

F. Pachet and P. Roy, Musical Harmonization with Constraints  
Shane Hoversten

The paper opens by pointing out the history of the relationship between computer science and music. Music has long been a source of interest to computer scientists, and lends itself to CS analysis, because it's an area with a long history, an extensive set of corpora, and a well-developed formalism. Music is amenable to the same sort of analysis that words are, and can reap the benefit of the assorted techniques that have been developed for symbolic and linguistic manipulation.

Tonal music, in particular, makes a nice target for computer analysis and manipulation, since it, too, has a long history, and a lot of treatises on the rules that govern it. Tonal music breaks the piece into sets of notes, called scales, which are themselves organized hierarchically.

The paper's focus is on harmonization, for which there is also a large body of work, with different sets of rules describing precisely what harmonization is, and how it should be performed. For instance, there are vertical rules, dealing with chords, and horizontal rules, dealing with repetitions of structure in time. The authors liken these rules of harmonization to a musical syntax.

One way of automatically composing harmonizations is to codify some set of rules into a Markov chain. Such generative techniques have been discussed before in other papers; using them for harmonization subjects them to all the usual problems, chiefly that of structure blindness: without an extremely ornate grammar, the music produced would lack the global coherence that gives music much of its character.

The authors then describe another technique for harmonization, constraint satisfaction. This is a well-known area of computer science, traditionally focused on mathematical programming and optimization. The nature of its application to music is straightforward, but a heavy demand is made on the user to codify the particular harmonization rules the CS system is to enforce. Depending on which set of rules one is following, this can be a significant task.

There are several flavors of harmonization, which grant differing levels of freedom to the CS system and which are thus of a range of difficulty. One might impose the melody, or the bass, and then have the system compose the other three voices. One might impose the bass as well as figures, which are "hints" that will further constrain the CS system's choices for the other three voices. Or one might impose the melody AND the bass, and leave the CS system to compose the other voice. (In my naive knowledge of arithmetic it would seem that there would then be two voices left to compose, but I never was very good at math.)

Various attempts at building CS systems for this harmonization problem are described. I will omit the particulars of the assorted systems, and instead concentrate on an interesting problem. Music consists of individual notes, as well as larger structures composed of

those notes. One such structure is the chord, which are groups of notes sounded simultaneously. Since various of the constraints that one might impose on the harmonization problem operate on the chord level, finding a way to introduce this aggregate into the system has the potential to greatly reduce the computational complexity of the problem. This idea applies to much larger domains than this one; the real issue is of manipulating an abstraction that describes the problem space at the appropriate level of intent.

This idea is loosely related to the comments I made on using genetic algorithms in my summary of the Wiggins paper. Introducing random note-mutations into the musical chromosome will never truly smooth out the difficulties in a piece, since music doesn't impose rules only at the note level. I think a vital component of any generative technique is in representing and manipulating the entities at the appropriate level of abstraction. This is an idea natural language processing is wrestling with; hopefully, as with many of the other tools developed for that domain, the solutions, when they arrive, will find their way into this one.

The authors close the paper by asserting that we can consider the problem of generating harmonization solved, since CS, when properly applied with a full range of techniques like arc-consistency, does such a nice job. They further assert that we should now concentrate efforts on finding out "what makes melodies interesting." I admit to finding this conclusion rather odd. Consider this paraphrase: "We can now automatically harmonize pieces of music. Of course, it sounds like shit." Assuming there is some merit in this paraphrase, could we really consider a domain solved that could be so described? Producing a body of solutions that meets some artificial characterization of the solution space seems disingenuous. We should consider the problem of automatic harmonization "solved" when it actually produces harmonies people want to hear.