Artificial Music Generation

An AI Introduction to Composing Music

Abstract

In this paper, we introduce the domain of music generation by computers. We highlight two four-part harmonizing systems, one rule-based and one evolutionary (GA), as well as two evolutionary jazz solo generating systems. We give an overview of their functions and research if they're able to produce human sounding music and if they perform this task in a human way. We conclude that for large-scale musical structure, a rule-based system is necessary, while for interesting melodies with direction a GA is inevitable. Therefore, we argue that a combination of both systems should be made in order to create human sounding music in an as-close-as-possible-to-human way.

Gertjan L. Wijnalda

Department of Artificial Intelligence, Vrije Universiteit Amsterdam
RedAnt Design

May 2002

Author Information

The author is a student at the Department of Artificial Intelligence at the Vrije Universiteit, Amsterdam (student number: *1152785*). He is also one of the founders of RedAnt, a company that specializes in websites entirely designed with optimal human-computer interaction in mind. As for musical experience, he is a long-term singer and composer of modern music and has written over fifty individual songs on piano, guitar and vocals.

The author is most efficiently contacted by e-mail at *gertjan@cs.vu.nl* or *g.l.wijnalda@redant.nl*. His VU homepage (where MIDI files of the examples in this paper are available) can be found at *http://www.cs.vu.nl/~gertjan* and his personal homepage at *http://www.publictear.com*.

Table of Contents

1 Introduction

1.1 Bach at the Court of Frederic the Great

Music has always been an important part of life. Which love couple does not have its own song? However, music does not longer play the part it had in some environments. For example, every self-respecting emperor or king in the 18th century had his own orchestra and music school.

In 1747, King Frederic the Great once summoned the old Johann Sebastian Bach to come to his court and play on his new Silbermann pianos. During this meeting, the King gave to Bach a musical theme and ordered Bach to improvise him a *fugue*[1]. After that, as is being said, Bach improvised a five voice fugue and later even an eight voice fugue. Of course, this story was written down as much as 27 years later, when the old Bach was already dead for 24 years, but it still shows the fame Bach had acclaimed in those days (David & Mendel, 1966).

1.2 Composing a Fugue

How extraordinary a six voice fugue is, can be shown if you recall that the entire *Wohltemperierte Klavier* (Bach, 1722), which consists of forty-eight preludes and fugues, only contains two five voice fugues – a six voice fugue is not even there! According to Hofstadter (1985), improvising a six voice fugue can possibly be compared to playing sixty games of blindfold chess and winning them all – simultaneously! Improvising an eight voice fugue is really a superhuman exercise.

It must, of course, be noted that the above story might presumably be exaggerated. However, the five voice fugue that ends the *Wohltemperierte Klavier* is so extremely complicated that we can safely conclude that Bach really must have been a genius. Composing fugues is an extraordinary difficult task, because they have to be accordant with both the rules of the fugue as with musical (melodical and harmonical) rules. It is this exercise that forms the main scope of this paper.

1.3 Research Question

What will we research in this paper? First, we'll see if it's possible for computers to compose three voices in harmony with a first, given, voice. This task can be viewed as the composition of the lowest three voices of a fugue after the first (highest) voice is written (please note that this is definitely not how every composer works, it is only *one* approach to the task). Secondly, we'll also look at the problem the other way around, creating instrumental solos based on given harmonies. The main question we will investigate is: are computers able to perform this task, and are they able do this is in a 'human' way?

In order to investigate this properly, we will first introduce the concept of four-part harmonization, the art of writing voices that are both consistent with (in this case) three other voices, both melodically and harmonically. Please note that this paper is written with the non-musician in mind, for those interested in AI and its applications in music in particular.

[1]) *Fugue*: a piece that, usually, has four voices and is based on a musical idea (a theme) of approximately eight bars. This theme is repeated in each of the voices in various heights and in various speeds, or even the other way around. A fugue obeys to very strict rules both in form as in harmony, which are beyond the scope of this paper.

2 Composing Music in the Old-Fashioned Way

In order to accurately research the question posed in the previous section, we will have to form an idea about the characteristics of the domain we're working in. First, we will discuss the physical foundation and practical tuning of harmonics, and briefly say something about harmonic progression and chord structures. With this knowledge as a starting point, we can discuss four-part harmonizing.

2.1 An Introduction to Harmonics

Our sense of music is strongly motivated by physics. Western tonal music is based on harmonic frequency ratios. Consider for example the triad (chord) shown in Figure 1.

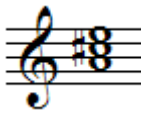


Figure 1: An A major triad.

This chord is called an A major triad. A triad because it contains three notes, 'A' because its base note (in this case, the lowest note) is an **a**, and major (as opposed to minor) because of the harmonic ratios between the notes. The base note, or tonic, is perceived as the fundamental note of the chord and thus the chord is named after it. Its frequency is, according to Western standard, exactly 440Hz. The highest note is an **e**, and is called the dominant note in the chord, since it is the next most stable note after the tonic. The **e** is called the *fifth* note of this particular key: from **a** to **e** is (**a, a#, b, c, d, e**) five steps. It has a frequency of 660Hz, exactly 1½ times the frequency of **a** – a harmonic ratio of $\frac{3}{2}$. The middle note, **c#**, has a harmonic ratio of $\frac{5}{4}$ (550 Hz).

Following this simple routine, we can find the frequency of a note by its relationship with the tonic: the fifth note (e) is $\frac{3}{2}$, the *third* $\frac{5}{4}$. The *second*, the note only one tone above the tonic, for example, will have a harmonic ratio of $\frac{9}{8}$. As we get closer to unity, the notes become less likely to appear in a composition in the given key (Campbell and Greated, 1988).

So, for this given key (A major), the harmonizing notes are determined by the frequency relationship with the given key. Of course, this applies for all keys. For example, when we lower the third note by a semi-tone, we get an A minor key. The same rules apply, only the third note is different. In both keys, an **a#** (or **b*b***) is unlikely to occur, since it is, simply said, too close to the tonic[2].

