Evolving Four-Part Harmony Using Genetic Algorithms

Patrick Donnelly and John Sheppard

Department of Computer Science, Montana State University,
Bozeman MT 59715
{patrick.donnelly2,john.sheppard}@cs.montana.edu

Abstract. This paper presents a genetic algorithm that evolves a four-part musical composition–melodically, harmonically, and rhythmically. Unlike similar attempts in the literature, our composition evolves from a single musical chord without human intervention or initial musical material. The mutation rules and fitness evaluation are based on common rules from music theory. The genetic operators and individual mutation rules are selected from probability distributions that evolve alongside the musical material.

Keywords: Genetic Algorithm, Evolutionary Programming, Melody, Harmony, Rhythm, Music Composition.

1 Introduction

Algorithmic composition is the process of creating music using either determinate or indeterminate algorithmic processes, or some combination thereof. While composing, musical composers use many simple rules during the creative process. Unlike stochastic composition techniques, algorithmic methods that encode knowledge from rules of voice leading and counterpoint can better approximate musically desirable solutions.

The research presented here attempts to compose four-part musical harmony with a GA without user intervention or initial musical material. Each musical piece begins as a single musical chord and is expanded using genetic operations. Unlike previous music experiments with GAs that focus on one aspect of music or require initial musical material, such as a melody to harmonize, our system evolves the entire composition, including the melodies, harmonies, and rhythms, using genetic operations. Our system features a variable length chromosome that represents a four-voice composition and uses the four basic genetic operators of mutation, duplication, inversion, and crossover. The mutation operator and the fitness function evaluation are based on music theory rules.

2 Related Work

Algorithmic composition has long been of interest to computer scientists and musicians alike. Researchers have used a variety of approaches toward algorithmic

© Springer-Verlag Berlin Heidelberg 2011

composition including rule-based systems [2], musical grammars, artificial neural networks, and Markov chains (see [11] for a review).

Previously, genetic algorithms have been applied in several areas of music composition. However, given the broad search space afforded by GAs and the multi-dimensional complexity of music itself, most studies have constrained themselves to focus on one dimension of music at a time, such as evolving rhythms, melodies, or harmonies, but we have found no studies that evolve all aspects together. Genetic algorithms have been used to compose single line melodies [3], creative jazz melodies based on a given chord progression [1], unpitched rhythms [5], Baroque-style harmony based on a given melody [10] [12] and figured bass [8], and the composition of microtonal music [6]. Compositional rules based on music theory have been used to control the fitness function [9] as well as the mutation operator [13] in the evolution of single line melodies.

As discussed in [11], the efficacy of any algorithmic composition method depends heavily on the underlying representation of musical knowledge. Most approaches to composition using GAs apply mutation and other operators stochastically, relying on selection and the fitness score to eliminate the weak individuals from the population. Our approach uses an probabilistic approach to operator selection, but unlike previous approaches, we allow these probabilities to evolve alongside the musical material. Therefore the individuals that succeed to subsequent generations contain information about which musical rules and genetic operators contributed to their higher fitness scores, thus allowing a more directed and efficient traversal of the broad evolutionary search space.

3 Genetic Algorithm

3.1 Genome Representation

Individual in the population consist of a variable length chromosome that represents a single musical composition. A chromosome contains four parts, each corresponding to a musical line. Every individual part consists of a list of tuples containing a pitch and duration value. Musical pitch is represented by a standard MIDI value, ranging $[0, 127]$. Duration is represented as an integer from $[1, 8]$ that corresponds to the length in semiquavers (eighth notes). A duration value of one indicates a single eighth note while a duration of eight represents a single whole note (Fig. 1). The musical representation contains no time signature, allowing the composition to grow and shrink from insertions and deletions anywhere in the piece. The chromosome also assumes a key of C-Major and does not contain any key modulations. The final output can be transposed to another musical key.

Each individual in the initial population consists of only a single whole note chord. The bass line always contains the tonic note (C) while each of the upper three parts contains a single note in the tonic chord selected randomly with replacement from the C-Major triad (C, E, or G). Each musical part will grow from this single initial note using mutation and the other genetic operators.

