Algorithmic Music Composition

MUS-15

Jan Dreier

July 6, 2015

1 Introduction

The goal of algorithmic music composition is to automate the process of creating music. One wants to create pleasant music without or with minimal human interaction.

This is a quite challenging task and currently, the best algorithms are far inferior to human composers. But over the years, progress has been made and this report shall present some approaches and common techniques used to address this problem.

In particular, we will discuss use of genetic algorithms for music generation. This technique starts with a set of simple musical compositions and iteratively improves them and makes them more complex. This is a common technique for algorithmic music generation [4] [5].

Section 2 shall serve as a short historic overview over the subject. In Section 3 we will discuss common approaches. Then, in Section 4 and 5 we will discuss a paper by Donnelly and Sheppard [3] in which they present an algorithm which composes four-part harmonies using genetic algorithms.

2 History

Algorithmic composition was already popular before the rise of the computer. In the Renaissance, dice based composition games were popular. In such game, a composition was stitched together from different fragments, where selection and order of these fragments was decided by the roll of a dice.

Then, in 1957 Hiller and Isaacson [8] were the first to make a computer generate a musical composition. They used a computer of the University of Illinois to generate a composition for a string quartet.

Another milestone happened in 1991, when Horner and Goldberg [1] introduced genetic algorithms into the domain of algorithmic composition. Until today, genetic algorithms remain popular in this domain.

3 Music Composition As Search Problem

We can see the task of algorithmic music composition as a search problem. The search space is the set of all possible compositions and the goal is to search through these compositions and return a composition which sounds good. This approach raises various challenges.

- First, we need a representation of the search space: Shall we represent music as audio-files, as sheet-music or maybe as some hierarchical data-structure?
- No matter how we represent our music, the search space of all possible compositions is to big to search exhaustively, so we need some heuristic to guide our search.
- Furthermore, we need to evaluate if a composition sounds good, which is a highly subjective problem and hard to quantify.

The aim of this report is to present common techniques to solve these problems. We will discuss

how to encode compositions, how to use genetic algorithms to guide the optimization process and how to use musical rules to evaluate the quality of a musical piece.

Music Representation

One option would be to represent compositions as audio-files, such as .wav or .mp3 files. This is the most precise description of a composition, as it leaves no room for interpretation like sheet-music does. But it has the disadvantage of very high dimensionality, which is bad for searching. Furthermore, it is hard to edit, e.g., it is a non-trivial task to shift the pitch of a single note up or down in an audio-file. Thus, in this report we prefer a representation which only stores pitch and duration of each note, which is more compact and easier to manipulate. So the output of the composition algorithm will be note information such as sheet-music or MIDI-files, which can then be played by human artists or software.

Searching Via Genetic Algorithms

Now we will discuss the use of genetic algorithms to guide the search for pleasant compositions. Genetic algorithms are a general purpose search heuristic inspired by the evolutionary process of species and natural selection. They are applied in various domains and can conquer even large search spaces efficiently. They have first been used for algorithmic music composition in 1991 [1] [2] and has since then become common tool for this task.

A genetic algorithm starts with a set of simple solutions to the search problem, the first generation. Then, through an iterative process, it computes further generations. Each generation is a set of possible solutions and the quality and complexity of these solutions shall gradually improve with each new generation. In the case of algorithmic music generation these solutions are compositions.

In order to push towards better solutions, only the fittest members of the current generation should donate to the gene-pool of the next generation. Thus, once a new generation is created, we evaluate the fitness of each member. How to evaluate the fitness is highly domain-dependent. It is common to define a fitness function which maps elements of the search space to real numbers and is maximal for optimal solutions.

After the evaluation of each member's fitness comes the mutation and crossover phase. In this phase we try to get better solutions to our problem by randomly modifying an already good solution or by merging two good solutions into one.

The terms are taken from biology: Modifying a single solution is called mutation and joining two solutions is called crossover. These rules are applied multiple times on random solutions of the current generation to yield the next generation. In order to improve the global fitness, we choose solutions with a high fitness value with a higher probably than those with low fitness.

We iterate this process multiple times. By taking only the fittest members for the mutation and crossover operations the global fitness should increase with each generation. In the end, we may stop if the fittest member of the current generation is good enough or does not improve anymore, or if some other measure of convergence is fulfilled. The fittest member of the last generation then is the final output. A flow chart representation of this process can be seen in Figure 1.

Fitness Evaluation

If we want to use genetic algorithms for composing music, we need a way to evaluate the quality or fitness of a composition. We will discuss three different approaches to do that.

