A Combinatorial Approach to Content-based Music Selection

François Pachet
Sony CSL-Paris,
6, rue Amyot,
75005 Paris, France
E-mail: pachet@csl.sony.fr

Pierre Roy
LIP6
University of Paris 6,
Paris, France
E-mail: roy@poleia.lip6.fr

Daniel Cazaly
Sony Music France
Paris, France

Abstract

Advances in networking and transmission of digital multimedia data will bring soon huge catalogues of music to users. Accessing these catalogues raises a problem for users and content providers, that we define as the music selection problem. We introduce three main goals to be satisfied in music selection: match user preferences, provide users with new music, and 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 constraint satisfaction techniques to solve it. The resulting system is an enabling technology to build better music delivery services.

1. Music Delivery and Selection

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 [7] allow now or in the near future to deliver quickly and safely music to users in a digital format through networks, either internet, or digital audio broadcasting. Additionally, digitalization of data makes it possible today to transport information on content, and not only data itself, as exemplified by the Mpeg-7 project [9]. All these techniques give users, at home, access to huge catalogues of annotated music.

These techniques address the *distribution* problem, but also raise the problem of choosing among these catalogues. In the case of music, a typical database of titles contains about 500.000 titles ([1, 10]). A database containing all tonal music recordings would probably reach 4 millions titles. Ethnic music and less "standard" types of music would probably double or triple this number. Every month, about 4000 CDs are created in western countries.

2. Goals of Music Selection

We define in this section the music selection problem according to the goals of the user and the content provider.

2.1 The user's viewpoint

The problem of choosing items is general in western societies, in which there is an ever increasing number of products available. For entertainment and specially music, however, the choosing problem is specific, because the underlying goals - personal enjoyment and excitement - do not fall in the usual categories of rational decision making. Although modeling a user's goals in accessing music is very complex, we identify two basic ingredients: desire of repetition, and desire of surprise.

The desire of *repetition* is well known in music theory and experimental psychology [8, 12]. At the melodic or rhythmic levels of music "repetition breeds content". For instance, sequences of repeating notes create expectations of the same note to occur. At a higher level, tonal music is based on structures that create strong expectations on the next events to come (e.g. dominant seventh chord in tonal music are expected to resolve). At the global level of music selection, this desire of repetition tends to have people wanting to listen to music they know already (and like) or that is similar to music they already know: 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 very 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 [20]. At a larger level, listeners want from time to time to discover new music, new titles, new bands, or new musical styles.

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

2.2 The content's provider viewpoint

From the viewpoint of record companies, the goal of music delivery is to achieve a better exploitation of 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", i.e. proposed to users, in the form of easily available products. More importantly, the analysis of music sales shows clearly 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" (e.g. "Best of love songs") are no longer profitable, because users already have the hits, and do not want to buy CDs in which they like only a fraction of the titles. 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.

The approaches to music selection can be examined according to these three goals: repetition, surprise, and exploitation of catalogues. We show in the next Section that current approaches only achieve partially the goals.

3. Approaches in Music Selection

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

3.1 The database approach

Query systems address database issues for storing and representing musical data. They propose means of accessing musical items using some sort of semantic information. 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). In all cases the database approach, however sophisticated, satisfies the goal of repetition, since it provides users with exactly what they ask for, so no novelty is achieved.

3.2 Collaborative filtering approaches

Collaborative Filtering (CF) Systems [19] address the "surprise" goal, i.e. issue personalized recommendations to users. CF has had some success in the field of music selection [1, 5, 6, 11] as well as in other domains such as books and news.

CF is based on the idea that there are *patterns* in tastes: tastes are not distributed uniformly. These patterns can be exploited very simply by managing a profile for each user connected to the service. The profile is typically a set of associations of items to

grades. In the recommendation phase, the system looks for all the agents having a similar profile the user's; will look for items liked by these similar agents, which are not known by the user, and will recommend these items to him/her.

Experimental results show that the recommendations, at least for simple profiles, are of good quality, once a sufficient amount of initial ratings is given by the user [19]. However, there are limitations to this approach, which appear by studying quantitative simulations of CF systems, using work on the dissemination of cultural tastes [4, 2]. The first one is the inclination to "cluster formation", which is induced by the very dynamics of the system. CF systems produce interesting recommendations for naïve profiles, but get stuck when the profiles get bigger: eclectic profiles are disadvantaged. Another problem, shown experimentally, is that the dynamics favors the creation of hits, i.e. items which are liked by a huge fraction of the population. If hits are not a bad thing in themselves, 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 users items which are similar to known items. However, cluster formation and uneven distribution of chances for items (e.g. hits) 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).