[2]) These 'rules' for (not) using specific notes in a key have been widely discussed throughout the history of Western music. We will focus on the styles that were common during the Renaissance, Baroque and Classical periods in music (from the end of the 16th century till c. 1815). After this period, composers like Arnold Schönberg and Anton Webern rejected the whole idea of a tonic and therefore a key as base for a composition. Later, their movement culminated in the introduction of *serial music* by composers such as Karlheinz Stockhausen and Pierre Boulez. For more information see, for example, Schönberg (1984) and Boulez (1963).

2.2 Equal Temperament

Even though the above is entirely based on physics, a little 'inconsistency' occurs here. I will illustrate this by an example from Wiggins (1998), dealing with the intervals shown in Figure 2.

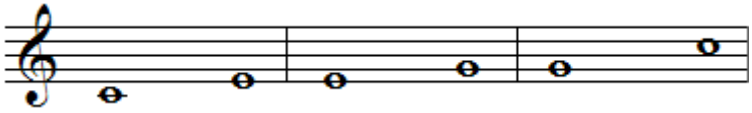


Figure 2: Intervals.
From ***c*** *to* ***e****,* ***e*** *to* ***g*** *and* ***g*** *to* ***c'****.*

Since the first and the last notes are both the same note with a difference of one *octave*, namely **c** and **c'**, we would want to end up with an octave when calculating the frequency as well. So, the frequency we would want to find would have to be equal to 2. The problem is this: every interval shown has a frequency ratio of $\frac{5}{4}$, so we end up with a frequency ratio of $\frac{5}{4}\times\frac{5}{4}\times\frac{5}{4}$ which is approximately 1.953…

In order to solve this inconsistency, equal temperament was introduced in the late 17th century; an event that had so much impact that it was celebrated by Bach himself by his *Wohltemperierte Klavier*. The simple but adequate solution is this: every semitone is tuned equally (hence the name equal temperament), thus giving a frequency ratio of $\sqrt[12]{2} \approx 1.059$. Three major thirds in this system gives 12 semi-tones: $\left(\sqrt[12]{2}\right)^{12} = 2$!

In this system, harmonies can easily be defined as the *x*th note of a given key, by using $\left(\sqrt[12]{2}\right)^{x}$. It is this way of tuning that modern keyboard instruments use to play a piece of music in any given key.

2.3 Harmonic Progression

In the so called Classic or traditional harmonics, triads based on the first, fourth and fifth note form the base of a piece. The starting point and finale of the music is **I**, the tonic. Together with the triads based on the dominant (**V**) and the subdominant (**IV**) the tonic triad is called a *main triad*. The triads on the other tones of the key (**II**, **III**, **VI** and **VII**) are considered as replacements of the *main triad*.

Chords (triads) can play different roles in different keys. For example, the triad based on the tones **d** – **f** – **a** (a D minor chord) is **II** in the key of C major, since it is the chord based on the second note in that key, namely **d**. It is as well the triad based on the fourth note, **IV**, in the key of A minor, which is **d** as well. Based on this ambiguous meaning, such a chord can be used to *modulate* from one key to another. In this new key, the function of the original major triads is lost, and new major triads are based on the notes of that key.

Modulating from C major to A minor is easy, since both keys feature the same chord. However, most of the times modulation is not so trivial, requiring additional keys for the entire modulation. Therefore, modulation is considered one of the most important and difficult parts of harmonics.

2.4 Four-Part Harmonization

Harmonizing voices is based on the idea of harmony as described in the previous sections. You can see that you can determine the key of a given piece and thus limiting the notes that are 'permitted' in this key. I write 'permitted' here, because one can think of musical expressions (such as twelve-tone music) where chromatic intervals (intervals of only a semitone between notes) are perceived as striking. However, as said before, in this paper we will only deal with harmonies that composers in, for example, the Renaissance, Baroque and Classical style periods (such as J.S. Bach) viewed upon as 'well'[3] harmonies.

When harmonizing, one has to keep in mind that the 'flow' of the piece is not disturbed. Of course, we have unlimited possibilities to write whatever music we like, but for the kind of music we're talking about (defined above; we will refer to this music as Classical music from now on), there has to be a certain *logic*. This logic can be described as different *viewpoints* from which we can look at the piece of music. I will describe the four viewpoints briefly hereafter.

The *chord skeleton* provides us with a rough outline of the piece. No rhythmic information is available in this view, only the overall progression of the piece from one chord to another. For example, chords that are based on the fourth note of a key (written as **IV**), tend to progress to the chord based on the fifth note (**V**), which progresses to **I** in a very natural way. This *cadence* (an order of chords that is very well-known) 'feels' like an ending to a piece, for example, quite a large part of Mozart's oeuvre tends to end with this series of chords. This is a *horizontal view*, since we look at the progression of the piece in time.

We can also look at the voices *vertically.* This means we're taking one slice out of the piece and look at the notes of the voices. The notes that are played together should be, of course, in harmony with each other.

When we consider the voices *individually*, we must make sure that the melody does not look too extreme: not too many big intervals – these tend to sound as leaps in the melody. A melody always has to be an organic coherent piece that has a clear beginning and end.

Rhythmically, the voices should also form a coherency: when one voice is extremely slow, the others should follow in this speed, thus creating a coherent 'feel' to the music. This applies vertically as well as horizontally: there can be fast sections opposed to slow sections and faster voices opposed to slower voices.

When harmonizing four voices, referred to as four-part harmonization, one can see that quite soon we will have to look everywhere at the same time. Changing one single note (or even its rhythm!) can inflict enormous changes on other parts of the piece as well. This is the complexity of the exercise that Bach is said to perform in front of the King, and he had to do this with an unfamiliar theme… and four voices extra!

[3]) We're using loose definitions in this text, in order to make this paper readable for non musicologists as well. In the remainder of the paper, we will mostly apostrophes to indicate the used term is not a musicological but a less precise term.