The simplest method of quality evaluation is to simply as a human. Present a song and tell him or her to rate it on a scale from 1 to 10, as it is done in [6]. This is of course very easy to implement and it may be particularly suited to find a composition which one particular person enjoys. But the overwhelming drawback of this technique is that it is quite time-consuming.

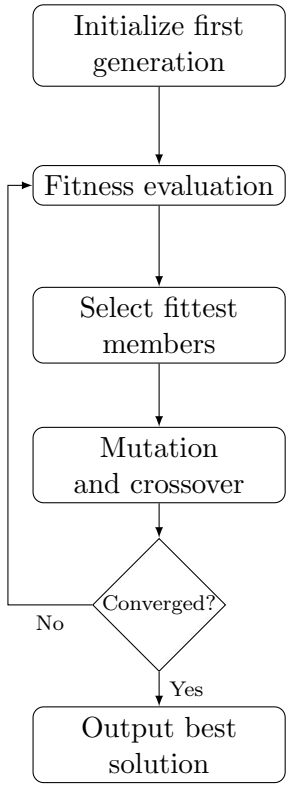

Figure 1: Flow chart of a genetic algorithm

If we were to run a genetic algorithm with 10 compositions per generation for 10 generations (these are very modest numbers), a user would already have to listen to and rate 100 compositions. Since our goal was to automate the process music composition as much as possible, we will no further discuss this approach.

On the other hand, one could use machine learning techniques for fitness evaluation [7]. The idea is to train a classifier such as a neural network or a support vector machine by giving it a collection of compositions it shall consider good and a collection it shall consider bad. After training, when the classifier is given a new composition, it decides which of those two collections the composition is more similar to. The composition is then assigned a high fitness-value if the classifier says that it is similar to the good ones and a low fitness, if it is similar to the bad ones. This technique seems to be quite effective but is beyond the scope of this report.

In this report we will discuss rule based fitness evaluation. Here, we define a strict set of musical rules which a composition needs to obey in order to be considered good. The better a song sticks to these rules, the higher its fitness will be rated. This approach has a strong music theoretical background, as it is founded on the observations the great composers have made during the last centuries. These rules are common knowledge to anyone with a background in classical music composition. In the next section we will discuss a rating algorithm which evaluates a composition's fitness by checking for many of these musical rules how well they are followed.

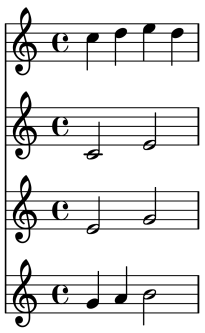

Figure 2: Sheet-representation of a short composition. The four lines play in parallel.

4 Evolving Four Part Harmony Using Genetic Algorithms

In this section we shall investigate how music representation, genetic algorithms and fitness evaluation are done in the paper "Evolving Four Part Harmony Using Genetic Algorithms" by Donnelly and Sheppard [3]. The aim of this paper is to develop an algorithm which composes classical pieces, consisting of four parallel voices. They use genetic programming to grow an initial single chord into a larger composition and maximize a fitness function based on musical rules.

Musical Representation

In [3], compositions consist of four parts that are played in parallel. Each part consists of a list of (pitch, duration) tuples, which are played sequentially. The pitches are restricted to the C-major key and notes may have a duration between $1/8$ and $8/8$. An example can be seen in Figure 2.

Genetic Algorithms

The first generation of the genetic algorithm is initialized with simple C-major chords. Then, with each new generation, these chords are grown into more complex structures. Each generation contains 1000 compositions.

During the optimization phase of the genetic algorithm the following operators are applied to the current generation: Mutation, crossover, inversion and duplication.

(a) input (b) output

Figure 3: Example of the pitch shift mutation rule: The third note on the left was randomly selected to be shifted upwards by a 5th.

The following process yields a composition of the next generation: At first, one out of the four operators is chosen randomly. Then, compositions of the current generation are chosen as input for the operator. Compositions with a high fitness score are picked with a higher probably. Mutation, inversion and duplication take one input and crossover takes two inputs.

For the mutation operator, there are a total of 15 possible mutation rules. One of them is chosen randomly and applied. These mutation rules include shifting the pitch of a randomly chosen note, duplicating a random note, changing the duration of a random note or inserting a random note. See Figure 3 for an example.

The crossover operator takes two compositions as input and splits them both in two parts. Then it creates a new composition by appending the first half of the first composition with the second half of the second composition. Inversion and duplication operate on a single composition by duplicating or inverting a short random interval of the composition. These operators shall iteratively evolve the initial generation of simple C-major chords into more complex compositions.