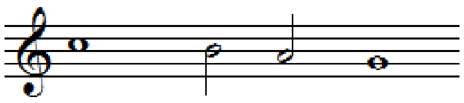


Fig. 1. Example chromosome for a single part represented as a list of pitch-duration tuples: $\{(72, 8); (71, 4); (69, 4); (67, 8)\}$

3.2 Operators

Operator Probability. In addition to the musical information, each chromosome also contains a set of probabilities used in the selection of the genetic operators. Each genetic operator *op* has an individual probability drawn from a total probability mass.

$$\mathbf{P}(op) = \begin{cases} P(mutation) \\ P(crossover) \\ P(duplication) \\ P(inversion) \end{cases} \in [0, 1]$$

$\mathbf{P}(op)$ represents the distribution that operator *op* will be selected. On each generation and for each chromosome, a single operator is probabilistically selected and applied. The probabilities of the genetic operators are themselves mutated in every generation. A probability corresponding to a single operator is randomly selected from the distribution, and a random value from $[-0.1, 0.1]$ is added to the probability. The distribution is then normalized to maintain a sum of one. The initial operator probability distribution is shown in Table 1.

Like the operator probabilities, individual mutation rules are also selected probabilistically. On each generation, if the mutation operator is selected, a single mutation rule m_i is selected probabilistically and applied to the chromosome. These individual mutation rules are drawn from a separate probability distribution, where

$$\sum_i P(m_i | op = mutation) = 1$$

Thus, the probability of selecting a specific mutation operator is given by $P(m_i, mutation) = P(m_i | mutation) \times P(mutation)$. The probabilities of the mutation rules are randomly perturbed each generation in the same manner as the operator probabilities. The probability distributions for the mutation rule selection are initialized uniformly with an equal fraction of the total distribution.

Selection Method. To create a new generation, the fitness score of each chromosome is calculated. Truncation elimination is applied to remove the weakest ten percent from the population. To preserve the most fit chromosomes, an elitist selection adds the top ten percent of the population to the successive generation without any change.

Table 1. Initial Operator Probabilities

Operator	Initial Probability
Mutation	0.4
Crossover	0.2
Inversion	0.2
Duplication	0.2

The remainder of the population is generated by altering chromosomes using one of the four genetic operators. A single chromosome from the previous generation is selected by tournament selection with a tournament size of two. This selected chromosome is then altered by one of the genetic operators probabilistically and added to the subsequent generation.

Mutation. If the mutation operator is selected, a single mutation rule is selected according to the rules probability. For each part, a note or pair of consecutive notes is selected at random and the mutation rule is applied. The mutation rules are described below.

1. **Repeat** – the selected note is repeated with the same pitch and duration value.
2. **Split** – the selected note is split into two notes. Each new note has the same pitch as the original, but half the original duration.
3. **Arpeggiate** – the selected note is split into two notes of equal duration. The first note retains the original pitch value while the second note is transposed randomly to pitch a third or fifth above the first note.
4. **Leap** – the pitch of the selected note is randomly changed to another pitch in the C-Major scale within the musical range of the part.
5. **Upper Neighbor** – for a pair of consecutive notes with the same pitch, the pitch of the second note is transposed one diatonic scale step higher than the original.
6. **Lower Neighbor** – for a pair of consecutive notes with the same pitch, the pitch of the second note is transposed one diatonic scale step lower than the original.
7. **Anticipation** – the selected note is split into two notes, each containing the original pitch, but an unequal duration. The duration of the first note is shorter than the second note by the ratio of 1:3. For example, if the original note is a half note, the new notes will be an eighth and a dotted-quarter note.
8. **Delay** – the selected note is split into two notes, each containing the original pitch, but of unequal duration. The duration of the first note is longer than the second note by the ratio of 3:1.
9. **Passing Tone(s)** – for a pair of selected consecutive notes, new notes are added in between that connect the two notes by stepwise motion. If the second note is lower than the first, the new notes connect in downward scalar motion. If the second is higher than and the first, the new notes connect the two original notes by upward scalar motion.

10. **Delete Note** – the selected note is removed from the part.
11. **Merge Notes** – for a pair of consecutive notes with the same pitch, the two notes are combined into a single note with a duration that is the sum of the two original notes, or eight, whichever is smaller.

Crossover. When the crossover operator is chosen, a pair of chromosomes is selected by tournament selection and recombined using single-point crossover to create two new chromosomes. For any selected pair of chromosomes C^1 and C^2 with lengths m and n respectively, cutpoints i and j are selected at random. A cutpoint is a duration point selected randomly anywhere between the start and the end of a piece. It may fall on a boundary between two notes, or bisect an individual note, in which case the note is split into two notes at this boundary. Two new chromosomes $C_3 = C^1_1..C^1_i C^2_{j+1}..C^2_n$ and $C_4 = C^2_1..C^2_j C^1_{i+1}..C^1_m$ are created and added to the next generation (Fig. 2(a)). The crossover operator is applied to each of the parts using the same cutpoints. The operator probabilities and the mutation rule probabilities for the new chromosome are the average of the respective probabilities of the original two chromosomes.

