A Combinatorial Approach to Content-Based Music Selection

François Pachet
Sony Computer Science Laboratory-Paris

Pierre Roy
INRIA

Daniel Cazaly
Sony Music France

Accessing large digital music catalogues raises a problem for both users and content providers. We propose a novel approach to music selection called RecitalComposer, which is based on computing coherent sequences of music titles. This amounts to solving a combinatorial pattern generation problem by using constraint satisfaction techniques.

Music delivery concerns the transportation of music in a digital format to users. Music delivery has recently benefited from technological progress in networking and signal processing. In particular, progress in networking transmission, compression of audio, and protection of digital data[1] allow quick and safe delivery of music through networks, either the Internet or digital-audio broadcasting services. Additionally, digitalization of data makes it possible to transport information on content, and not just data itself, as exemplified by the MPEG-7 project.[2] These techniques give users access to huge catalogues of annotated music.

These technologies address the distribution problem, but also raise the issue of massive amounts of data from which users can choose. In the case of music, a typical database of titles, such as Amazon.com or CDNow, contains about 500,000 titles. A database containing all existing tonal music recordings would probably reach 4 million titles. Including ethnic music and less standard types of music would double or triple this number.

Goals of music selection

The music selection problem is double sided. We define it from both the goals of the music listener and those of the content provider.

The listener's viewpoint

The problem of choosing items is common in Western societies, where an ever-increasing number of available products exist. For entertainment and especially music, however, the choice problem is specific, because the underlying goals—personal enjoyment and excitement—don't fall in the usual categories of rational decision making. Although modeling a listener's goals in accessing music is very complex, we identify two basic ingredients: desire for repetition and desire for surprise.

The desire for repetition is well known in music theory and experimental psychology.[3,4] At the melodic or rhythmic levels of music, repetition breeds contentment. For instance, sequences of repeating notes create listener expectations for the same note to reoccur. At a higher, more complex, level, tonal music is based on structures that create strong expectations of events to come (for example, listeners expect a dominant seventh chord in tonal music to resolve). At the global level of music selection, the desire for repetition leads people to listen to music they already know and like or that is similar to music with which they are familiar.

On the other hand, the desire for surprise is a key to understanding music at all levels of perception. The very theories that emphasize the role of expectation in music also show that listeners don't favor expectations that are always fulfilled—they enjoy surprises and atypical musical progressions.[5] At a broader level, listeners want to occasionally discover new music, new titles, new bands, or new styles.

Of course, these two desires are contradictory. The issue in music selection is to find the right compromise between providing listeners with titles they already know and with titles they don't know, but will probably like.

The content provider's viewpoint

From the record companies' viewpoint, the goal of music delivery is to better exploit the catalogue. Indeed, record companies have problems with the exploitation of their catalogue using standard distribution schemes. For technical reasons, only a small part of the catalogue is actually active, that is, proposed to users, in the form of easily available products. More importantly, the

1070-986X/00/$10.00 © 2000 IEEE

analysis of music sales clearly shows decreases in the sales of albums, and short-term policies based on selling lots of copies of a limited number of items (hits) are no longer efficient. Additionally, the sales of general-purpose samplers (such as a best-of-love-songs collection) are no longer profitable, because consumers already have the hits and don't want to buy CDs containing only one or two titles that they like. Instead of proposing a small number of hits to a large audience, a natural solution is to increase diversity by proposing more customized albums to music consumers.

We examine the approaches to music selection according to these three goals: repetition, surprise, and exploitation of catalogues. We show that current approaches only partially achieve these goals.

Approaches in music selection

We divide current approaches in music selection into two categories: query systems and recommendation systems. In both cases, these approaches provide sets of items to the user from which he or she still must choose.

The database approach

Query systems address database issues for storing and representing musical data. They provide a means of accessing musical items using some sort of semantic information. Users can issue various types of queries, either very specific (such as the title of the Beatles song that contains the word "pepper") or largely underspecified (such as jazz titles). In all cases, the database approach, however sophisticated, only satisfies the goal of repetition, since it provides users with exactly what they ask for, therefore no novelty or surprise is achieved.

Collaborative filtering approaches

Collaborative filtering (CF) systems[6] address the goal of surprise by issuing users personalized recommendations. CF has had some success in the field of music selection (Amazon.com, Firefly, Infoglide, and MyLaunch for example) as well as in other domains such as books and news.

