `attack`. Expressive parameters are described using qualitative labels as follows:

Changes on dynamics are described relative to the average loudness of the phrase by means of a set of five ordered labels. The middle label represents average loudness and lower and upper labels represent respectively, increasing or decreasing degrees of loudness.

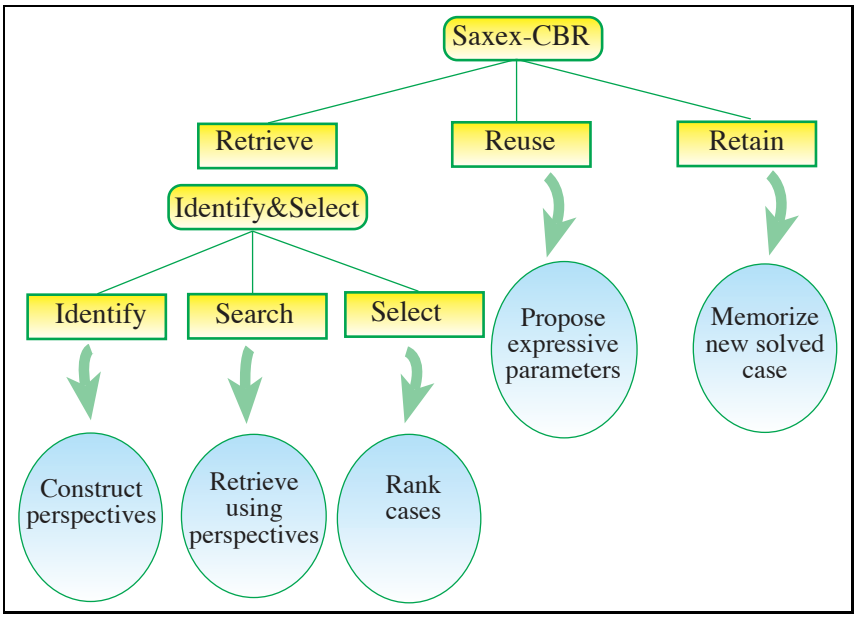


Figure 5: Task decomposition of the SaxEx CBR method.

Changes on rubato are described relative to the average tempo also by means of a set of five ordered labels. Analogously to dynamics, qualitative labels about rubato cover the range from a strong accelerando to a strong ritardando.

The vibrato level is described using two parameters: the frequency vibrato level and the amplitude vibrato level. Both parameters are described using five qualitative labels from `no-vibrato` to `highest-vibrato`.

The articulation between notes is described using again a set of five ordered labels covering the range from `legato` to `staccato`.

Finally, SaxEx distinguishes two transformations over a note attack: (1) reaching the pitch of a note starting from a lower pitch, and (2) increasing the noise component of the sound. These two transformations were chosen because they are characteristic of saxophone playing but other transformations can be introduced without altering the system.

In the conclusions section we discuss different alternatives we are considering for improving the expressive model of the system.

2.2 The SaxEx task

Given a musical phrase, SaxEx infers a specific set of expressive transformations to be applied to every note in the phrase. These sets of transformations are inferred note by note. For each note in the phrase the same problem solving method is performed.

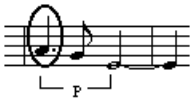
The problem solving method developed in SaxEx follows the usual subtask decomposition of CBR methods described in Section 1.2: *retrieve, reuse, and retain* (see Figure 5). Since the *revise* subtask requires a global vision of the phrase, it is not included as a subtask. Nevertheless, we are studying the possibility of providing an interactive revision mechanism of the proposed SaxEx

solutions.

Given a current note problem of a problem phrase, the overall picture of the subtask decomposition of SaxEx method is the following:

- *Retrieve*: The goal of the retrieve task is to choose the set of notes (cases) most similar to the current note problem. This task is decomposed in three subtasks:
  - *Identify*: The goal of this task is to build retrieval perspectives using two complementary biases: a first bias based or Narmour's implication/realization model, and a second bias based on Lerdahl and Jackendoff's generative theory.
  - *Search*: The goal of this second task is to search cases in the case memory using Noos retrieval methods and previously constructed perspectives.
  - *Select*: The goal of the select task is to rank the retrieved cases using Noos preference methods. The preference methods use criteria such as similarity in duration of notes, harmonic stability, or melodic directions.
- *Reuse*: the goal of the reuse task is to choose a set of expressive transformations to be applied in the current problem from the set of more similar cases. The first criterion used is to adapt the transformations of the most similar case. When several cases are considered equally similar, transformations are selected according to the majority rule. Finally, when previous criteria are not sufficient, all the cases are considered equally possible alternatives and one of them is selected randomly.
- *Retain*: the incorporation of the new solved problem to the memory of cases is performed automatically in Noos. All solved problems will be available for the reasoning process in future problems.