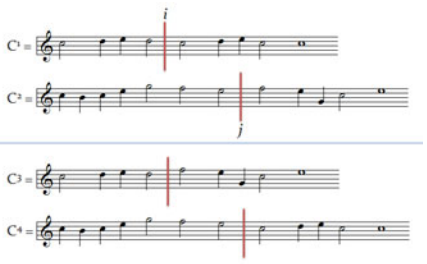
Duplication. A single chromosome is selected by tournament selection and a portion of the chromosome is repeated. An initial cutpoint i is selected at random and a second cutpoint j is selected at random, such that $j > i$ and $j < i + 8$. The duplication operator restricts repetition to at most the duration of a whole note so that the chromosome does not grow in length disproportionately with other individuals in the population. A new chromosome containing a repetition of notes from i to j is created and added to the next generation (Fig. 2(b)).

Inversion. A single chromosome C is selected by tournament selection and a subsequence of the chromosome is reversed, similar to John Holland's original inversion operator [4]. An initial cutpoint i is selected at random and a second cutpoint j is selected at random, such that $j > i$. The order of the notes between i and j are then reversed. The new chromosome containing the retrograde musical subsequence is added to the next generation (Fig. 2(c)).

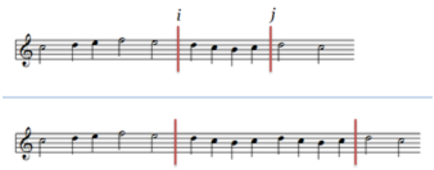
3.3 Fitness Function

The fitness function evaluates the quality of various aspects of the piece of music according to several common rules of musical composition and voice leading [7]. As in [9], the fitness of the entire chromosome is a weighted sum of R individual fitness rules. In order to balance the relative importance of some rules over others, each rule has a weight associated with it. Each rule is analyzed separately. An individual fitness rule r_i produces a $score(r_i) \in [0, 1]$, and w_i indicates its corresponding weight in the total fitness score.

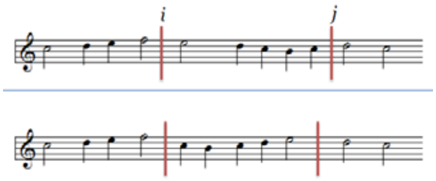
$$fitness = \sum_{i}^{R} w_i \times score(r_i)$$



(a) Example of the crossover operation. Above, original chromosomes C^1 and C^2 with cutpoints i and j, respectively. Below, new chromosomes C^3 and C^4 resulting from crossover of C^1 and C^2.



(b)) Example of the duplication operation. Above, the original chromosome with cutpoints i and j. Below, the new chromosome with duplicated subsequence.



(c) Example of the inversion operator. Above, the original chromosome with cutpoints i and j. Below, the new chromosome with the inverted subsequence.

Fig. 2. Example of the crossover, duplication, and inversion operators for a single musical line

An individual fitness rule indicates how well the composition conforms to the individual rule alone. We analyze each opportunity n that the rule might be analyzed and each violation v of the rule is counted.

$$score(r_i) = \frac{n - v}{n}$$

For rules that involve only a single note, such as Part Range, n corresponds to a count of the notes in the composition. However, for rules that must analyze two notes, such as Leap Height, n corresponds to the sum of the number notes in each part minus one. A score of one indicates there were no rule violations while a value of zero indicates the rule was violated at every opportunity.

In our present experiments, we naïvely weight the rules according to the type of rule: melody, harmony, or rhythm. The sum of the rules within a type are normalized to value between $[0, 1]$, and the total fitness score of the chromosome is the sum of these three types, a value between $[0, 3]$. In future work, we wish to experimentally tune the weights as in [9] by empirically testing each fitness rule individually on compositions from the literature and weighting each rule according to its potential for fitness score gains.

Melodic Rules. The following rules examine pitch within a single part:

1. **Opening Chord** – the first chord should be a tonic chord.

