EXPERIENCE WITH A HARMONY-LEARNING PROGRAM

Suzanne Haig, Gershon Horowitz, Malcolm C Harrison

Computer Science Department, New York University
Warren Weaver Hall, New York NY 10012, USA

ABSTRACT

This paper describes a series of experiments with a harmonization program. The input to the program is a set of melodies and their desired harmonization, and the program learns these harmonies by adjusting a set of tables. The ability of the program to learn musically significant styles, and generate harmonies in these styles, is discussed.

1. Introduction

In a previous paper [Harrison, Haig, Horowitz 1989] we described an algorithm for adding harmony to a melody. The algorithm uses an evaluation function for assessing the "goodness" of a harmonization, along the traditional lines suggested in [Piston 78, Schoenberg 78]. The function takes into account a number of criteria, including a measure of the compatibility of a chord with the melody notes which it accompanies, and a measure of the compatibility of the chord with the preceding chord. These measures are stored in the form of tables, whose values can be modified to provide different evaluation functions. (In our previous paper we described an evaluation function which would take into account a number of other properties of the harmony, including the duration of notes and chords, and position of chords. In the experiments reported here, we are just using the two tables, with fixed chord positions and weight proportional to the duration of notes).

The overall objective of our work is to capture one or more aspects of harmony, in the sense that the harmonies generated by our program might be recognizably similar to a particular musical style. Our objectives with this algorithm alone are quite modest in this respect; we expect that musically significant inventions will require further techniques. On the other hand, we are hoping to be able to establish a set of tables whose intelligent use (by a musician or a more sophisticated higher-level program) can generate useful or interesting harmonizations. Use by a musician might take the form of running the program, listening to the results, and modifying the output by vetoing or forcing some of its choices. A secondary objective is to produce a useful tool for analysis of harmony.

For each melody, the algorithm constructs a graph with each node representing a chord and a sequence of melody notes, and each edge representing a chord-chord transition. Our algorithm identifies that harmonization which maximizes the evaluation function, using a search for the best path through the graph.

The amount of information in the tables is, however, very large (for 12 notes and 30 chords a total of 12*30 + 30*30 = 1260 parameters), and a more systematic approach was needed. Since our evaluation function is linear in the values in the tables, a linear learning procedure can be used to find values of the tables which will "correctly" (i.e. as harmonized by the composer or performer) harmonize a group of melodies when such values exist.

In this paper we present the results of experiments with the learning aspects of this program.

2. The Learning Procedure

Our learning procedure is somewhat different from the usual linear classification algorithm [Nilsson 1965]. We will describe how it works for a single piece, and then describe the extension for multiple pieces.

For each piece we provide the "correct" harmonization (the one used by the composer or performer), and compute the goodness value according to the current values of the table pair. This value is of the form $\sum H_i T_i$, where the T_i are table values and the H_i depend only on the melody and the harmonization. We compute the H_i, which correspond to one test case (the "positive" example). Our algorithm then generates the best solution according to the table pair, and computes its H values, say H'_i. The values of the tables are then adjusted, according to:

$$T_i \rightarrow T_i + \alpha (H_i - H'_i)$$

where α is a parameter which can be chosen to adjust the learning speed. This is repeated until the correct harmonization is picked as best (we say that the phrase has "resolved"), or until resolution seems unlikely.

Since the H_i values for any harmonization are independent of the table pair values, this process generates a single positive example and a finite number of negative examples. This means that, as in the classical case, resolution, if possible, is assured in a finite number of iterations.

For many pieces, we construct a single large piece by concatenation of the individual pieces, with a marker between pieces which suppresses the effect of the chord transition. If this resolves, this means that all the constituent pieces resolve with the same table pair.

3. Initial Experiments

Early results suggested that the harmonies used by different musical styles did correspond to different tables, and such tables could regenerate approximations of these styles on new melodies. On the other hand, our program was not able to adjust its parameters to generate exactly the correct harmonies for a substantial number of melodies, even when these are musically related (by the same composer, for example). For simple melodies and harmonies such as folk tunes and blues we could do quite well (7 melodies were correctly harmonized by the same table), but melodies with a greater variety of chords were not as easy. For Bach chorales, for example, we could get about 70% correct for 10 pieces, and less than this for Cole Porter songs. Even so, the use of these tables with other melodies produced not unpleasing results.

Of course it is not reasonable for us to expect 100%, since in many cases there are a number of chords which could be reasonably chosen. The particular choice is sometimes made arbitrarily (one day the performer plays one chord, next day another), and sometimes for reasons which are inaccessible to our algorithm (e.g. voicing considerations, or lyrics).

