# Automatic Generation of Music Programs

François Pachet[1] and Pierre Roy[2]

[1]SONY CSL-Paris, 6, rue Amyot, 75005 Paris, France
pachet@csl.sony.fr
[2]INRIA, Domaine de Voluceau, Rocquencourt, France
CREATE, Dept. of Music, UCSB, California, USA
pierre@create.ucsb.edu

**Abstract.** Advances in networking and transmission of digital multimedia data bring huge catalogues of multimedia items to users. In the case of music, accessing these catalogues raises a problem for users and content providers, which we define as the music selection problem. From the user point of view, the goals are to match preferences, as well as provide them with new music. From the content provider viewpoint, the goal is to exploit the catalogue in an optimal fashion. We propose a novel approach to music selection, based on computing coherent sequences of music titles, and show that this amounts to solving a combinatorial pattern generation problem. We propose a language to specify these sequences and a solving technique based on global constraints.

## 1 Music Delivery and Music Selection

Electronic music delivery - transportation of music in a digital format to users - is now becoming a reality. On the one hand, progress in networking transmission, compression of audio, and protection of digital data (Memon and Wong, 1998) will allow in the near future to deliver quickly and safely music to users in a digital format through networks or digital broadcasting. On the other hand, digitalization of data makes it possible to transport not only data itself, but also information on content (see e.g. Mpeg-7, 1998). Together, these techniques give users access at home to huge catalogues of annotated multimedia data, music in particular.

A typical database of music titles contains about 500.000 titles (see, e.g. the MusicBoulevard or Amazon databases). A database containing all published music recordings would probably reach 10 millions titles. Every month, about 4000 CDs are created in western countries. These technological advances raise a new problem for the user as well as for the content provider: how to choose among these catalogues?

### 1.1 The User's Viewpoint

How to choose items is a general problem in western societies, in which there is an ever increasing number of available products. For entertainment and especially music,

© Springer-Verlag Berlin Heidelberg 1999

the choosing problem is specific, because the underlying goals - personal enjoyment and excitement - do not fall in the usual scheme of rational decision making. Although understanding a user's goals in listening to music is complex in full generality, we can summarize the problem to two basic and contradictory ingredients: desire of *repetition*, and desire of *surprise*.

The desire of repetition is well known in music cognition and experimental psychology. At the melodic or rhythmic levels of music "repetition breeds content." Music theorists have tried to capture this phenomenon by proposing theories of musical perception based on expectation mechanisms (Meyer, 1956), for instance for modeling the perception of melodies (Narmour, 92). At the global level of music selection, this desire of repetition tends to have people wanting to listen to music that they know already (and like) or music that is similar to music they already know. For instance, a Beatles fan will probably be interested in listening to the latest Beatles bootleg containing hitherto unreleased versions of his favorite hits.

On the other hand, the desire for surprise is a key to understanding music, at all levels of perception. The theories that emphasize the role of expectation in music also show that listeners do not favor expectations that are always fulfilled, and enjoy surprises and untypical musical progressions (Smith and Melara, 1990). At a larger level, listeners want from time to time to discover new music, new titles, new bands, or new musical styles.

These two desires are contradictory, and the issue in music selection is precisely to find the right compromise between these two desires: provide users with items they already know, and also with items they do not know, but will probably like.

## 1.2 The Content Provider's Viewpoint

From the viewpoint of record companies, one goal of music delivery is to achieve a better exploitation of the catalogue than achieved by standard distribution schemes. The analysis of music sales shows clearly decreases in the sales of albums, and that short-term policies based on selling lots of copies of a limited number of *hits* are no longer profitable. Additionally, the general-purpose "samplers" (e.g. "Best of Love Songs") appear to be no longer appealing, because users have already the hits in their own discotheque, and do not want to buy samplers in which they like only a fraction of the titles. Exploiting more fully the catalogues has become a necessity for record companies. 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 users.