2. **Closing Chord** – the final chord should be a tonic chord.
3. **Final Cadence** – the last two chords should form an authentic cadence in which the bass line should close with movement from a fifth above or a fourth below to the tonic note and the other three parts should close by stepwise motion to the final note.
4. **Part Range** – the difference between the lowest note and the highest note in each part should be no more than an interval of a 13th (octave and a fifth).
5. **Leap Height** – individual leaps should be no more than an interval of a 9th (octave and a second).
6. **Leap Resolution** – leaps more than an interval of a major sixth should resolve in the opposite direction by stepwise motion.
7. **Voice Crossing** – a lower part should not contain a pitch higher than an upper part and an upper part should not contain pitches below a lower part.
8. **Voice Range** – each part should not contain pitches outside its range (Fig. 3).
9. **Repeating Pitch** – no part should repeat the same pitch more than three times consecutively.
10. **Stepwise Motion** – at least half of all melodic intervals in each part should be stepwise motion. This helps forms a coherent melodic line by penalizing excessive leapiness.

Harmonic Rules. The following rules examine pitch across multiple parts:

1. **Parallel Octaves** – no two parts should contain motion that forms parallel octaves (two parts one octave apart moving to new notes also an octave apart).
2. **Parallel Fifths** – no two parts should contain motion that forms parallel fifths.
3. **Vertical Consonance** – no two parts should form a dissonant harmonic interval (seconds, fourths, or sevenths).

Rhythm Rules. The following rules examine duration within a single part:

1. **Rhythmic Variation** – the piece should contain instances of each note duration type, from the eighth note to the whole note.
2. **Note Repetition** – the piece should not repeat a note of the same duration more than three times consecutively.

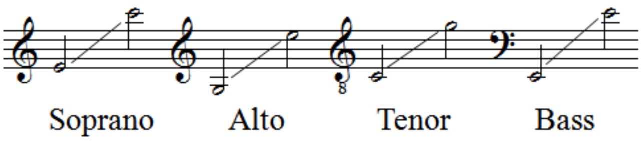


Fig. 3. Musical range of each part

4 Results

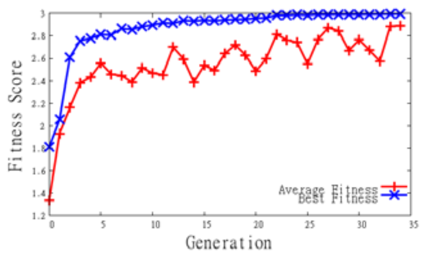
In our experiments, we use a population size of 1000. On each generation we calculate the average fitness of the entire population. The genetic algorithm is halted when the average fitness of the entire population converges to a value no more than 0.01 from the previous generations average fitness. The chromosome with the highest fitness score is saved as a MIDI file. In an experimental run of 100 trials, the average number of generations was 11.06 and the average fitness score of the final generation was 2.668 out of a maximal possible fitness score of 3.0. Fig. 4(a) shows the average and best fitness scores for a single experimental run of 34 generations.

We examined the evolution of the individual operator probabilities throughout the evolutionary process. In each generation, we average each of the four operator probabilities over the entire population. At the final generation, the probability of the crossover operator dominated the distribution, accounting for approximately 97% of the probability mass, leaving only trivial probabilities of the other three operators. In a separate experiment, we examined adding a fifth operator that consisted of no operation, allowing an individual to move to the next generation unchanged. However, even with this additional operator the crossover probability continued to dominate. This annealed preference of the crossover operator demonstrates that crossover presents the best opportunity for fitness score gains (Fig. 4(b)).

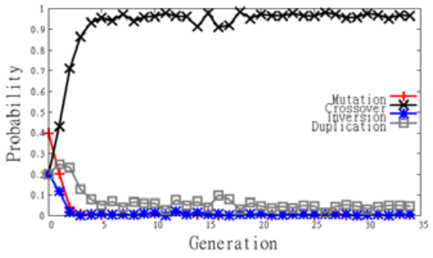
Our preliminary results demonstrate that it possible to evolve four-part musical compositions entirely from an initial single chord (Fig. 5). The resulting compositions show a number of desirable musical qualities, such as preference for leaps in the outer parts, preference for stepwise motion in the inner parts, contrary motion between parts, and singable parts in the proper musical ranges. While our compositions are musical, they suffer from several limitations. Since we allow each musical line to grow or shrink independently, analyzing the harmony only in the fitness evaluation, the parts tend to move independently and the composition often lacks coherence between parts. Additionally, our results feature a dominance of C-Major chords, although Fig. 5 indicates there are some exceptions. Although the mutation rules do change individual pitches, the resulting harmony rule score tends to penalize the composition before pitches in the other parts have a chance to mutate and form a new chord. These limitations can be overcome with a better tuning of the fitness score weights and the addition of new mutation rules.