After describing the subtask decomposition of SaxEx problem solving method, we will introduce a simplified example, using musical notation, to help its understanding. Let us suppose that SaxEx has to infer a set of expressive transformations for the encircled note within the following phrase:



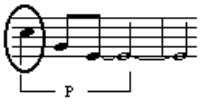
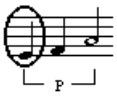
The first subtask engaged is the *retrieve* task. The *retrieve* task engages in turn the *identify* subtask. Taking as example the following bias based on Narmour's model:

Determine as relevant the role that a given note plays in a implication/realization structure.

We obtain the following perspective for our note problem:

that is, the first note of a P process.

Then, the *search* subtask is engaged in order to find similar situations among the precedent cases. Let us assume that the *search* subtask finds the following two notes (called `P1` and `P2`) as precedent cases.

P1:  P2: 

Next, the *select* subtask is engaged for ranking the precedents. Taking as preference criterion the melodic direction, precedent `P1` is considered as the most relevant precedent (since it belongs to a process with descending `direction` like the note problem).

After choosing precedent `P1` as the most relevant precedent, the *reuse* subtask is engaged. For this simplified example, since we have only selected one precedent, the set of expressive transformations to be applied to the current note problem `A` are the same that were applied to precedent `P1` and that are stored as part of precedent case `P1` information.

3 Experiments

We study the issue of musical expression in the context of tenor saxophone interpretations. We have done several recordings of a tenor sax performer playing several Jazz standard ballads ("All of me", "Autumn leaves", "Misty", and "My one and only love") with different degrees of expressiveness, including an (almost) inexpressive interpretation of each piece. These recordings are analyzed, using the SMS spectral modeling techniques, in order to extract basic information related to the expressive parameters. The set of extracted parameters together with the scores of the pieces constitute the set of structured cases of the case-based system. From this set of cases and using similarity criteria based on background musical knowledge, the system infers a set of possible expressive transformations for a given piece. Finally, using the set of inferred transformations and the SMS synthesis procedure, SaxEx generates new expressive interpretations of the same jazz ballads as well as of other similar melodies.

We have performed two sets of experiments combining the different Jazz ballads recordered. The first set of experiments consisted in using examples of three different expressive performances of twenty note phrases of a piece in order to generate new expressive performances of another phrase of the same piece. This group of experiments has revealed that SaxEx identifies clearly the relevant cases even though the new phrase introduces small variations with respect to the phrases existing in the memory of precedent cases.

The second set of experiments consisted in using examples of expressive performances of some pieces in order to generate expressive performances of other pieces. More concretely, we have worked with three different expressive performances of pieces having about fifty notes in order to generate expressive performances of new twenty note phrases. This second group of experiments has revealed that the use of perspectives allows to identify situations such as long notes, ascending or descending melodic lines, etc. Such situations are also usually identified by a human performer.

As an example, let as describe briefly some of the expressive transformations applied by SaxEx to the first phrase of the 'Autumn Leaves' theme (see the

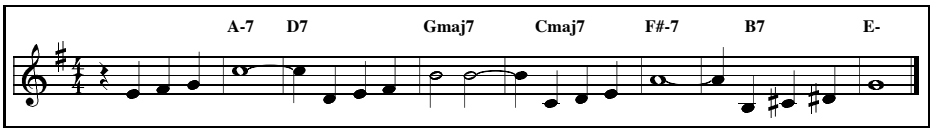


Figure 6: First phrase from the 'Autumn Leaves' theme.

score in Figure 6) based on precedent cases of similar phrases. Concerning to changes of dynamics, the ascending melodic progressions are transformed using crescendo. For instance, the first note (`E`) of the theme starts piano and the dynamics is successively increased yielding a forte in the fourth note (`C`). Concerning rubato, after the fourth note (`C`) the attack of the fifth note (`D`) is delayed and brought closer to the next note, then the duration of sixth note (`E`) is expanded, and finally the duration of the next note (`F`) is reduced. Vibrato is applied over notes with long duration combined with a dynamics decay (for instance, over the fourth note). In ascending melodic progressions, the articulation is also transformed by decreasing the interruption between notes (i.e. playing closer to legato than to staccato). Finally, the transformation of the attack consisted in reaching the eighth and ninth notes (`B` and `B`) starting from a lower pitch.

The reader can visit our web site for sound examples at <http://www.iiia.csic.es/Projects/music/Saxex>.