We will now examine the main approaches to music selection according to these three goals: repetition, surprise, and exploitation of catalogues, and show that these approaches only achieve the goals partially. We then propose a novel approach to music selection based on building music *programs*, and discuss its advantages and applications for music delivery services.

## 2 Approaches to Music Selection

Current approaches in music selection can be split up in two categories: database systems and recommendation systems. Both approaches provide sets of items to the user, which he/she has still to choose from.

### 2.1 The Database Approach

Database approaches to music selection address the issues of storing and representing musical data, with an access through explicit queries. Various kinds of queries can be issued by users, either very specific (e.g. the title of the Beatles song which contains the word "pepper"), or largely under specified, e.g. "Jazz" titles. More sophisticated query systems were proposed for music experts, in which *semantic* queries can be made. For instance, in the Humdrum system (Huron, 1994), the user can issue queries such as "all Mozart sonatas which modulate 3 times." In all cases the database approach, however sophisticated, satisfies the goal of repetition, since it provides users with exactly what they ask for. It addresses the goal of surprise in a restricted way only because the properties of the resulting items have to be explicitly specified.

### 2.2 Collaborative Filtering Approaches

Collaborative filtering approaches (Shardanand, and Maes, 1995) aim at achieving the "surprise" goal, i.e. issue recommendations of novel titles to users, with the hope that the user will like them. Recommendation systems based on collaborative filtering techniques have had some success in many domains such as books (Amazon), or personalized news (Resnick et al., 1994). Typical collaborative filtering systems for music are the Firefly system (Firefly, 1998), MyLaunch (MyLaunch, 1998), Amazon (Amazon, 1998), or the similarity engine (Infoglide, 1998).

Collaborative filtering is based on the idea that there are *patterns* in tastes: tastes are not distributed uniformly. This idea can be exploited simply by managing a *profile* for each user connected to the service. The profile is typically a set of associations of items to grades. For instance, in the *MyLaunch* system, grades vary from 0 (*I hate it*) to 5 (*this is my preferred item*). In the recommendation phase, the system looks for all the users having a *similar* profile. This similarity can be computed by a distance measure on profiles, such as a Hamming or Pearson distance. Finally, the system recommends items liked by these similar users.

Experimental results achieved so far on real systems show that such systems produce interesting recommendations for naïve profiles (Shardanand, and Maes, 1995). However, there are limitations to this approach. These limitations appear by quantitative studies of collaborative filtering systems, using simulations techniques inspired from works on the dissemination of cultural tastes (Epstein, 1996; Cavalli-Sforza and Feldman, 1981).

The first one is the inclination to "cluster formation," which is induced by the very dynamics of the system. Recommendation systems get stuck when profiles get bigger (about 120 items): eclectic profiles are somehow disadvantaged. Another problem, shown experimentally, is that the dynamics inherently favors the creation of hits, i.e. items which are liked by a huge fraction of the population. These hits limit the probability of other items to "survive" in a world dominated by weight sums, and hence bias the exploitation of the catalogue.

Collaborative filtering is a means of building similarity relations between items, based on statistical properties of groups of agents. As such, it addresses the goal of surprise in a restricted way, by proposing users items which are similar to already known ones. Cluster formation and uneven distribution of chances to items are the main drawbacks of the approach, both from the user viewpoint (clusters from which it is difficult to escape), and the content provider viewpoint (no systematic exploitation of the catalogue).

## 3 On-the-Fly Music Programs

We propose a different approach to music selection, by shifting the focus of attention: instead of proposing sets of individual titles, we build fully-fledged music programs, i.e. sequences of titles, satisfying particular properties.

There are several motivations for producing music programs, rather than unordered collections of titles. One is based on the recognition that music titles are rarely listened to in isolation: CD, radio programs, concerts are all made up of temporal, ordered sequences of pieces.

The second 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 coming titles. This order is most of the time significant, and the whole craft of music program selection is precisely to build coherent sequences, rather than simply select individual titles. An individual title may not be particularly enjoyed by a listener *per se*, but may be the *right piece at the right time* within a sequence.