3 Using Computers to Harmonize Music

In this chapter we will roughly outline two systems that are able to harmonize music. The first system we will discuss, CHORAL, is based on production rules, while the other utilizes genetic algorithms. After presenting the systems we will compare them and try to answer the research question posed in section 1.3 for both systems. Finally, the 'best of both worlds' is proposed and tested against the research question as well.

3.1 Rule-Based Harmonizing

Most systems in AI use a kind of rule-based system in order to solve the given problem. When writing a big problem, it becomes obligatory to make a division between two parts of the rule-base: the actual *knowledge base*, which contains all the domain specific knowledge, and an *inference engine* that exploits the knowledge. According to Van Harmelen (1989), an extendable and modifiable system can be created as long as we keep this division in mind.

The system we will discuss is an example of a rule-based system, designed to harmonize four-part chorales, CHORAL (Ebcioglu, 1993). It is based on the belief that the approaches until then, merely statistical and generalizing techniques, are too economic. Traditional music, usually composed making no use of a computer, 'does not seem to permit such economical representations' (p. 385). Ebcioglu (1993) argues that the task of composing music is so complex, that it requires vast amounts of domain specific knowledge, such as knowledge of harmony, strict counterpoint, fugue and orchestration (for example, the specific notes that are included in the typical range for the instrument playing a given voice).

The knowledge base of the CHORAL system was formalized using first-order predicate calculus and then coded in a specially designed programming language, BSL (Backtracking Specification Language). BSL is fundamentally different from other logic programming languages, such as Prolog, since each BSL program corresponds to an assertion in first-order predicate logic and executing the program amounts to proving the corresponding assertion. This is exactly what is needed for an ambitious music expert system. To implement this language on real computers, it's translated by a LISP program into 'very efficient backtracking programs in C' (p. 387).

We will now briefly outline the CHORAL system. The system is based on several viewpoints of the generated music, all controlled by a special *clock* process that ensures the turn-by-turn execution of several different viewpoints (discussed below). 'Thus, each step of the clock process determines the total amount of work done in one complete trip around the round-robin scheduling chain.' (p. 389). One of the advantages of this system is that it is, as Prolog, completely backtrackable. In this way, decisions made in one viewpoint that cannot be unified with other viewpoints, can be retracted so that a different solution can be tried. A viewpoint can be regarded as separate knowledge models. We will now discuss four viewpoints of the CHORAL system. These are not *all* viewpoints, but are chosen as to represent the ideas behind the system.

The first viewpoint is the *chord skeleton view*. In this view, the chorale is perceived as a sequence of chords with some additional symbols indicating key and degree within key (such as C-IV, meaning the chord based on the fourth note of the C key). This process is effectively used as

the clock process in CHORAL, and generates one chord every clock cycle (step). Without this viewpoint, compositions tend to sound Gregorian: without (obvious) direction. Rules that are part of this view are, for example, 'production rules that enumerate the possible ways of modulating to a new key [...] and heuristics that prefer Bachian cadences' (p. 393).

In the *fill-in view*, the actual chorale is generated in the form of suspensions, passing tones and ornamentations. For providing the notes, this view depends heavily on the chord skeleton view mentioned previously. Every step, the fill-in view generates 'the cross product of all possible inessential notes (passing tones, neighbour tones, suspensions and other chorale-specific ornamentations) in all the voices' (p. 393). Then, the unacceptable combinations of notes are removed from this collection of notes (*i.e.*, notes that are not in harmony with each other), and the desirable combinations are selected. For example, some Bachian clichés are encoded in the production rules of this view, such as a heuristic about following a certain type of ornament by another in the same voice.

The *merged melodic string view* is necessary for recognizing bad melodic patterns (such as melodies with a lot of repetitive notes). It observes a sequence of notes of the different voices from a purely melodic point of view, thus concentrating most of the melodic production rules in it. For example, constraints that enforce melodic interest[4] in the inner voices are placed here, so that voices are not only in harmony with the lead melodic voice and the bass (often the most important voices of the piece) but are also melodic in their own right. This is a sub view of the fill-in view, implemented in the same process with the fill-in as master view among them.

The last view we will consider in this paper is the *Schenkerian analysis view*. This view is based on a theory by Schenker (1979). The core of this theory is that the bass and descant lines of a chorale can be generated from a starting pattern by repeatedly applying a set of rewriting rules. The view reads the fill-in output and observes the chorale as 'the sequence of steps of two nondeterministic bottom-up parsers for the descant and bass.

The chorale melody is given as input to the program, and the views are cyclically processed: chord skeleton (adds a new chord to the chorale), fill-in (fills in the available chords and produces actual music until no more chords are available), Schenker-bass and Schenker-descant. Those last views are terminated when there are no more available notes. When a process does not have any input ready, it terminates, thus giving control to the next process without doing anything.

An example result of the program is shown on the next page (Figure 3). According to the author, 'the program has reached an acceptable level of competence in its harmonization capability' (p. 399), described as that of a talented student of Bach chorales. While the method used is not necessarily an accurate cognitive model of the human compositional process, the author argues that it seems to work and it is capable of producing musical results – results that can be perceived as music that could have been made by a human composer as well.

4) Melodic interest is defined by Ebcioglu (1993) in the paragraph in brackets on page 394, 'The melodic interest ... high-corner positions'.