CF is based on the idea of patterns in tastes—that is, tastes are not distributed uniformly. Content providers can exploit these patterns by managing a profile for each user connected to the service. The profile is typically a set of associations of items to grade ratings. In the recommendation phase of CF, the system finds all agents having a similar profile as the user's. It then searches for items preferred by these similar agents but aren't known to this user and recommends these items to him or her.

Experimental results show that the recommendations, at least for simple profiles, are of good quality once the user gives a sufficient amount of initial ratings.[6] However, there are limitations to this approach, evident by studying quantitative simulations of CF systems using work on the dissemination of cultural tastes.[7,8] One limitation is the inclination to cluster formation, which is induced by the very dynamics of the system. CF systems produce interesting recommendations for simple profiles, but get stuck when the profiles get bigger: eclectic profiles are disadvantaged. Another problem, shown experimentally, is that the dynamics favor the creation of hits, that is, items liked by a large fraction of the population. If hits are not intrinsically bad things, they nevertheless limit the possibility of other items to survive in a world dominated by weight sums.

CF addresses the goal of surprise in a safe way by proposing to users items that are similar to items known by them. However, cluster formation and uneven distribution of chances for items (that is, hits) are the main drawbacks of the approach both from the user viewpoint and the content provider viewpoint. Users get clusters from which it's difficult to escape, and content providers don't get systematic exploitation of the catalogue.

On-the-fly music program generation

Instead of proposing sets of individual titles to users, we propose building full-fledged music programs—that is, sequences of music titles—satisfying particular properties.

Overview

There are two motivations for proposing music programs rather than unordered collections of titles. One is that music titles are rarely presented in isolation. For example, CDs, radio programs, and concerts are all made up of temporal sequences of pieces in a certain order. This order is frequently significant because different orders don't produce the same impressions on music listeners. The craft of music programming is to build coherent sequences rather than to just select individual titles.

The other motivation is that properties of sequences play an important role in the perception of music. For instance, several music titles in a similar style convey a particular atmosphere and create expectations for the next titles. As a consequence, a listener may not particularly enjoy an individual title in abstract, but the title may succeed as the right piece at the right time within a sequence.

Rather than focusing on the similarities of individual titles, we can exploit properties of sequences to satisfy the three goals of music selection. We propose building a database of titles, with content information for each title. We specify sequences of music titles (music programs) by defining the properties or patterns we want the program to have. These properties are represented as constraints, in the sense of constraint satisfaction techniques. Finally, a constraint solver computes the solutions of the corresponding combinatorial pattern-generation problem.

Working example

The problem is to build music programs as temporal sequences that satisfy the three goals of music selection: repetition, surprise, and exploitation of catalogues. As an example, we will take a music program for which we specify the desired properties. We focus on the format of the database and the nature of constraints.

Following is a liner-note description of a typical music program. The properties of the sequence are grouped in three categories. This example describes a music program called *Driving a Car*, ideally suited for car music:

1. Listener preferences
   - No slow or very slow tempos
   - At least 30 percent female-type voice
   - At least 30 percent purely instrumental pieces
   - At least 40 percent brass
   - No more than 20 percent country pop style
   - One song by Harry Connick, Jr.
2. Constraints on the coherence of the sequence
   - Styles of titles are close to their neighbors (successor and predecessor) to ensure some style continuity in the sequence
   - Authors are all different
3. Constraints on the exploitation of the catalogue
   - Contains 12 different pieces to fit on a typical CD or Sony MiniDisc format
   - Contains at least five titles from the label Epic/Sony Music, a bias to exploit the catalogue in a particular region

Database of music titles

The database of music titles contains content information needed for specifying the constraints.

Format of the database

Each item is described by attributes taking their value in a predefined taxonomy. The attributes are technical and content. Technical attributes include the title, the author, the duration of the piece, and the recording label (for example, *Learn to Love You*, Harry Connick, Jr., 279 seconds, Epic/Sony Music). Content attributes describe musical properties of individual titles, for example, style (jazz-crooner), type of voice (muffled), music setup (instrumental), type of instruments (brass), tempo (slow-fast), and other optional attributes such as the type of melody (consonant), or the main theme of the lyrics (love).

Currently, in our project the database is created by hand by experts (including coauthor Cazaly). However, current research focuses on extracting some of these attributes (such as tempo[9] or rhythm) automatically, when possible, in the context of the MPEG-7 standardization process.

Taxonomies of values and similarity relations

An important aspect of our project's database is that similarity relations link the values of content attributes. We use these similarity relations for specifying constraints on the continuity of the sequence (the preceding example, *Driving a Car*, contains a constraint on the continuity of styles). We use these taxonomies of attribute values to establish links of partial similarity between items according to a specific dimension of musical content.