### 3.1 General Idea

Our proposal is the following. First, we build a database of titles, with content information for each title. Then we specify the properties of the program as a whole. Our main contribution is a language to specify these properties. Before describing this language, we will give an example of a typical music program, designed with the three goals of music selection in mind.

### 3.2 Working Example

Here is a "liner-note" description of a typical music program, seen as a set of partial properties of the program. These properties are of different nature and may be grouped in three categories: 1) user preferences, 2) global properties on the coherence of sequences, and 3) constraints on the exploitation of the catalogue. The following example describes a music program called “Driving a Car,” ideally suited for listening to music in a car:

**User preferences**
- No slow tempos
- At least 30% female-type voice
- At least 30% purely instrumental pieces
- At least 40% brass
- At most 20% “Country Pop” style
- One song by “Harry Connick Jr.”

**Properties on the coherence of the sequence**
- Styles of titles should be close to their successor and predecessor. This is to ensure some sort of continuity in the sequence, style-wise.
- To ensure variety on the authors, we impose to have at least 10 different authors in the program.

**Properties on the exploitation of the catalogue**
- The total duration is less than 74 minuntes, to fit on a typical CD format.
- The program should contain at least 5 titles from the label “Epic/Sony Music.” This is a bias to exploit the catalogue in a particular region.

## 4 Database of Music Titles

The database contains all the content information required to specify the properties of music programs.

### 4.1 Format of the Database

Each item is described by a set of attributes, which take their value in predefined taxonomies. The attributes are of two sorts: technical attributes and content attributes.

Technical attributes include the *title* (e.g. “Learn to love you”), the *author* (e.g. “Connick Harry Jr.”), the *duration* (e.g. “279 s”), and the *label* (e.g. “Epic/Sony Music”).

Content attributes describe musical properties of individual titles. The attributes are the following: *style* (e.g. “Jazz Crooner”), *type of voice* (e.g. “muffled”), *music setup* (e.g. “instrumental”), *type of instruments* (e.g. “brass”), *tempo* (e.g. “slow-fast”), and other optional attributes such as the *type of melody* (e.g. “consonant”), or the main *theme* of the lyrics (e.g. “love”).

## 4.2 Taxonomies

An important aspect of the database is that the values of content attributes are linked to each other by *similarity* relations. These relations are used for specifying properties of continuity in the sequence, and therefore establish links of partial similarity between items, according to a specific dimension of musical content. For instance, the working example specifies continuity on styles.

Some of these relations are simple ordering relations, e.g. tempos and durations. Other attributes such as *style*, take their value in full-fledged taxonomies.

The taxonomy of styles is particularly worth mentioning, because it embodies expert knowledge on music catalogues. Music retailers, such as Amazon (1998) or MusicBoulevard (1998), have designed several taxonomies of musical styles. These classifications are usually hierarchies oriented toward query-based search. Figure 1 shows such a classification with a relation of "generalization-specialization" between styles: "Blues" is more general than "Chicago-Blues." The classification is well suited for finding under-specified items. However, it does not represent similarities between styles, for instance styles having common origins, like, say, "Soul-Blues" and "Jazz-Crooner".

- All Styles
  - Blues
    - Acoustic Blues Chicago Blues
    - …
  - Country …
  - Jazz …

**Fig. 1.** Excerpts of the taxonomy of styles of the Amazon web site (a tree of depth three). Here, a focus on the "Blues" node

Conversely, we designed a taxonomy representing explicitly relations of similarity between styles. Our taxonomy is a non-directed graph in which vertices are styles and edges express similarity. It currently includes 120 styles, covering most of western music (see Figure 2).