Chorale no. 75

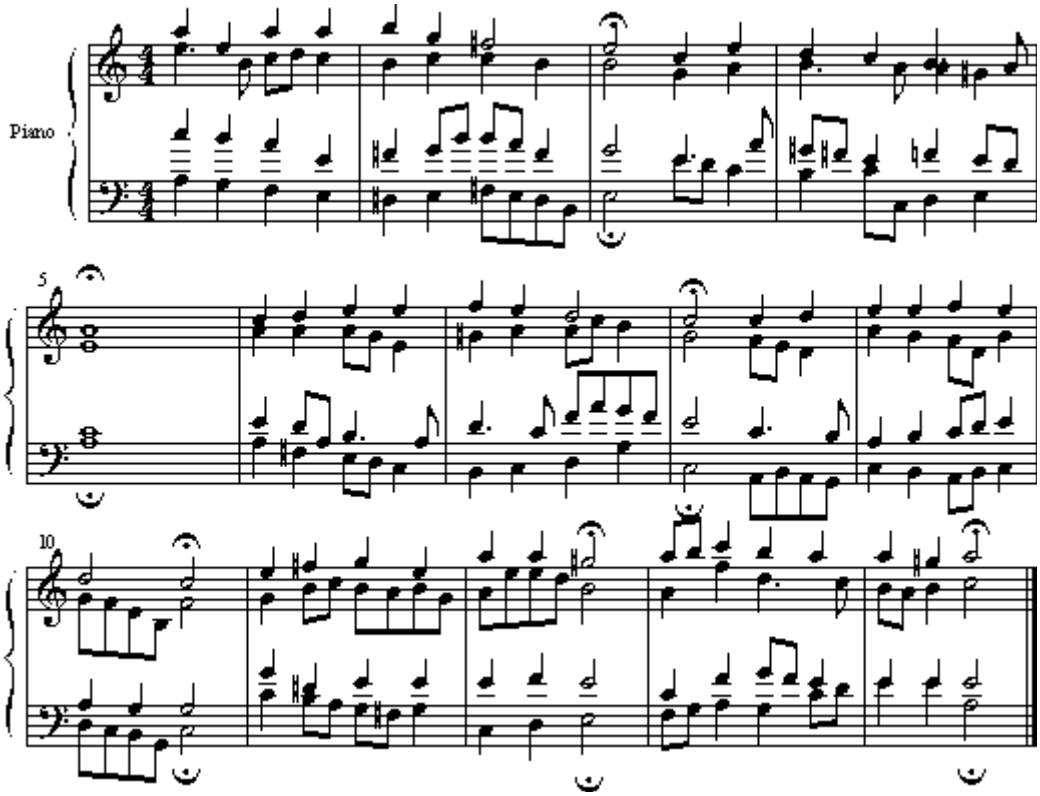


Figure 3: Example output of CHORAL.
(From: Ebcioglu, 1993, transcription: Wijnalda, 2002)

3.2 Evolutionary Harmonizing

Another popular approach to solve AI problems are evolutionary or genetic algorithms (GAs). The basic model of a GA is derived from the Theory of Natural Selection by Darwin (1859). We start with a pool of *chromosomes*, each representing a possible solution to the problem.

By using two types of operators, we are able to *mutate* the members of the pool (*mutation operators*) and to combine different parts of existing chromosomes to create new ones (*crossover operators*). These operators are based on the natural processes of asexual and sexual reproduction, respectively. In order to stimulate the 'survival of the fittest', as proposed by Darwin, we need to supply a *fitness function*. This function returns a measurement, often numerical, of how good the solution that a given chromosome represents is.

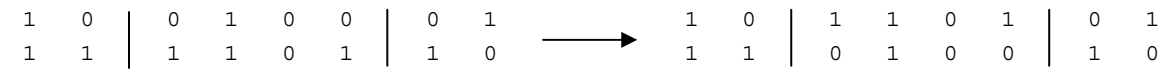


Figure 4: Two-point crossover.
This operator picks two points at random along two strings and swaps the contents of the strings between those points to produce children.

Schematically, we need the following components to create a complete GA:

- a *representation* for the chromosomes;
- an *initial population* of chromosomes;
- a set of *operators* to generate new solutions from current members of the population;
- information about when the operators should be applied;
- a *fitness function* that measures the validity of a given solution[5];
- a *selection method* that gives good solutions a better chance of survival.

In this chapter we will outline a system proposed by Wiggins *et al.* (1999) for the creation of four voice harmonic pieces based on a given melody. The goal of this system is the same as the CHORAL system outlined before, but it does not specialize in the J.S. Bach chorale-style, but in 'plain' harmonizing.

The algorithm that is used by Wiggins *et al.* is that of a GA with *steady-state reproduction.* This is not the only method for creating a GA, yet it is one of the simplest that effectively demonstrates the purpose of the system. First, an initial population of chromosomes is generated randomly. The evaluation function is then applied to all chromosomes. Then parent solutions are selected according to their fitness and operators are randomly applied to generate new chromosomes. This new chromosomes are then evaluated and if it is fitter than the least fit member of the population, it is substituted into the population. This process of selecting parents and generating children is repeated until a *stopping criterion* is reached[6].

[5]) As a fitness function, human users are often used in similar GAs. Such a GA is called an interactive GA (IGA). This system is not implemented in an interactive way, because that doesn't allow determining how useful a fitness function can be for this task. Also, 'human listeners tend to become more open to music on repeated hearings, and are prone to other inconsistencies based on mood, attention span, and so on' (p. 6).
[6]) Examples mentioned in Wiggins et al. (1999) include a situation of convergence (when all members of the population are identical), when a fixed number of evaluations has been computed or when a solution of a predefined quality is found (p. 3).

As a representation for the chromosomes, the authors of the system chose to create a matrix of $4 \times n$ strings of equal, fixed length. Each string represents one of the voices. The melody (supposed to be the input) is contained in the first string. The GA presented will then harmonize the melody with three other voices using the operators described below.

The fitness function judges the chromosomes according to musical criteria. These criteria are based on musical 'facts', that do not fall into the scope of the present paper, since we are not basically interested in the exact implementation that was chosen, but rather the method that the authors of this genetic algorithm harmonizer used. Nonetheless, we will discuss a few of the criteria used, to gain a little insight in the knowledge involved. For example, within voices, large leaps in melody are penalized, and between voices, crossed voices (when a lower voice plays a higher tone than the next higher voice, for example, when the bass plays an **e** and the tenor just above plays a higher note, say **d**, in the same octave) are prohibited. Important to note is that in the implementation made for the article we're currently discussing, large scale harmonic progression is not implemented. The fitness function currently determines only 'the plausible movement between two adjacent chords' (p. 7).