Some of these relations are simple ordering relations. For instance tempos take their value in the ordered list (fast, fast-slow, slow-fast, slow). Other attributes such as style take their value in full-fledged taxonomies. The taxonomy of styles is particularly important because it embodies a global musical knowledge that the system exploits.

Internet music retailers and others have designed various classifications of musical styles. These classifications are mainly designed for a query-based approach. For instance, the taxonomy of Amazon.com is a tree-like classification that embodies a relation of generalization and specialization between styles, for example, "blues" is more general than "Memphis blues." As such, this type of classification is well suited for navigating the catalogue to find underspecified items, but it does not represent similarities between styles, for instance, having a common origin, like soul-blues and jazz-crooner.

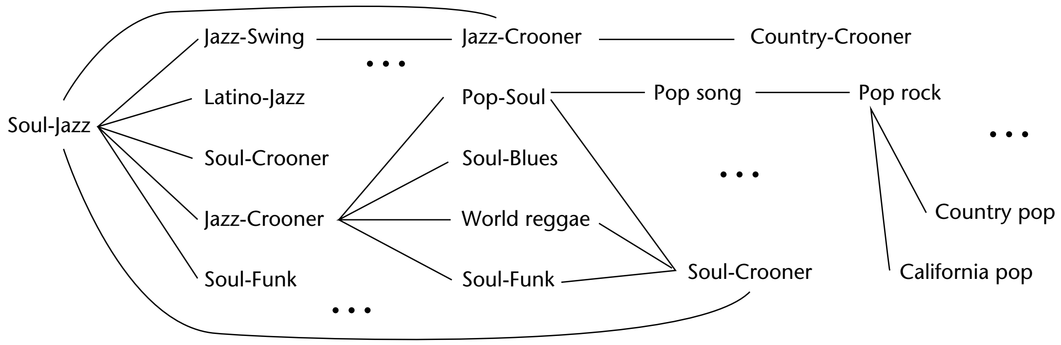

Figure 1. An excerpt of our taxonomy of musical styles. Links indicate a similarity relation between styles, such as between jazz-crooner and soul-blues.

Our taxonomy of styles explicitly represents relations of similarity between styles as a nondirected graph in which vertices are styles and edges express similarity. It currently includes 120 different styles, covering most of Western music (see Figure 1).

CSP for building music programs

Building music programs that satisfy sets of constraints is a combinatorial pattern-generation problem. This problem cannot be solved trivially using a database access technique. Solving the sequence-generation problem using conventional database techniques would require defining a database containing all possible sequences, and this is not possible. For example, the number of sequences of 20 items, out of a catalogue with 100,000 possible values for each item (about the size of a catalogue of a major record label) is 10^{100}.

The task is in fact the opposite of pattern matching because in pattern matching, one looks for patterns in given sequences. Here, we want to create sequences with given patterns.

Constraint satisfaction programming (CSP) is a paradigm for solving difficult combinatorial problems, particularly in the finite domain. In this paradigm, problems are represented by variables having a finite set of possible values, and constraints represent properties that the values of variables should have in solutions. CSP is a powerful paradigm because it lets the user state problems declaratively by describing a priori the properties of its solutions and use general-purpose algorithms to find them.

Most of the CSP resolution algorithms are based on the same principles, where a backtracking procedure is interlaced with a domain reduction phase.

The backtracking procedure. The backtracking procedure progressively instantiates the variables with values taken in their domains and goes backward if an inconsistency is detected that precludes finding any solution. After each step of the search, the algorithm performs a domain reduction phase.

The domain reduction phase. The goal here is to reduce the overall size of the remaining unsolved problem by removing values from the domains of the variables. This leads to developing a smaller search tree and thus speeds up the resolution. Of course, because we don't want to discard any solution, we have to remove values that cannot pertain to any solution. For this, we use constraint-filtering (also called local-consistency or constraint-propagation) techniques.

Filtering a constraint means removing, from the domains of the variables, values that are inconsistent with the current instantiations of the problem, with respect to that constraint. The filtering operation depends heavily on the nature of the constraint considered.[10] The objective of CSP is to identify general-purpose constraints that we can use to specify particular classes of problems (known as global constraints) and design efficient filtering procedures for them. In effect, the efficiency of this general-purpose resolution scheme depends on the efficiency of individual filtering procedures.