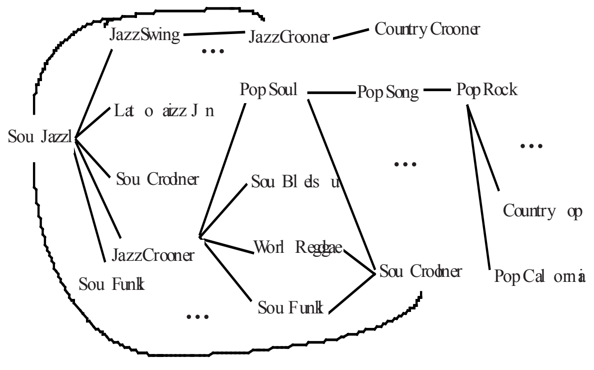

**Fig. 2.** A part of our taxonomy of musical styles. Links indicate a similarity relation between styles. For instance, the relation *similar ("Soul-Blues", "Jazz-Crooner")* holds

## 5 Constraints for Music Sequences

The properties of music programs that are sought hold on sequences as a whole, and not on individual items. In this respect, building music programs may be formulated as a combinatorial pattern generation problem.

Constraint satisfaction programming (CSP) is a paradigm for solving combinatorial problems, particularly in the finite domain (Mackworth, 1977). In CSP, problems are represented by variables, having a finite set of values, and constraints, which represent properties of the solutions. The most widely used CSP algorithms are based on the notion of constraint filtering: each constraint is taken individually to reduce the search space; and filtering depends heavily on the constraint (Roy et al., 1999). An important trend in CSP is to propose *global constraints* (Beldiceanu and Contejean, 1994), i.e. general-purpose constraints that can be used to specify particular classes of problems, with efficient filtering procedures.

In the following section we propose a small set of global constraints that can be used to specify most of music programs, together with efficient filtering procedures. The resulting system, *RecitalComposer* consists of a solver, a database, and taxonomies of attribute values.

### 5.1 Sequence Constraints

A music program can be seen as a solution of a constraint satisfaction problem, where the sequence is composed of successive items represented as variables $v_1, v_2, \ldots v_n$. Each $v_i$ represents the $i^{th}$ item in the sequence. The domain of each $v_i$ is the - finite - catalogue to look from. Properties of the sequence can be expressed as constraints holding on the variables $v_i$, and their attributes $v_i^j$ (see 4.1). This formulation yields a hard combinatorial problem. For instance, finding a sequence of 20 items, with 100,000 values for each item (about the size of a catalogue of a major label) represents a search space of $10^{100}$. Therefore, efficient filtering procedures have to be designed to find solutions in a reasonable time.

Constraints on sequences have been studied in the community of constraint satisfaction. The *Sequence Constraint* of CHIP (Beldiceanu and Contejean, 1994) enables the expression of complex regulation rules. This constraint is used to control the occurrences of patterns in a sequence. This constraint is typically used for timetable problems to specify regulation rules (e.g. any employee has at least a two-day rest twice a month (Chan et al., 1998)). The *Global Sequencing Constraint* (Régin and Puget, 1997) of IlogSolver (Puget and Leconte, 1995) is used to specify the number of successive items having their values in a given set. This constraint is a generalization of the global cardinality constraint (Régin, 1996) and is filtered by the same method. This constraint was successfully applied, for instance, to schedule the production of cars on an assembly line.

Our problem is different because we need to constrain not only the value of each item, but also the value of item's attributes (e.g. *style*, *tempo*, etc). For instance, we want to have five Jazz titles and 3 slow titles in a raw. These requirements cannot be expressed neither in terms of the Sequence Constraint of CHIP nor of the Global Sequencing Constraint. We state these requirements by a set of global constraints whose description follows.

## 5.2 Global Constraints

Most of the properties needed to specify music programs (user preferences, constraints on program coherence, and constraints on the exploitation of the catalogue) can be expressed using the following global constraints: similarity, difference, and cardinality.