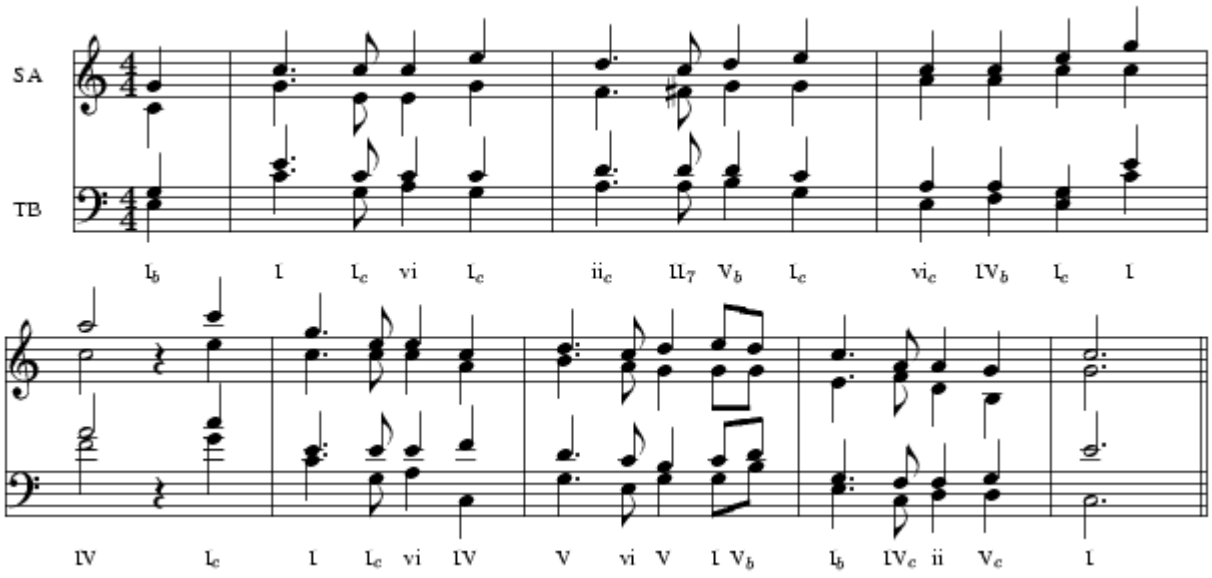
In combination with the used operators (which include crossover and mutation type operators), the generation/selection algorithm is able to find the most suitable solutions. The reproduction operators contain domain specific knowledge, and come in six flavours:

- *splice* makes a vertical crossover between two chromosomes;
- *perturb* allows the three voices that are not the melody (*i.e.* not the upper most voice) to transpose one or two semitones up or down (mutation);
- *swap* changes the position of the voices, so for example, the bass becomes the second highest voice;
- *rechord* mutates to a different chord type on a given position;
- *phrase start* mutates the beginning of a melodic phrase with the tonic on the first beat;
- *phrase end* mutates the end of a melodic phrase to end with a tonic chord.

Reviewing this brief discussion of the parameters of the system, we can conclude that the musical knowledge of the system is primarily contained in the fitness function. The other parameters merely define the search strategy. From the two output examples included on the next page, we can see that the system is good at getting the basics right (the generated notes are in harmony and don't earn a lot of penalties) – even, as the authors state, better than a student at harmonizing (p. 7). However, due to the shallow knowledge of the system, large scale harmonic progression is definitely not found in the generated output.

In the following sections we will discuss the results of both introduced systems in more detail. We will present a comparison study between the two different kinds of system. Also, we will recall the research question and the answer of both systems to it, and based on this discussion we will show a path for research that might be more interesting than the systems provided.

Auld Lang Syne



Joy to the World

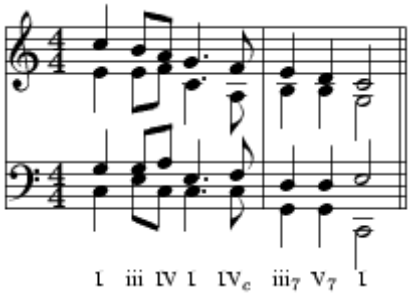


Figure 5: Example output of the GA by Wiggins *et al.*
The absence of explicit direction can be seen here by the chord numbers that are displayed under the staves. Recall that a 'normal' cadence like, for example, ***IV-V-I*** *is not found anywhere in this examples. However, each vertical note is in harmony with all other voices.*
(From: Wiggins et al., 1999)

3.3 Rule-Based and Evolutionary Harmonizing Systems Compared

Foremost: the results found in the previous sections are encouraging. The generated harmonies are harmonic indeed, and tend to sound musical. However, the results generated by CHORAL can be described as much more 'human' than those generated by the GA. Partly, this can be explained by the much more comprehensive knowledge encoded in the CHORAL system. The GA can not display any abilities that are not encoded, obviously. However, the authors of the GA nonetheless conclude that a conventional rule-based system is a more appropriate method for the harmonization task – due to the very nature of a genetic algorithm.

A GA uses random changes in a chromosome, which means that the search space is very sharply convoluted. So, when a local solution is found, it is very hard to find other solutions because many attributes of the chromosome would have to be changed at the same time. Even if a GA would be able to make so many mutations at the same time, the chances of this happening randomly at the same moment are very small. This means, there is no guarantee of an optimal solution. Additionally, GAs are a very attractive weak method search, but as more knowledge is encoded into the system, it requires lost of computation time.

Apart from the amount of *explicit* domain knowledge that is different if both systems, there is another difference: the amount of *implicit* domain knowledge is different as well. In the GA presented by Wiggins *et al.* (1999), there is no guidance at all in the process of harmonizing. To put it in another way: the *order* that is used in the CHORAL system to harmonize ensures it performs better than the GA. This can be illustrated by the research done by Phon-Amnuaisuk *et al.* (1999).

In the article mentioned, the authors construct a rule-based systems and a GA that each have the same amount of explicit domain knowledge. The GA works almost exactly as the one described in section 3.2, while the rule-based system solves the problem in the order cadence section, precedence section, body, introduction (from the end of the piece to the beginning) and uses chronological backtracking in its search process. Thus, the difference in the results displayed in Figure 6 is due to the fact that the rule-based system contains additional, implicit, knowledge. It uses a structured search mechanism, while the GA uses, per definition, an unstructured search mechanism.