CSP for building sequences

We formulated a music program satisfying constraints of a finite domain CSP where the sequence is composed of successive items represented as constrained variables. The domain of the variables is the (finite) catalogue that is searched. Constraints establishing properties of the sequence are expressed in the CSP paradigm and hold on the variables and their attributes. As previously discussed, this formulation yields a hard combinatorial problem. We needed to design efficient filtering procedures to find solutions in a reasonable time.

The community of constraint programming has studied constraints on sequence. For instance, the Sequence Constraint of the CHIP[11] commercial constraint solver enables the expression of complex regulation rules. This constraint is used to control the occurrences of some patterns in a sequence. Specific filtering techniques handle the Sequence Constraint efficiently. This constraint is typically used for complex timetable problems to specify regulation rules (for example, any employee has a two-day rest at least twice a month).

Another kind of sequence constraint is the Global Sequencing Constraint [12] of IlogSolver.[13] This constraint is used to specify the number of successive items having their values in a given set. It's a generalization of the Global Cardinality Constraint[14] and is filtered by the same method.

Our problem is different because we need to constrain not only the value of each item, but also the value of an item's attributes (such as style, tempo, and so on). For instance, we want to have five jazz music titles and three slow motion titles in a row. These requirements cannot be expressed in terms of the Sequence Constraint of CHIP nor of the Global Sequencing Constraint. They are stated by a set of individual cardinality constraints.

RecitalComposer

We can express the constraints needed to specify music programs (listener preferences, program coherence, and exploitation of the catalogue) using a small number of global constraints: similarity constraints, difference constraints, and cardinality constraints. Our resulting system, RecitalComposer, is composed of a constraint solver, a database, and associated taxonomies of attribute values.

Similarity constraints. These state that within a given range, the items are successively similar to each other with respect to the underlying taxonomies. For instance, using similarity constraints, it's possible to state that the first 10 pieces of a program should have similar styles with respect to the classification of styles shown by Figure 1.

Difference constraints. These are used to ensure some variety in the sequence by forbidding identical attribute values for a set of items. For instance, we might require that a radio broadcast doesn't feature songs by the same singer too frequently. This is achieved by stating difference constraints holding on the author attribute and involving any succession of, for example, 10 items in the program.

Technically, this is an extension of the well-known all-different constraint, for which efficient filtering procedures already exist.[15]

Cardinality constraints. These impose numerical restrictions on sets of items. They are the most difficult from a combinatorial point of view, because they prescribe nontrivial properties on the whole sequence. In our context, we identified two such cardinality constraints: cardinality on items and cardinality on attributes.

- *Cardinality on items*. These specify bounds on items for the number of occurrences of certain values in the sequence. For instance, in a classics-of-rock CD sampler, one may require between three and five titles by either the Beatles or the Rolling Stones. More precisely, a cardinality constraint on items is defined by an attribute A, a set Var of variables, a set Val of values, and a number range $[a, b]$. It requires that the number of variables in Var whose value has its attribute A in Val be in $[a, b]$.

- *Cardinality on attribute values*. These control the number of different values taken by a given attribute. This constraint can control the amount of variety in the sequence. A typical use for this constraint would require pieces from at least three different labels in a given program.

Example. We can now express the *Driving a Car* example as a CSP on sequences, by instantiating the global constraints defined above.

- No slow or very slow tempos: simple unary constraints on each variable.

- At least 30 percent female-type voice: cardinality constraint on attribute "voice type."

- At least 30 percent purely instrumental pieces: cardinality constraint on attribute "music setup."

- At least 40 percent brass: cardinality constraint on attribute "instrument."

- No more than 20 percent country pop style: cardinality constraint on attribute "style."

- One song by Harry Connick, Jr.: cardinality constraint on attribute "author."

Table 1. A solution to the music program sequence.