One obvious deficiency of the linear evaluation function approach is that it is not capable of expressing context-sensitive decisions (e.g. play chord 1 the first time it is appropriate, chord 2 the second; strongly prefer cadences at the end of a piece). To overcome such difficulties we propose to apply our algorithm to phrases, rather than to complete melodies. Initially we will specify the phrases, but we anticipate wanting to automate this. In use, the harmonic strategy would be provided (by a human user, or a higher-level program) by specifying the table pairs to be used for each phrase, while the harmonic tactics would be determined by our algorithm.

4. Phrase Subsets

We propose to use our learning algorithm to identify subsets of these phrases which are compatible (i.e. whose harmonies can be generated by the same pair of tables). We can not test all possible subsets of a set of phrases, so we will try to extract as much information from each test as possible. We do this according to the following algorithm:

1. For a set of phrases and their preferred harmonies, run the learning program till no further improvement is occurring, keeping track of the number of errors at each chord position; if there are no errors, stop.

2. Identify the chord positions with the largest number of recent errors; if there is more than one, choose that which would cause the most improvement if fixed.

3. Force the choice of the correct chord at this position, and go to step 1.

This will clearly terminate with a set of phrases which are correctly harmonized except for certain chord positions. The idea is that the positioning of the errors will provide information about which phrases are least compatible; these phrases are dropped from the set, and the process iterated until a compatible subset remains.

5. Multi-chord Sequences

It is widely accepted that sequences of three or more chords, rather than just two, play an important role. Our algorithm does not account for these directly, but we have been able to simulate three-chord sequences to some extent in the following way. For a set of candidate sequences, rename each middle chord to be unique. Enter these chords in the tables in the normal way. The result is that these sequences can have their table values adjusted independently of the same chords not used in the sequences. The main disadvantage of this technique is that it permits too much flexibility, since these new chords are free to build up associations with arbitrary other chords, so results are difficult to interpret. For example, it is not clear to what extent it permits four-chord sequences to be recognized as two overlapping three-chord sequences (though it can recognize five-chord sequences).

6. Results

At the time of writing we have tested the learning procedure on portions of 10 Bach chorales, each regarded as a single phrase. For 167 notes and chords, our program was able to adjust the tables so that 116 chords were correctly chosen (i.e. 51 errors). After the forcing procedure was applied, 31 chords needed to be forced to get resolution. The individual pieces varied from one force out of 15 to six forces out of 21. We also noted that when we chose smaller subsets of the chorales, and when we broke the pieces into phrases, the errors were for the most part in the same chord positions.

We have also tested two groups of 7 Bach chorales whose harmonizations were specified to contain 10 three-chord sequences. This gave 60 percent fewer errors than the experiments without the three-chord sequences. The significance of this is not clear, however, since this could be attributed simply to the availability of more parameters which can be adjusted to fit the data.

7. Conclusions

Our results so far are only preliminary, but are in general encouraging. The learning algorithm appears to converge, when it is going to, within several hundred iterations. This is a considerable improvement over a process we used previously, in which we ran several iterations on one phrase before going on to the next; this showed much slower convergence, and much greater difficulty in determining non-convergence.

The fact that errors tend to be in particular chord positions also suggests that the process of phrase subsetting is worth further investigation.

A remaining difficulty is that our learning procedure cannot distinguish between minor and major errors. One approach which we plan to investigate is to take into consideration how far away the desired chord is from being chosen (for example, if we veto the current choice, and the desired chord is the second choice, this would be more acceptable than if it was the third or fourth choice).

8. References

Cope, D. 1987. "An Expert System for Computer-assisted Composition." *Computer Music Journal* 11(4):30-46.

Ebcioglu, K. 1987. *Report on the CHORAL Project: an Expert System for Harmonizing Four-Part Chorales.* Yorktown Heights, NY: IBM Research Division.

Harrison, M. C., Haig, S., and Horowitz, G. 1989. "A Shortest-path Algorithm for Musical Harmony" *International Computer Music Conference*, Columbus, Ohio.

Nilsson, N. J. 1965 *Learning Machines*, McGraw-Hill, New York.

Piston, W. 1978. *Harmony*, 4th ed. New York: W.W. Norton & Co.

Schoenberg, A. 1978. *Theory of Harmony.* Berkeley, CA: University of California Press.

Smoliar, S. 1980. "A Computer Aid for Schenkerian Analysis." *Computer Music Journal* 4(2):41-59.

Winograd, T. 1968. "Linguistics and the Computer Analysis of Tonal Harmony". *Journal of Music Theory 12:2-49.*