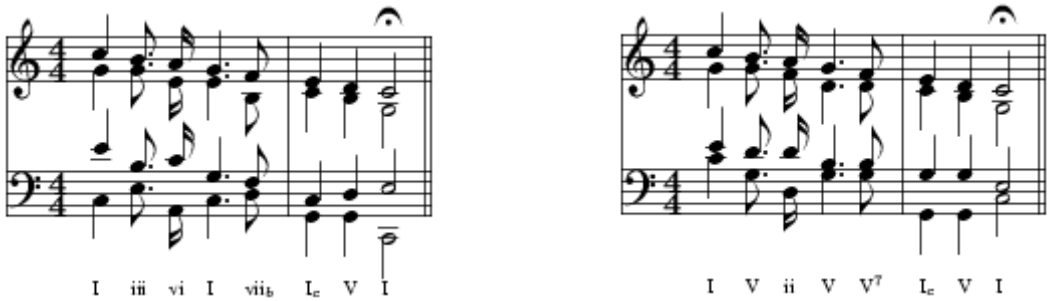


Figure 6: Harmonization of "Joy to the World' by a genetic algorithm (left) vs. a rule-based system (right) containing the same explicit knowledge.
(From: Phon-Amnuaisuk et al., 1999)

3.4 Discussion

We ended the last section with the conclusion that a GA uses an unstructured search mechanism. To put that in another way: it is extremely unlikely that a large-scale structure would arise from the limited knowledge and lack of guidance in the search algorithm itself. What we can conclude is that GAs can be surprisingly good at small tasks, constrained and using only simple parameters. But due to the lack of direction, of a musical 'plan', we can hardly argue that any composer would work in this same way. Therefore, this kind of programming does not provide us with an insight in how humans compose music.

When we look at the CHORAL system, we see that the complex set of rules the system possesses enables it to put more intent in its compositions and therefore makes them sound more human as a whole. However, the way in which the direction is created is by no means a human way, but rather a formal method to write a piece from start to finish[7]. What we know from hundreds of years of composing experience is that no composer works from beginning to end, and does not write all voices from top to bottom. This way is nevertheless a way in which humans would be able to compose, if forced, unlike the unstructured search the GA uses.

What we argue in this paper is that we need to analyze the way in which modern composers work better. In this way, we might be able to create an even better working model, since the quality of any system depends heavily on the amount of knowledge, whether implicitly or explicitly, it possesses. The CHORAL system uses a rigid knowledge base, which is neither easily understood nor adapted. When a system would be structured that contains both musical knowledge (like the CHORAL system) as well as information about musical processes, the system might presumably compose even better pieces.

For example, we know by analysis of Bach's work (see, for example, David & Mendel, 1966) that Bach used make particular difficult choices first. First, he chose cadences, and then he wrote most of the melody and bass parts before 'filling the piece up' with other parts[8]. At the same time, he altered the melodies in order to cope with changes in the 'fill-up' voices. In order to build a system that creates even more human sounding musical pieces, we need to make explicit the reasoning the system uses. This is described as the *explicitly structured knowledge paradigm* by Phon-Amnuaisuk *et al.* (1999).

In his paper about the same topic, Wiggins (1998) proposes the use of constraint logical programming (CLP) as a better solution for this task, and thereby argues that GAs are a kind of constraint mechanisms as well. To extend his line of thought, we would like to propose the use of GAs to solve *part* of the problem. Using a GA is much more effective than simply generating all possible 'solutions', because the generation process is more directed. By using the solutions that the GA comes up with in a larger knowledge base system, intend or musical progression can be created. For example, a rule-based system can generate a large progressive piece of music and direct the GA for individual parts – by altering the parameters and constraints of the GA. In this way, we get the best of both worlds: the ability to create music like a human composer would (explicit structural knowledge) and the power of a genetic algorithm (explicit musical knowledge).

7) Rather, from finish to start as described before.
8) In AI terms: Bach used a real-world least commitment strategy!

4 Using Computers to Generate Instrumental Solos

As we outlined in the introduction of this paper, composing music can be seen as the combination of harmonizing and generating a melody. Now we have discussed harmonizing extendedly, we will change our viewpoint from harmonizing music to generating instrumental solos. We will also leave the Classical age behind, and focus on modern 'light' music instead, because the domain of solos is typically the domain of Jazz music.

We will describe two systems for the generation of jazz solos, both with a genetic algorithm at their core. The first (and oldest) of the two is the GenJam system by Biles (1994) 'a genetic algorithm-based model of a novice jazz musician learning to improvise'. The other and most recent system is proposed by Papadopoulos *et al.* (1998). In the following section, we will outline the domain we're investigating. Thereafter, we'll look into the systems with a little more depth and see whether they're able to answer the research question posed in section 1.3.

4.1 Genetic Algorithms in Jazz Music

While composers search for the right chords to fit a melody or the right melody to fit a progression of chords, improvisers search for the right phrases to play over a chord progression. This is not necessarily a conventional melody but rather a set of musical 'lyrics' with a beginning and an end, with repetitive patterns or fragments. Of course, what is 'right' is arbitrary, but every musician has a set of patterns and melodic ornaments he or she likes, and this aesthetic sense should be able to direct the search process.

To model the aesthetics, we have different choices. Biles chose to create an interactive GA (IGA) in which a human listener will 'grade' the quality of the generated solos, while Papadopoulos *et al.* chose to model a fitness function[9]. This fitness function is based on how well the output fits the preferences of a given user. While we would not want to question the musical results of Biles' system (which are, in fact, quite interesting), we argue the chosen implementation with an IGA is too limited for further discussion in this paper. We're interested in the 'automatic' generation without interference of a human supervisor. Biles (1994) himself called it 'fitness bottleneck', a main drawback of most IGAs, as well as subjectivity (a user is likely to be influenced by previous listenings). Therefore, we will focus mainly on the GA by Papadopoulos *et al.* and merely highlight some differences with GenJam where appropriate.