Index number	Title Author Voice type	Style Interpretation type	Duration In seconds Main instrument	Tempo Secondary instrument
3	Sunrise Chet Atkins Instrumental	Jazz California Instrumental	250 Jazz guitar	Slow-fast Strings
21	Surrounded Kreviazuk chant Powerful	California pop Woman	238 Piano	Slow-fast Strings
6	Still is Still Moving to Willie Nelson Nasal	Country California Man	210 California guitar	Fast California guitar
9	Not a Moment too Soon Tim MacGraw Hoarse	Country California Man	222 California guitar	Slow-fast Piano
10	Lovin' All Night Rodney Crowell Normal	Country pop Man	227 California guitar	Fast Brass
11	(The) Hard Way Mary Carpenter Normal	Country pop Woman	262 California guitar	Slow-fast Piano
17	(The) Point of Rescue Hal Ketchum Normal	Country California Man	265 California guitar	Fast California guitar
50	At Seventeen Janis Ian Soft	Pop folk woman	281 Acoustic guitar	Slow-fast Brass
27	Dream On Bill Labounty Broken	California pop Man	298 Keyboard	Slow-fast Brass
106	Another Time Another Place Steely Dan Instrumental	Jazz California Instrumental	245 Piano	Slow-fast Keyboard
112	Learn to Love You Harry Connick, Jr. Muffled	Jazz crooner Man	279 Brass	Slow-fast Strings
137	Heart of My Heart Les Elgart Instrumental	Jazz swing Instrumental	151 Double brass	Slow-fast Brass

- Styles of titles are close to their neighbors (successor and predecessor): similarity constraint on attribute "style."
- Authors are all different: difference constraint on attribute "author."
- Contains 12 different pieces: standard all-different constraint on variables.
- Contains at least five titles from the label Epic/Sony Music: cardinality constraint on attribute "label."

A solution to this problem appears in Table 1. RecitalComposer computes the solution within a few seconds using our global constraints and our constraint solver[10] on a sample catalogue containing 300 titles.

The simplest application of RecitalComposer is a system targeted at music professionals for building music programs from a given database.

Evaluation

We can't compare RecitalComposer to other systems, since we don't know of any other attempts at generating sequences of multimedia data. We can evaluate the scale-up to large catalogues and the quality of results.

Technical evaluation of the CSP approach

We used the current RecitalComposer prototype, written in Java, on a sample database of about 300 titles. For most problems, solutions were computed within a few seconds. Because we don't have a full database with a larger number of items, we experimented on randomly generated databases containing up to 10,000 items. These experiments showed that resolution time grows linearly with the size of the database. However, those results are questionable, since the structure of a randomly generated database may differ from the structure of a real catalogue.

Experiments on databases larger by an order of magnitude are in progress. We claim that such an increase in size is not as critical an issue as it may appear. The complexity of the problem depends more on the density of solutions in the search space than on the size of the search space itself. In our case, the density of solutions augments with the size of the database. Additionally, we may divide the database into smaller domains of interest, thus leading to a moderate increase in the size of the search space.

Evaluation of resulting sequences

The solutions found by RecitalComposer satisfy two goals of music selection. Listener preferences (repetition) are satisfied by definition. Exploitation of the catalogue is systematic, that is, no clustering or bias is introduced and therefore the system searches the entire database for solutions. As illustrated in the working example, specific constraints can be added to force the system to exploit particular regions of the catalogue.

Assessing the surprise goal is more difficult. Unknown titles may be inserted in music programs with a high probability of acceptance by the listener because of the properties of continuity in the sequence. We conducted experiments to compare programs produced by RecitalComposer and programs produced by human experts (including coauthor Cazaly) on the same sample database. Results show that in all cases, experts weren't able to find correct solutions and had to remove one or more constraints. Experts considered the solutions found by the program good, in particular because it yielded items that they wouldn't have considered.

Conclusion

The technique presented here is an enabling technology to build music delivery services. The simplest application of RecitalComposer is a system targeted at music professionals for building music programs from a given database. In the application, the user can specify the constraints and launch the system on a database. The user has full control of all the constraints, so it's aimed at professionals who want to express all the properties of the desired programs.

PathBuilder is an application of RecitalComposer targeting nonprofessionals, in which the user specifies the first and last titles of the program to build. The system contains hidden constraints that guarantee the coherence of the sequence, like continuity of styles and tempos. PathBuilder can, for instance, find a smooth path between Celine Dion's *All by Myself* and Michael Jackson's *Beat It*. In another application the user only specifies the stylistic structure of the program. Users can create a long program for parties or radio broadcasts that has an overall stylistic structure known in advance (such as begin with pop and rock, then slow songs, and so on).

Finally, our approach can produce music programs in various styles by adding domain- specific constraints. We designed and implemented a prototype application dedicated to Baroque music. We used the application to build recitals of Baroque harpsichord music. Recitals of Baroque music (seventeenth century) follow rules identified by musicologists, while allowing a great deal of freedom to performers. A typical rule concerning the structure of recitals is the continuity of tempos between consecutive pieces. More specific rules are also in use, such as rules on the tonality, since during this period of musical history, recitals where allowed to modulate—change tonality from one piece to another—only once. Other constraints concern the structure of the recital (such as the style of the pieces forming the introductory part and requiring that consecutive pieces in the recital are of different type).