4 Related work and conclusions

Previous work on the analysis and synthesis of musical expression has addressed the study of at most two parameters such as rubato and vibrato (Clynes, 1995) (Desain and Honing, 1995) (Honing, 1995), or rubato and articulation by means of an expert system (Johnson, 1992). Other work such as in (De Poli et al., 1998) is focalized on the study of how musician's expressive intentions influence the performances.

However, to the best of our knowledge, the only previous work addressing the issue of learning to generate expressive performances based on examples is that of Widmer (Widmer, 1996), who uses explanation-based techniques to learn rules for dynamics and rubato in the context of a MIDI electronic piano. In our approach we deal with additional expressive parameters in the context of an expressively richer instrument.

Furthermore, to the best of our knowledge, this is the first attempt to deal with this problem using case-based techniques as well as the first attempt to cover the full cycle from an input sound file to an output sound file going in the middle through a symbolic reasoning and learning phase.

The results obtained are comparable to a human performance specially for dynamics, rubato and vibrato, however the articulation and attack needs further work.

Concerning future work, we intend to:

- model the degree of the different expressive parameters by means of fuzzy sets, since they are closer than discrete labels to the continuous character of the SMS analysis.

- model the decay of long notes by means of different envelope functions decreasing more or less rapidly.
- experiment further with different expressive parameters and their different degrees of expressiveness.
- With the aim of making our system useful for musicians we intend to provide the possibility of interactive revision of the proposed solutions by the user. In this way the user will have the possibility to filter those solutions that should be retained. This capability will allow the user to tailor the system according to his preferences.

References

Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7(1):39–59. online at <url:http://www.iiia.csic.es/People/enric/AICom_ToC.html>.

Arcos, J. L. and López de Mántaras, R. (1997). Perspectives: a declarative bias mechanism for case retrieval. In Leake, D. and Plaza, E., editors, *ICCBR'97*, Lecture Notes in Artificial Intelligence. Springer-Verlag.

Arcos, J. L. and Plaza, E. (1996). Inference and reflection in the object-centered representation language Noos. *Journal of Future Generation Computer Systems*, 12:173–188.

Arcos, J. L. and Plaza, E. (1997). Noos: an integrated framework for problem solving and learning. In *Knowledge Engineering: Methods and Languages*.

Armengol, E. and Plaza, E. (1994). Integrating induction in a case-based reasoner. In Haton, J. P., Keane, M., and Manago, M., editors, *Advances in Case-Based Reasoning*, number 984 in Lecture Notes in Artificial Intelligence, pages 3–17. Springer-Verlag.

Clynes, M. (1995). Microstructural musical linguistics: composers' pulses are liked most by the best musicians. *Cognition*, 55:269–310.

De Poli, G., Rodà, A., and Vidolin, A. (1998). Note-by-note analysis of the influence of expressive intentions and musical structure in violin performance. *Journal of New Music Research*. This issue.

Desain, P. and Honing, H. (1995). Computational models of beat induction: the rule-based approach. In *IJCAI 95*.

Honing, H. (1995). The vibrato problem, comparing two solutions. *Computer Music Journal*, 19 (3).

Johnson, M. (1992). An expert system for the articulation of Bach fugue melodies. In Baggi, D., editor, *Readings in Computer-Generated Music*, pages 41–51. IEEE Computes Society Press.

Kolodner, J. (1993). *Case-based reasoning*. Morgan Kaufmann.

Lerdahl, F. and Jackendoff, R. (1993). An overview of hierarchical structure in music. In Schwanaver, S. M. and Levitt, D. A., editors, *Machine Models of Music*. The MIT Press. Reproduced from Music Perception.

Narmour, E. (1990). *The Analysis and cognition of basic melodic structures : the implication-realization model*. University of Chicago Press.

Plaza, E. (1995). Cases as terms: A feature term approach to the structured representation of cases. In Veloso, M. and Aamodt, A., editors, *Case-Based Reasoning, ICCBR-95*, number 1010 in Lecture Notes in Artificial Intelligence, pages 265–276. Springer-Verlag.

Serra, X. (1997). Musical sound modelling with sinusoids plus noise. In Poli, G. D., Picialli, A., Pope, S. T., and Roads, C., editors, *Musical Signal Processing*. Swets and Zeitlinger Publishers.

Serra, X., Bonada, J., Herrera, P., and Loureiro, R. (1997). Integrating complementary spectral methods in the design of a musical synthesizer. In *Proceedings of the ICMC'97*.

Widmer, G. (1996). Learning expressive performance: The structure-level approach. *Journal of New Music Research*, 25 (2):179–205.