Chord	Scale	Notes
Cmaj7	Major (avoid 4th)	C D E G A B
C7	Mixolydian (~ 4th)	C D E G A Bb
Cm7	Minor (avoid 6th)	C D Eb F G Bb
Cm7b5	Locrian (~ 2nd)	C Eb F Gb Ab Bb
Cdim	W/H Diminished	C D Eb F F# G# A B
C+	Lydian Augmented	C D E F# G# A B
C7+	Whole Tone	C D E F# G# A#
C7#11	Lydian Dominant	C D E F# G A Bb
C7#9	Altered Scale	C Db Eb E F# G# Bb
C7b9	H/W Diminished	C Db Eb E F# G A Bb
Cm7b9	Phrygian	C Db Eb F G A Bb
Cmaj7 #11	Lydian	C D E F# G A B

Table 7: Permitted notes in certain jazz keys.
(From: Biles 1994)

[9]) Their argument for not using IGAs is in fact the same as Wiggins *et al.* used in their GA for harmonizing, as is described in Note 5 on p. 10 of this paper.

Back to basics: in order to generate a solo, a chord progression must be given, and the generated melody must be in harmony with this progression. In jazz, harmonies are based on broader rules than in Classical music – a single dissonant tone is sometimes regarded as part of the unique style of a performer rather than as an error. There are special scales in which some notes are considered more plausible than others, see for example Table 7. This means we're dealing with very loose constraints, thus we're left with a large search space and lots of choices to be made for representation, operators and musical structures, not to mention the fitness function.

4.2 Knowledge for Playing a Solo

As opposed to Biles, who used integers to represent notes, Papadopoulos *et al.* argue for an extended knowledge representation in which a chromosome is a sequence of <*extended-degree, duration*> pairs with a chord progression that is defined as <*key, chord-type, duration*> triplets. This representation forms the core of this algorithm. Recall that other parameters of the GA include, for example, the fitness function and the used operators. We will discuss those parameters first, before discussing the output generated by the algorithm.

Local mutation operators transpose a random chromosome fragment of random length by, for example, a random number of degrees, or permute it in time. Also possible are changes of pitch while maintaining the rhythm, or the opposite, changes in rhythm while maintaining pitch; reversal in time; and sort in ascending or descending pitch. Apart from the large-scale change operators there's also a one-note mutation that changes one specific note in pitch. *Copy and operate mutation operators* copy a random fragment to a different position and possibly mutate it locally using one of the local mutations. *Restricted copy and operate mutation operators* also select a random fragment, but now the fragment is forced to start on the first or third beat of a bar (the rhythmically most important beats in the music) and with a size of only half a bar. *One-point* and *two-point crossover operators* (see section 3.2) are also implemented. Each of the operators is selected with a probability between 10% and 20%.

The *fitness function* judges the chromosomes according to musical criteria. We will not include all criteria in the present paper, but we'll highlight some because of the insight they can give in the working of this GA. For example, large intervals between notes are penalized because they contradict the Gestalt law of proximity (Gleitman *et al.* 1981). As well, repetitive patterns are rewarded because of the Gestalt law of similarity (Gleitman *et al.* 1981). When looking at melodies composed by humans, we can see there is always a large part of repetition in the melody to allow the listener for recognition. The authors note that the operators used also stimulate repetition, but nevertheless include pattern matching in the fitness function 'to allow for the case where motivic development occurred by chance' (p. 4). Other applications of the fitness function include the stimulation of scale notes on the first and third beat of a bar.

Improvising musicians often use a set of familiar musical and rhythmical patterns. Out of this set a selection is drawn and used in the solo. This is not currently encoded in the model of Papadopoulos *et al.*, but can be easily added due to the modular structure of the system. That would lead to the use of existing motifs from a real-world database, exactly what real-life musicians do – so, this would simulate human improvisation closely.

4.3 Discussion

Because of the used domain specific operators, the resulting output of the solo generating GA converged very quickly to high fitness. To get a feeling of theme development, the pattern matching operators were very beneficial in particular. The resulting output is amenable to improvement, but it definitely can be called musically plausible. The authors note that the weight attached to pattern matching operators is especially crucial: a low value makes the effect unnoticeable, whereas a high value results in a boring, repetitive solo.

The authors are not very enthusiastic about the results: they say the algorithm merely 'reduces the search space by ruling out solutions which are most probably musically unacceptable. It therefore increases the probability of a listenable output'. They cut and pasted two melodies together because it was suggested the two halves would fit together nicely, and in fact, they do (see Figure 8). In a formal sense, of course, this is an IGA since it 'requires' human interference. However, since all musical information was generated by a computer we'd like to argue that further refinements of the algorithm might create an output of this quality without human interference.

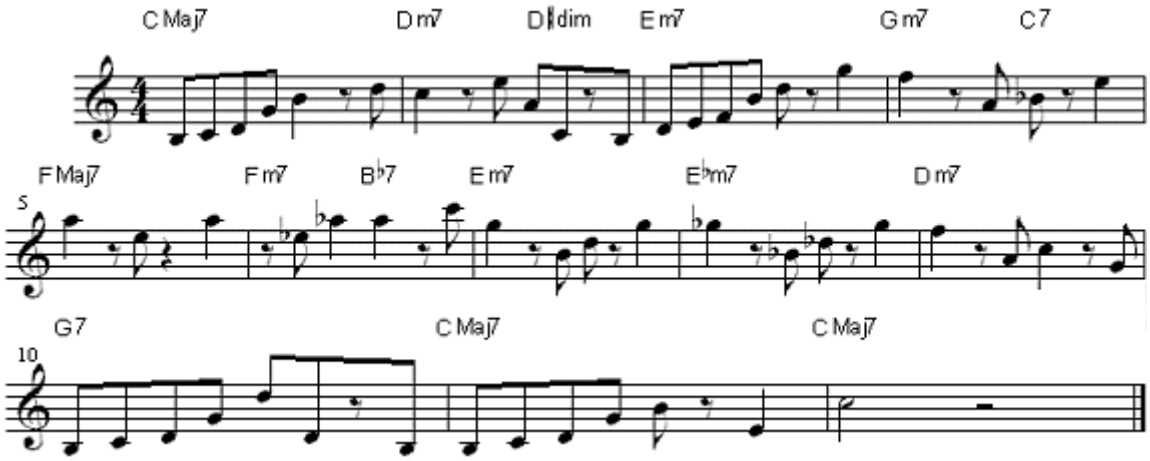


Figure 8: A combination of two GA generated melodies
(From: Papadopoulos et al., 1998, transcription: Wijnalda, 2002)