We envisage other applications for set-top-box services and digital-audio broadcasting. Our current work focuses on the semiautomatic creation and maintenance of large databases of titles. Indeed, we can extract some of the attributes automatically from input signals and others, such as similarity relations between styles, by using collaborative filtering techniques. **MM**

References

1. N. Memon and P. Wong, "Protecting Digital Media Content," *Comm. of the ACM*, Vol. 41, No. 7, July 1998, pp. 34-43.
2. Report ISO/IEC JTC1/SC29/WG11, "Context and Objectives, International Organization for Standardization," Oct. 1998, http://drogo.cselt.stet.it/mpeg/standards/mpeg-7/mpeg-7.htm.
3. L. Meyer, *Emotions and Meaning in Music*, Univ. of Chicago Press, Chicago, 1956.
4. E. Narmour, *The Analysis and Cognition of Melodic Complexity*, Univ. of Chicago Press, Chicago, 1992.
5. D. Smith and R. Melara, "Aesthetic Preference and Syntactic Prototypicality in Music: 'Tis the Gift to Be Simple," *Cognition*, Vol. 34, 1990, pp. 279-298.
6. U. Shardanand and P. Maes, "Social Information Filtering: Algorithms for Automating 'Word of Mouth'," *Proc. ACM Conf. Human Factors in Computing Systems*, ACM, New York, 1995, pp. 210-217.
7. L. Cavalli-Sforza and M. Feldman, *Cultural Transmission and Evolution: A Quantitative Approach*, Princeton University Press, Princeton, N.J., 1981.
8. J. Epstein, *Growing Artificial Societies: Social Science from the Bottom Up*, MIT Press, Boston, 1996.
9. E. Scheirer, "Tempo and Beat Analysis of Acoustic Musical Signals," *J. of the Acoustical Society of America*, Vol. 103, No. 1, 1998, pp. 588-601.
10. P. Roy and F. Pachet, "Reifying Constraint Satisfaction in Smalltalk," *J. of Object-Oriented Programming (JOOP)*, Vol. 10, No. 4, July/Aug. 1997, pp. 43-51.
11. M. Dincbas, H. Simonis, and P. Van Hentenryck, "Solving Large Combinatorial Problems in Logic Programming" *J. of Logic Programming*, Vol. 7, No. 1, 1990.
12. J.-C. Régin and J.-F. Puget, "A Filtering Algorithm for Global Sequencing Constraints," *Proc. Third Int'l Conf. on Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, Vol. 1330, Springer, Berlin, 1997, pp. 32-46.
13. J.-F. Puget and M. Leconte, "Beyond the Glass Box: Constraints as Objects," *Proc. Int'l Logic Programming Symp.*, MIT Press, Boston, 1995.
14. J.-C. Régin, "Generalized Arc Consistency for Global Cardinality Constraints," *Proc. Int'l Conf. Artificial Intelligence*, Vol. 1, AAAI Press, Cambridge, Mass., 1996.
15. J.-C. Régin, "A Filtering Algorithm for Constraints of Difference in CSPs," *Proc. Int'l Conf. on Artificial Intelligence*, AAAI Press, Cambridge, Mass., 1994, pp. 362-367.

François Pachet, civil engineer, is an assistant professor at the University of Paris 6, where he also received a PhD in computer science and artificial intelligence. He heads the music research team at the Sony Computer Science Laboratory in Paris, where he develops techniques, including constraint-based, for designing and building interactive multimedia systems. He has researched computer music for 10 years.

Pierre Roy is a postdoctorate researcher at the Create computer music research center at UC Santa Barbara, where he designs query systems for music catalogues. He received a PhD from the University of Paris 6. Roy has designed and implemented several constraint frameworks including the BackTalk solver, which was used to implement the RecitalComposer system. He has published several papers on the integration of constraint satisfaction techniques with objects and has presented tutorials on the subject at several international conferences.

Daniel Cazaly has worked as a musical expert for record labels for 30 years, holding positions ranging from disk jockey to working in the marketing department of Sony Music. He is interested in classifications of musical knowledge and the preservation of cultural heritage. He currently works in catalogue design, management, and marketing at Sony Music France.

Readers may contact Pachet at Sony CSL-Paris, 6 Rue Amyot, 75005 Paris, France; pachet@csl.sony.fr.