Rule Based Fitness

The genetic algorithm needs to be guided by a fitness function which evaluates the quality of each new generation. We shall now discuss how rules from music theory are phrased and implemented by Donnelly and Sheppard [3] to create such a fitness function. For each musical rule they construct a fitness function and define the total fitness as a weighted sum of these individ-

ual fitness-values.

At first, let us take a look at some rules. There are a total of 15 rules defined in the paper, but we will only discuss three of them.

The first rule says, that between two consecutive notes of the same part, there should be no jump larger than the interval of a 9th. This rule shall enforce that the melody flows smoothly and does not jump all over the place.

Another rule orders the four parts from low to high and states, that whenever two parts play a note at the same time, the higher part needs to play the higher pitch. This ordering is desired, because if two melodies play at the same pitch it becomes hard to distinguish between them and the song starts to sound muddy.

A third rule says, that whenever two notes sound at the same time they should form an harmonic interval. We know from music theory that different intervals convey different moods. Disharmonic intervals like a 2nds or 7ths sound odd or create tension and harmonic intervals, such as 3rds or 6ths sound well. So the former shall usually be avoided.

Now that we have defined some rules, we can define a fitness value for each rule:

$$\text{fitness}(\text{rule}_i) = \frac{n_i - v_i}{n_i}$$

where n_i is the number of locations where rule_i can be violated and v_i is the number of locations where it actually is violated. So if the rule is never violated the fitness becomes 1 and if it violated whenever possible the fitness becomes 0.

For example, for the rule limiting the magnitude of intervals between consecutive notes n_i equals the number of note transitions in all four parts and v_i equals the number of transitions, that are larger than a 9th.

Additionally, each rule_i has a certain handchosen weight ω_i. The total fitness value of a composition then is a weighted sum over all partial fitness rules:

$$\text{total fitness} = \sum_i \omega_i \cdot \text{fitness}(\text{rule}_i)$$

This way, important rules can be given a higher weight. In summary, a composition will archive a high fitness value if it violates as few rules as possible.

5 Evaluation

The algorithm we just discussed creates four-part harmonies by iteratively growing simple chords using mutation and crossover operations. After each step, a fitness value is assigned to each composition by checking how well it obeys to different musical rules.

To the best of my knowledge, the authors have not provided many compositions of their algorithm and did not do a rigorous evaluation. They say that after 100 runs of their algorithm, the compositions yield an average fitness score of 2.668. The maximal possible score is 3. The authors provided a single, 15-second composition with the exceptional score of 2.94. This composition sounds quite pleasant to my ear. Locally, it contains many of the desired properties of a good composition, such as harmonic intervals or contrary motion between parts. But it lacks a global structure.

To me, it seems like this technique works quite well if the musical domain is restrictive enough that it can be formalized by simple rules. However, I see the disadvantage that for each new genre the user of the algorithm has to analyze this genre and formulate a set of rules. So algorithmic composition approaches are still lacking the skill and creativity of human composers.

References

[1] Horner, Andrew, and David E. Goldberg. "Genetic algorithms and computer-assisted music composition." (1991): 337-441.

[2] Gibson, P. M., and J. A. Byrne. "Neurogen, musical composition using genetic algorithms and cooperating neural networks." Artificial Neural Networks, 1991., Second International Conference on. IET, 1991.

[3] Donnelly, Patrick, and John Sheppard. "Evolving four-part harmony using genetic algorithms." Applications of Evolutionary Computation. Springer Berlin Heidelberg, 2011. 273-282.

[4] Moroni, Artemis, et al. "Vox populi: An interactive evolutionary system for algorithmic music composition." Leonardo Music Journal 10 (2000): 49-54.

[5] Gartland-Jones, Andrew. "Musicblox: a real-time algorithmic composition system incorporating a distributed interactive genetic algorithm." Applications of Evolutionary Computing. Springer Berlin Heidelberg, 2003. 490-501.

[6] Johanson, Brad, and Riccardo Poli. GP-music: An interactive genetic programming system for music generation with automated fitness raters. University of Birmingham, Cognitive Science Research Centre, 1998.

[7] Manaris, Bill, et al. "A corpus-based hybrid approach to music analysis and composition." Proceedings of the National Conference on Artificial Intelligence. Vol. 22. No. 1. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[8] Hiller, Lejaren Arthur, and Leonard M. Isaacson. Experimental Music; Composition with an electronic computer. Greenwood Publishing Group Inc., 1979.