4. On-the-fly Music Program Generation

Instead of proposing users sets of individual titles, we propose to build full-fledged music programs, i.e. sequences of music titles, satisfying particular properties.

4.1 General idea

There are several motivations for proposing music programs rather than unordered collections of titles. One is simply based on the recognition that music titles are rarely listened to in isolation: CD, radio programs, concerts are all made up of temporal sequences of pieces, in a certain order. This order is most of the time significant: different orders do not produce the same impressions on listeners. The craft of music programming is precisely to build coherent sequences, rather than just select individual titles.

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. As a consequence, an individual title may not be particularly enjoyed by a listener *in abstracto*, but may be the *right piece at the right time* within a sequence.

Rather than focusing on similarity of individual titles, we can exploit properties of sequences to satisfy the three goals of music selection. The proposal is therefore the following. First we build a database of titles, with content information for each title. Then we specify music programs by giving 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.

4.2 Working example

The problem is therefore to build music programs seen 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. In the next sections, we will focus on the format of the database and the nature of constraints.

Here is a "liner-note" description of a typical music program. The properties of the sequence are grouped in three categories: user preferences, properties on the coherence of sequences, and constraints on the exploitation of the catalogue. This example describes a music program called "Driving a Car", ideally suited for car music:

User preferences
- No slow/very 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".

Constraints on the coherence of the sequence
- Styles of titles are close to their neighbors (successor and predecessor). This is to ensure some continuity in the sequence, style-wise.
- Authors are all different

Constraints on the exploitation of the catalogue
- Contains twelve different pieces. This is to fit on a typical CD or minidisk format.
- Contains at least 5 titles from the label "Epic/Sony Music". This is a bias to exploit the catalogue in a particular region.

5. Database of Music Titles

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

5.1 Format of the database

Each item is described attributes which take their value in a predefined taxonomy. The attributes are of two sorts: technical attributes and content attributes. Technical attributes include the name of the title (e.g. "Learn to love you"), the name of the author (e.g. "Connick Harry Jr."), the duration (e.g. "279 sec"), and the recording label (e.g. "Epic/Sony Music"). Content attribute 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").

In the current state of our project, the database is created by hand, by experts (including the third author). However, it should be noted that 1) some attributes could be extracted automatically from the signal, such as the tempo, see e.g. [18] and 2) all the attributes are simple, i.e. do not require sophisticated musical analysis.

5.2 Taxonomies of values and similarity relations

An important aspect of the database is that the values of content attributes are linked to each other by similarity relations. These similarity relations are used for specifying constraints on the continuity of the sequence (e.g., the preceding example contains a constraint on the continuity of styles). More generally, the taxonomies on attributes values 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 worth mentioning, because it embodies a global knowledge on music that the system is able to exploit.

Various classifications of musical styles have been designed, particularly by internet music retailers [1, 10]. These classifications are mainly designed for a query-based approach. For instance, the taxonomy of Amazon is a tree-like classification which embodies a relation of "generalization / specialization" between styles: "Blues" is more general than "Memphis Blues". As such, it is well suited for navigating in the catalogue to find under-specified items, but it does not represent similarities between styles, for instance, having a common origin, like, say, "Soul-Blues" and "Jazz-Crooner".
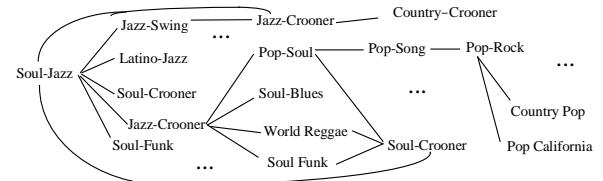

Figure 1. Our taxonomy of musical styles. Links indicate a similarity relation between styles. "Jazz-Crooner" is represented as similar with "Soul-Blues".

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

6. CSP for Building Music Programs

Building music programs that satisfy sets of constraints is a combinatorial pattern generation problem. The problem is the opposite of pattern matching: 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 hard 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 allows to state declaratively the properties of solutions, and use general purpose algorithms to find them. These algorithms are based on the notion of constraint filtering: each constraint is taken individually to reduce the search space; this reduction - filtering - depends heavily on the constraint [17]. The whole issue of CSP is to identify general purpose constraints that can be used to specify particular classes of problems (so-called "global constraints"), and design efficient filtering procedures for these global constraints.

In the next section, we formulate the music program problem as a finite domain CSP. In the following section we propose a small set of global constraints to specify most of music programs. The resulting system, *RecitalComposer* is composed of a constraint solver, a database and associated taxonomies of attribute values.