### 5.2.1 Similarity Constraints

This constraint states that within a given range, items are successively "similar" to each other. The similarity is defined by a binary predicate, holding on one given attribute $j$. The general formulation is:

$$S(a, b, j, similar(,))$$

meaning that:

For every item $v_i$, $i \in [a, b - 1]$, $similar(v_i^j, v_{i+1}{}^j)$.

where $a$ and $b$ are integers representing indices, $j$ is an attribute, and *similar(,)* is a binary predicate. Each variable of the predicate denotes an item's $j^{th}$ attribute. For instance, this constraint allows to state that the 10 first pieces should have "close" styles, in the sense of the similarity relation of the classification of styles.

This constraint is decomposed into a series of binary constraints holding on each pair of successive items $v_i$ and $v_{i+1}$, so no specific filtering procedure is needed.

### 5.2.2 Difference Constraints

This constraint enforces differences of attributes on a set of items. Its general formulation is:

$$D(I, j)$$

meaning that: all items $v_i$, $i \in I$, have pairwise different values for attribute $j$.

Here, $I$ is a set of item indices, $j$ is an attribute. This constraint allows to state that, e.g. the 10 first pieces should have different authors, or different styles. This constraint is an extension of the *all-different* constraint, for which an efficient filtering procedure was proposed in (Régin, 1994).

### 5.2.3 Cardinality Constraints (CC)

Cardinality constraints (CC in short) impose properties on *sets of items*. They are the most difficult from a combinatorial point of view because they state properties on the whole sequence. In our context, we identified two such CCs: cardinality on items and cardinality on attributes.

*Cardinality on Items*
This constraint states that the number of items whose attribute $j$ belongs to a given set E is within [a, b]. The general formulation is:

$$CI(I, j, a, b, E)$$

meaning that:

$$Card\ \{i \in I;\ v_i^j \in E\} \in [a, b]$$

where I is a set of item indices, $j$ is an attribute, $a$ and $b$ are integers and E is a subset of the possible values for attribute $j$. For instance, this constraint can be used to state that there should be between 4 and 6 pieces within a given range (e.g. the first 10), whose style is "Rock."

*Cardinality on Attribute Values*
This constraint states that the number of different values for attribute $j$ of a number of items is within [a, b]. The general formulation is:

$$CA(I, j, a, b)$$

meaning that:

$$Card\ \{v_i^j;\ i \in I\} \in [a, b]$$

where I is a set of item indices, $j$ is an attribute index, $a$ and $b$ are integers. This constraint can be used for instance to state that among a sequence, there should be pieces from at least three different labels.

*Filtering CCs*
The filtering procedures for CCs are more sophisticated than for previous constraints. For reasons of space limitation, we describe only the filtering procedures for CCs on items.

One can implement CCs in two different ways: as single global constraints, with a specific filtering method, or as a logically equivalent set of elementary standard constraints. The first approach is used for the Cardinality constraint implemented in Ilog Solver (Puget and Leconte, 1995), while the second one is used in the clp(FD) system (Codognet and Diaz, 1996). Both implementations are efficient, as shown by benchmarks using these systems (Fernandez and Hill, 1998). We have followed the second approach to implement our CCs. The detailed implementation follows: we define CC on items by a set of Boolean variables linked by elementary constraints

- $CI(I, j, a, b, E)$
- $\forall i \in I$, let $B_i$ be a Boolean variable (0/1)
- $\forall i \in I$, state constraint $B_i = 1 \text{ iff } v_i^j \in E$
- State linear constraints: $a \leq \Sigma_{i \in I} B_i \leq b$

A CC holding on $n$ variables is defined by $n$ additional Boolean variables, $n$ constraints of type $x \in$ E and two linear inequalities. In this implementation, there is no specific filtering procedure for the CC constraints, because the $x \in$ E constraints and the linear inequality constraints are efficiently filtered during the resolution.

When several CCs are stated on a common sequence of items, the problem can become very hard. For example, consider a 12-title sequence and the 2 following CCs:

(C1) At least 7 titles with author = "Sinatra"
(C2) At least 7 titles with style = "Hard Rock"

The problem has obviously no solution because Sinatra has never recorded any Hard-Rock title. However, this implicit relation between Sinatra and Hard Rock is not represented explicitly. Therefore, the resolution is extremely hard for any consistency-based CSP algorithms. This is because constraints are considered (filtered) individually during resolution of the problem. Note that this is independent of the underlying implementation of the CCs.