Papadopoulos *et al.* propose several extensions to the presented GA. The most encompassing and ambitious would be to include the building of tension: this is often said to be the main reason of meaning in music. The authors believe 'this would be a big step forward in overcoming the lack of reasoning which the system exhibits' (p. 9). We can introduce the line of thought stated in section 3.4 as well: when combining the present GA with an overarching rule-based system, we might formalize the cut and paste that was done by humans during the research of Papadopoulos *et al.*.

Recapitulating, as well as to answer the research question for this system, we can say it is able to compose music that sounds human.

5 Conclusion

In this paper, we showed part of the domain of music generation, introduced harmonization features and problems, and highlighted two harmonizing systems as well as two jazz solo generating systems. What we tried to do, as well, is illustrate and answer our research question: are computers able to perform this task, and are they able do this is in a 'human' way? We divided the question in two parts in order to answer the question more properly and now's the time to bring back both parts together.

Although we answered the main part of our research question in the respective discussion sections (3.4 and 4.3), there's still some remarks we need to make. We can say from the research discussed before, that computers are indeed able to compose interesting music, but that they're not yet doing that in the way human composers work. This applies for both the harmonizing as for the solo generating systems, although improvising a solo is something that is based on intuition rather than contemplation and is thus more suitable for the GA approach that generates without real intend. An improvising musician starts somewhere and knows that in a few bars from now, he or she needs to get 'back' at the same point. This disqualifies the use of rule-based systems for solo generation: they're simply too structured.

For large-scale progression structures, a rule-based system is much better suited than a GA. A GA system uses unstructured search that generates a lot of musical information in which it is not likely something unlikely as a large-scale progression will emerge. This disqualifies GAs for this particular use, again due to the nature of the problem that is to be solved.

In summary, what we can conclude is that the more knowledge (both implicitly and explicitly) the system possesses, the better its output is. In order to structure the musical structure knowledge better, we argued that we should explore the ways in which composers created their music in the past 2000 years and model the structural knowledge in that way. As for domain knowledge, it is inevitable that more and more musical experience will be encoded and used in systems such as those discussed. In the end, a combination has to be found between GAs and rule-based systems in order to cope with the combined problem of creating structures with large-scale musical development, and parts of that structure that have to be melodic (a kind of 'solos') with intend but without predefined structure.

So, if we direct our research towards this joint-venture between GAs and rule-based systems we might get more and more sophisticated results. And with the speed of present day technology advancement, artificial music generation might even rival Bach some day…

References

Bach, J.S. (1722). *The Well-Tempered Klavier (Das wohltemperierte Klavier)*. London, UK: Associated Board of the Royal Schools of Music, 1951.

Biles, J.A. (1994). GenJam: A genetic algorithm for generating jazz solos. In: *ICMC Proceedings 1994*, pages 131-137. San Francisco, CA: The Computer Music Association.

Boulez, P. (1963). *Penser la musique aujourd'hui.* Mayenne, France: Gonthier.

Bratko, I. (1990). *Prolog: Programming for Artificial Intelligence* (2nd edition). Reading, MA: Addison-Wesley.

Campbell, M. and C. Greated (1988). *The Musician's Guide to Acoustics.* New York, NY: Schirmer.

Darwin, C. (1859). *On the Origin of Species.* London, UK: John Murray.

David, H.T. and A. Mendel (1966). *The Bach Reader*. New York, NY: W.W. Norman.

Ebcioglu, K. (1993). An expert system for harmonizing four-part chorales. In: S.M. Schwanauerand D.A. Levitt (eds.), *Machine Models of Music*, pages 385-402. Cambridge, MA: MIT Press.

Gleitman, H., A.J. Fridlund & D. Reisberg (1981). *Psychology.* 5th edition (1999). New York, NY: W.W. Norton & Company.

Harmelen, F. van (1989). A classification of meta-level architecture. In: Jackson, P.; H. Reichgelt and F. van Harmelen (eds.), *Logic-based knowledge representation*, pages 13-36. Cambridge, MA: MIT Press.

Hofstadter, D.R. (1979). *Gödel, Escher, Bach: an Eternal Golden Braid.* New York, NY: Random House.

Papadopoulos, G. and G. Wiggins (1998). A genetic algorithm for the generation of jazz melodies. In: *Proceedings of STeP 98.* Jyväskylä, Finland: STeP.

Phon-Amnuaisuk, S. and G. Wiggins (1999). The four-part harmonization problem: a comparison between genetic algorithms and a rule-based system. In: *Proceedings of AISB 99.* Edinburgh, Scotland: University of Edinburgh.

Schenker, H. (1979). *Free Composition (Der freie Satz).* Translated by E. Oster (ed.). New York, NY: Longman Press.

Schönberg, A. (1984). Edited by L. Stein. *Style and Idea: selected writings of Arnold Schönberg.* Translated from German by L. Black. Berkeley, CA: University of California Press.

Wiggins, G. (1998). The use of constraint systems for musical composition. In: *Proceedings of the ECAI 98 Workshop on Constraints and Artistic Applications.* Brighton, UK: ECAI.

Wiggins, G.; G. Papadopoulos, S. Phon-Amnuaisuk and A. Tuson (1999). Evolutionary methods for musical composition. In: *International Journal of Computing Anticipatory Systems* 1999.