6.1 CSP for building sequences

A music program satisfying constraints can be seen as a solution of a finite domain CSP: the sequence is composed of successive items represented as variables $v_1, v_2, \ldots v_i$. Each variable v_i represents the ith item in the sequence. The domain of the variables v_i is the - finite - catalogue to look from. Constraints establishing properties of the sequence are expressed in the CSP paradigm, and hold on the variables v_i, and their attributes v_i^j (see 5.1). This formulation yields a hard combinatorial problem. To give an idea, finding a sequence of 20 items, with 100,000 possible values for each item (about the size of a catalogue of a major label) represents a search space of 10^{100}. Efficient filtering procedures have to be designed in order to find solutions in a reasonable time.

Constraints on sequence have been studied in the community of constraint programming. For instance, the *Sequence Constraint* of CHIP [3] is designed to enable the expression of complex regulation rules. This constraint is used to control the occurrences of some patterns in a sequence. Specific filtering techniques are designed to handle this sequence constraint efficiently.

This constraint is typically used for complex timetable problems to specify regulations rules (e.g. any employee has at least twice a month a two-day rest). Another kind of sequence constraint is the *Global Sequencing Constraint* [15] of IlogSolver [13]. This constraint 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 [16] 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 item's attributes (e.g. *style*, *tempo*, etc). For instance, we want to have five Jazz music titles and 3 slow motion titles in a raw. These requirements cannot be expressed neither in terms of the Sequence Constraint of CHIP nor of the Global Sequencing Constraint. They are stated by a set of individual cardinality constraints. This approach raises efficiency issues that led us to develop specific filtering techniques, not described in this paper for reasons of space.

6.2 Similarity, difference and cardinality

The constraints needed to specify music programs (user preferences, program coherence, and exploitation of the catalogue) can be expressed using a small number of global constraints: similarity constraints, difference constraints, and cardinality constraints. We describe below these constraints, with examples of use.

6.2.1 Similarity constraints.

This constraint states that within a given range, the 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 :

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

Where *a* and *b* are integers representing indexes, *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.

6.2.2 Difference constraints

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

$$D(I, j) \text{ meaning that:}$$

All items $v_i, i \in I$, have pairwise different values for attribute *j*. Here, *I* is a set of item indexes, *j* is an attribute index. 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 well-known *all-different* constraint, for which efficient filtering procedures have already been proposed in the literature [14].

6.2.3 Cardinality constraints

These constraints allow to 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 cardinality constraints: cardinality on items and cardinality on attributes.

6.2.3.1 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 :

$$CI(I, j, a, b, E) = | \{i \in I; v_i^j \in E\} | \in [a, b]$$

Where I is a set of item indexes, *j* is an attribute index, *a* and *b* are integers and E is a subset of the possible values of attribute *j*. For instance, this constraint can be used to state that there should be between 4 and 6 pieces within a the first 10, whose style is "Rock".

6.2.3.2 Cardinality on attribute values

This constraint states that the number of different values for some attribute is within [a, b]:

$$CA(I, j, a, b) = | \{v_i^j; i \in I\} | \in [a, b]$$

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

6.2.4 Example

We can now express the example of Section 4.2 as a CSP on sequences, by instantiating the global constraints defined above.

- No slow/very slow tempos: simple unary constraints on each variable.
- At least 30% female-type voice: cardinality constraint on attribute "voice-type".
- At least 30% purely instrumental pieces: cardinality constraint on attribute "music setup".
- At least 40% brass: cardinality constraint on attribute "instrument".
- At most 20% "Country Pop" style: cardinality constraint on attribute "style".
- One song by "Harry Connick Jr": cardinality constraint on attribute "author".
- 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 twelve different pieces: standard all-diff constraint on variables.
- Contains at least 5 titles from the label "Epic/Sony Music": cardinality constraint on attribute "label".

A solution of this problem is listed in Figure 2. The solution is computed within a few seconds by our Java prototype, an extension of the framework described in [17] with sequence constraints, and a sample catalogue containing 200 titles.

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

Figure 2. A Solution of the music program defined in Section 4.2.

7. Evaluation

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

7.1 Technical evaluation of the CSP approach

The current prototype was used on sample database of about 200 titles, using a Java prototype. Solutions are computed within a few seconds. Because we do not have so far a full database with more items, 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, using a non optimized Java prototype.

Experiments on databases larger by an order of magnitude are in progress and not reported here, but we claim that such an increase in size do not pose any problem for two reasons: 1) The database may be split up in smaller domains of interest for the solver, using simple heuristics, and 2) the increase of the number of "different" items is not related to the number of backtracks: the only relevant parameter is the "density" of solutions in the search space, which, in our case, is always high.