To address this issue, we propose to add redundant CCs, which will represent a link between two different attributes (here, author and style). To do so, we use formal reasoning. The method has two steps:

1) We introduce a CC (C1') that holds on the type attribute and that is deduced from (C1). In our example, (C1') would be the following constraint:

(C1') At least 7 songs have their voice style attribute in {Pop Song, Love Song, Crooner Song}.

2) We combine (C1') with (C2). This results in detecting that we need 7 hard rock titles and 7 non-Hard Rock titles, which leads to an inconsistency right away, without backtracking.

Note that this problem cannot be handled directly using the Global Cardinality Constraint (Régin, 1996). In effect, the initial constraints do not involve the same variables, and after rewriting one of the constraints (C1 in the example above), the two constraints involve the same attribute variables, but not the same set of indices.

Therefore, we use a specific strategy to combine the CCs. The general formula for two CCs on the same attribute $i$ is the following:

Let $CI(I, i, a, b, E)$ and $CI(J, i, a', b', F)$ be two CC on items, holding on the same attribute $i$.
Let $n := card(E \cap F)$
We state the two following redundant CCs:

$$CI(I \cup J, i, a + a' - n, b + b', E \cup F)$$
$$CI(I \cap J, i, a + a' - n - |I| - |J| + 2.|I \cap J|, b + b', E \cup F)$$

where $|I|$ denotes the number of elements of set $I$. The constraints are stated only if they are not trivially satisfied, i.e. if $a + a' - n > a$ and $a + a' - n > a'$

In practice, redundant constraints improve drastically the resolution of problems with several CCs. **Table 1** shows the number of backtracks needed to find solutions in several variations around the working example.

**Table 1.** Number of backtracks needed to compute solutions of four problems, with and without redundant constraints. #1 is the working example, #2 is #1 with a general difference constraint on authors, #3 is #2 with at least 20% Country-Pop songs, #4 is #3 with one song by Harry Connick Jr. Since the resolution time here is proportional to the number of backtracks, this shows the interest of redundant constraints to speed up the resolution

| | No Redundant Constraint | With Redundant Constraints |
|---|---|---|
| #1 | 393 backtracks | 358 backtracks |
| #2 | 898 backtracks | 898 backtracks |
| #3 | 364 backtracks | 73 backtracks |
| #4 | 2,531 backtracks | acktracks |

## 5.3 Example

We can now express the example given in Section 3.2 as a constraint satisfaction problem on sequences, by instantiating the global constraints defined above.

- No slow tempos: unary constraints on each variable.
- At least 30% female voice: CC on “voice-type”
- At least 30% instrumental pieces: CC on “music setup”
- At least 40% brass: CC on “instrument”
- At most 20% “Country Pop” style: CC on “style”
- One song by “Harry Connick Jr”: CC on “author”
- Styles of titles are close to their successor and predecessor: similarity constraint on attribute “style”
- At least 10 authors are different: cardinality on item constraint with attribute “author”
- Different pieces: standard all-different constraint
- At least 5 titles from label “Epic/Sony Music”: CC on “label”