7.2 Evaluation of resulting sequences

The solutions found by *RecitalComposer* satisfy two goals of music selection: user preferences (repetition) are satisfied by definition, and exploitation of the catalogue is systematic: no clustering or bias is introduced, so 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. The basic idea is that unknown titles may be inserted in music programs with

a high probability of being accepted, because of the properties of continuity in the sequence. Experiments are currently conducted to compare 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 human experts would not have thought about.

8. Services

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 using an interface, and launch the system on a database. In this system, the user has full control on all the constraints, so it is aimed at professionals, who want to express all the properties of the desired programs.

Applications targeted at non professionals have also been developed using RecitalComposer. *PathBuilder* is an application in which the user can specify a starting title and an ending title. The system contains hidden constraints on continuity of styles, and tempos are fixed. For instance, find a continuous path between Céline Dion's "All by myself", and Michael Jackson's "Beat it". Another similar application allows users to specify only the stylistic structure of the program. This may be used for instance for creating long programs for parties, in which you know in advance the structure (e.g. begin with Pop, then Rock, then Slows, etc.).

Finally, our approach can be used to produce music programs in specific styles, by adding domain specific constraints. A prototype application dedicated to Baroque music has been designed and implemented in our lab. The application allows to build various "recitals" in the domain of Baroque harpsichord music. Recitals of Baroque music (XVIIth 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: at this period of musical history, recitals where allowed to modulate - i.e. change tonality - only once. Other constraints concern the structure of the recital (introductory part with necessary piece types), as well as necessary alternation of piece types.

Other applications are envisaged for set-top-boxes services and digital audio broadcasting which we do not detail here for reasons of space.

9. Conclusion

RecitalComposer is an enabling technology for building high-level 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 queries. One of its main advantages over other approaches is that it produces ready-for-use music programs which satisfy the goals of music selection: repetition, surprise, and exploitation of catalogues.

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; others such as similarity relations between styles could be extracted using collaborative filtering techniques.

10. References

[1] Amazon Music Store web site, http://www.amazon.com, 1998.

[2] Cavalli-Sforza, L. Feldman, M. *Cultural Transmission and Evolution: a Quantitative Approach*, Princeton University Press, 1981.

[3] Dincbas, M. Simonis, H. and Van Hentenryck, P. "Solving Large Combinatorial Problems in Logic Programming" Journal of Logic Programming, vol. 7 (1), 1990.

[4] Epstein, Joshua M. *Growing Artificial Societies: Social Science from the Bottom Up*, MIT Press, 1996.

[5] Firefly web site, http://www.firefly.com, 1998.

[6] Infoglide web site, http://www.infoglide.com, 1998.

[7] Memon, N, Wong, P. W. Protecting Digital Media Content. *Communications of the ACM*, July 1998, pp. 34-43, 1998.

[8] Meyer, L. *Emotions and meaning in Music*. University of Chicago Press, 1956.

[9] Mpeg-7, Context and objectives, International Organization for Standardization, report ISO/IEC JTC1/SC29/WG11, Oct. 1998.

[10] MusicBoulevard web site, http://www.musicblvd.com, 1998.

[11] MyLaunch web site: www.mylaunch.com, 1998.

[12] Narmour, E. *The analysis and cognition of melodic complexity*. University of Chicago Press, 1992.

[13] Puget, J.-F. and Leconte, M. "Beyond the Glass Box: Constraints as Objects," ILPS'95, Portland, Oregon, 1995.

[14] Régin, J.-C. A Filtering Algorithm for Constraints of Difference in CSPs AAAI'94, Seattle, WA, pp. 362-367, 1994.

[15] Régin, J.-C. and Puget J-F. A filtering algorithm for global sequencing constraints, proceedings Third International Conference on Principles and Practice of Constraint Programming, 1997, pp. 32-46.

[16] Régin, J.-C. "Generalized Arc Consistency for Global Cardinality Constraints" AAAI' 96, Seattle, WA, 1996.

[17] Roy, P. Pachet, F. Reifying Constraint Satisfaction in Smalltalk. *Journal of Object-Oriented Programming* (JOOP), 10 (4), pp. 43-51, July/August 1997.

[18] Scheirer, E. D. (1998). "Tempo and beat analysis of acoustic musical signals." Journal of the Acoustical Society of America 103(1): 588-601.

[19] 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.

[20] Smith, D. Melara, R. Aesthetic preference and syntactic prototypicality in music: 'Tis the gift to be simple, Cognition, 34 (1990) pp. 279-298.