Figure 3 shows a solution of this problem, computed within a few seconds by our Java constraint solver (Roy et al., 1999), extended with sequence constraints, and applied to a 200-title sample catalogue.

```
1. Sunrise (Atkins Chet, Jazz Calif, 250s, slow fast, instr,
   instr, jazz guitar, strings)
2. Surrounded (Kreviazuk Chant, Pop-Calif, 238s, slow fast
   powerful Woman piano strings
3. Still is still moving to (Nelson Willie, Country Calif, 210s,
   fast, nasal, Man, calif guitar, calif guitar)
4. Not a moment too soon (Mac Graw Tim, Country Calif, 222s, slow
   fast, hoarse, Man, calif guitar, piano)
5. Lovin' all night (Crowell Rodney, Country Pop, 227s, fast,
   normal, Man, calif guitar, brass)
6. Hard way (the) (Carpenter Mary, Country Pop, 262s, slow fast,
   normal, Woman, calif guitar, piano)
7. Point of rescue (the) (Ketchum Hal, Country Calif, 265s, fast,
   normal, Man, calif guitar, calif guitar)
8. At seventeen (Ian Janis, Pop Folk, 281s, slow fast, soft,
   Woman, acoustic guitar, brass)
9. Dream on (Labounty Bill, Pop Calif, 298s, slow fast, broken,
   Man, keyboard, brass)
10.Another time another place (Steely Dan, Jazz Calif, 245s, fast
   slow, instrumental, Instrumental, piano, keyboard)
11.Learn to love you (Connick Harry Jr, Jazz Crooner, 279s, slow
   fast, muffled, Man, brass, strings)
12.Heart of my heart (Elgart Les, Jazz Swing, 151s, slow fast,
   instrumental, Instrumental, double bass, brass)
```

**Fig. 3.** A solution of the program defined in Section 5.3

## 6 Evaluation

The comparison of *RecitalComposer* with other systems is difficult, since we do not know any other attempt at generating sequences of music titles. We give here indications about the scale-up to large catalogues, and the quality of results.

### 6.1 The Constraint Approach

The current prototype was used on a sample database of 200 titles. Solutions are computed within a few seconds. We did experiments on a dummy database of 10,000 items consisting of the initial database duplicated 50 times. These experiments show that resolution times grow linearly with the database size.

Experiments on databases larger by an order of magnitude are in progress and not reported here. We claim that such an increase in size do not pose any problem for at least two reasons. First, the database may be split up in smaller domains of interest for the solver, using simple heuristics. This permits to use only a small part of the actual database during the search. Second, the increase of the number of items is not related to the number of backtracks. More precisely, the only relevant parameter is the *density of solutions* in the search space, which, in our case, increases directly with the size of the catalogue, thus leading to easier problems.

### 6.2 Resulting Sequences

The solutions found by *RecitalComposer* satisfy trivially two goals of music selection: user preferences (repetition) are satisfied by definition, and exploitation of the catalogue is as systematic as can be; no clustering or bias is introduced, so the system searches the entire database for solutions. Moreover, 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 of course more difficult. The basic idea is that unknown titles may be inserted in music programs with a high probability of being accepted, because of the underlying properties of continuity in the sequence. Experiments are currently conducted, which consist in comparing programs produced by *RecitalComposer*, and programs produced by human experts (Sony Music) on the same sample database. Preliminary results show that the solutions found by the program are good, and yield unexpected items that experts would not have thought about.

## 7 Music Delivery Services

The simplest application of *RecitalComposer* is a system targeted at music professionals for building music program from a given database. In this system, the user has to express explicitly all the properties of the desired programs.

Other applications are dedicated to average users and allow them to express only their preferences, using automatic profiling systems, and contain predefined, fixed constraints sets for the coherence properties and catalogue exploitation, according to predetermined "ambiences" or configurations. Typical configurations are 1) "Progressive programs", in which the user only specifies the stylistic structure of the program (e.g. the styles of the beginning, middle and end), 2) "Path across different titles", in which the user specifies only a starting title and an ending title. The system contains hidden constraints on continuity of styles, and tempos, and builds a "morphing" path between the two titles 3) Applications for particular music domains, like Baroque Music, for which specific stylistic constraints are already predefined. Other applications for set-top-boxes and digital broadcasting are not detailed here for reasons of space.

## 8 Conclusion

*RecitalComposer* is a constraint system for building music delivery services. The system is based on the idea of creating explicit sequences of items, specified by their global properties, rather than computing sets of items satisfying explicit queries. Its main advantage over database or collaborative filtering approaches is that it produces ready to use ordered sequences of items, which satisfy the three goals of music selection, i.e. repetition, surprise, and exploitation of catalogues.

In the current state of our project, music experts create the database and related taxonomies by hand. Current work focuses on the semi-automatic creation and maintenance of large databases of titles. Indeed, some of the attributes can be extracted automatically from input signals (e.g. the tempo, see (Scheirer, 1998)). Finally, relations such as similarity relations between styles could be extracted using collaborative filtering techniques.

## References

1. Amazon Music web site, www.amazon.com, 1998
2. Beldiceanu, N. Contejean, E. Introducing Global Constraints in CHIP, *Journal of Mathematical and Computer Modeling*, Vol. 20 (12), pp. 97-123, 1994
3. Cavalli-Sforza, L. and Feldman, M. Cultural Transmission and Evolution: a Quantitative Approach, Princeton University Press, 1981
4. Chan, P. Heus, K. Weil, G. Nurse scheduling with global constraints in CHIP: GYMNASTE, 4th International Conference on the Practical Application of Constraint Technology, London (UK), pp. 157-169, 1998
5. Codognet, Ph. And Diaz, D. Compiling Constraints in clp(FD), *The Journal of Logic Programming*, 27, pp. 1-199, 1996
6. Epstein, J. M. Growing Artificial Societies: Social Science from the Bottom Up, MIT Press, 1996
7. Fernandez, A. and Hill, P. A Comparative Study of Eight Constraint Programming Languages over the Boolean and Finite Domains, University of Leeds, School of Computer Studies, Technical Report #98.19.
8. Firefly web site, http://www.firefly.com, 1998
9. Huron, D. The Humdrum Toolkit Reference Manual, Center for Computer Assisted Research in the Humanities, Menlo Park, 1994
10. Infoglide web site, www.infoglide.com, 1998
11. Mackworth, A. Consistency on networks of relations, *Artificial Intelligence*, (8) pp. 99-118, 1977
12. Memon, N, Wong, P. W. Protecting Digital Media Content, CACM, July 1998, pp. 34-43, 1998
13. Meyer, L. Emotions and meaning in Music, University of Chicago Press, 1956
14. Mpeg-7, Context and objectives, International Organization for Standardization, report ISO/IEC JTC1/SC29/WG11, October 1998
15. MusicBoulevard web site, www.musicblvd.com, 1998
16. MyLaunch web site: www.mylaunch.com, 1998
17. Narmour, E. The analysis and cognition of meldic complexity, University of Chicago Press, 1992
18. Puget, J.-F. and Leconte, M. Beyond the Glass Box: Constraints as Objects, ILPS'95, Portland, Oregon, 1995
19. Régin, J.-C. A Filtering Algorithm for Constraints of Difference in CSPs AAAI'94, Seattle, pp. 362-367, 1994

20. Régin, J.-C. and Puget J-F. A filtering algorithm for global sequencing constraints, 3rd Int. Conference on Principles and Practice of Constraint Programming, pp 32-46, 1997
21. Régin, J.-C. Generalized Arc Consistency for Global Cardinality Constraints AAAI' 96, Seattle, WA, 1996
22. Resnick, P. Iacovou, N. Sushak, M. Bergstrom, P. Riedl, J. GroupLens: An Open Architecture for Collaborative Filtering of Netnews, CSCW conference, October 1994
23. Roy, P. Liret, A. Pachet, F. A Framework for Constraint Satisfaction, Object-Oriented Application Frameworks, Vol. 2, Fayad, M. Eds, Wiley, 1999
24. Scheirer, E. D. Tempo and beat analysis of acoustic musical signals, *Journal of the Acoustical Society of America* 103(1): 588-601, 1998
25. Shardanand, U. and Maes, P. Social Information Filtering: Algorithms for Automating "Word of Mouth", Proceedings of the 1995 ACM Conference on Human Factors in Computing Systems, pp. 210-217, 1995
26. Smith, D. Melara, R. Aesthetic preference and syntactic prototypicality in music: 'Tis the gift to be simple, Cognition, 34, pp. 279-298, 1990