5 Conclusion and Future Work

In this paper we present a method to evolve four-part musical compositions using genetic algorithms. While there have been other attempts in the literature to compose music using GAs, these works generally constrain themselves to one aspect of music alone. Our work examined the feasibility of using an evolutionary technique to create entire music compositions, including the melodic lines,



(a) Average fitness score and the best fitness score for each generation.



(b) Operator probabilities averaged over entire population.

Fig. 4. Experimental run of 34 generations

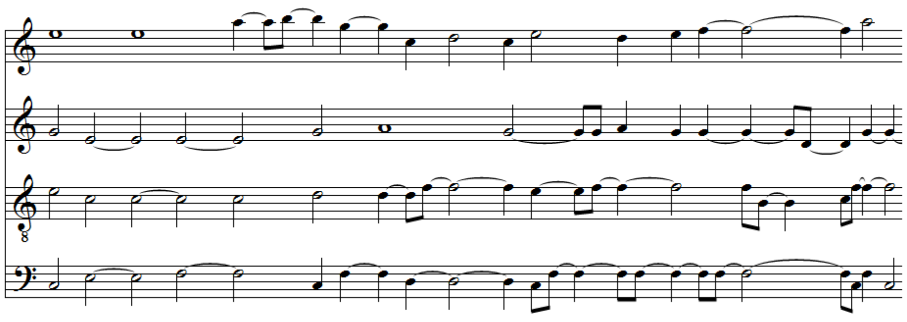


Fig. 5. Excerpt of a composition with a fitness score of 2.94

rhythms, and harmonies, from a single musical chord, using mutation rules and fitness evaluation based on rules of music composition. Another major difference is that our work evolves the probabilities of the operator and mutation rule selection. As the most fit individuals of the population survive to reproduce from generation to generation, their probabilities most often reflect the rules which contributed to their high fitness and are probabilistically more likely to be used again.

As future work, we plan to examine and encode more mutation and fitness rules based on more complicated rules from music theory, such as examining melodic contour, encouraging contrary motion of the parts, as well as a more complicated analysis of the chords in the resulting harmony. Furthermore, we also plan to encode a key-signature in the chromosome to allow for the evolution of richer harmonies and more complicated chord progressions. We will also examine using our system to evolve individual musical phrases, embedded within a second higher-level GA that will combine individual phrases into a longer composition including key-modulations and well-defined cadences. Lastly, to improve our fitness function and balance the many fitness rules we employ, we will empirically test the fitness rules on a selection of compositions from the Renaissance and Baroque periods to experimentally determine a set of weights.

References

1. Biles, J.: GenJam: A genetic algorithm for generating jazz solos. In: Proceedings of the International Computer Music Conference, pp. 131–131 (1994)
2. Cope, D.: Virtual Music: Computer Synthesis of Musical Style. The MIT Press, Cambridge (2004)
3. Göksu, H., Pigg, P., Dixit, V.: Music composition using Genetic Algorithms (GA) and multilayer perceptrons (MLP). In: Advances in Natural Computation, pp. 1242–1250 (2005)
4. Holland, J.: Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor (1975)
5. Horowitz, D.: Generating rhythms with genetic algorithms. In: Proceedings Of The National Conference On Artificial Intelligence, pp. 1459–1459. John Wiley & Sons, Chichester (1995)
6. Jacob, B.: Composing with genetic algorithms, pp. 452–455 (1995)
7. Kostka, S., Payne, D., Schindler, A.: Tonal harmony, with an introduction to twentieth-century music. McGraw-Hill, New York (2000)
8. Maddox, T., Otten, J.: Using an Evolutionary Algorithm to Generate Four-Part 18th Century Harmony. Mathematics and Computers in Modern Science: Acoustics and Music, Biology and Chemistry, Business and Economics, 83–89 (2000)
9. Marques, M., Oliveira, V., Vieira, S., Rosa, A.: Music composition using genetic evolutionary algorithms. In: Proceedings of the 2000 Congress on Evolutionary Computation, vol. 1, pp. 714–719 (2000)
10. McIntyre, R.: Bach in a box: The evolution of four part baroque harmony using the genetic algorithm. In: Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, pp. 852–857 (2002)
11. Papadopoulos, G., Wiggins, G.: AI methods for algorithmic composition: A survey, a critical view and future prospects. In: AISB Symposium on Musical Creativity, pp. 110–117 (1999)
12. Phon-Amnuaisuk, S., Wiggins, G.: The four-part harmonisation problem: a comparison between genetic algorithms and a rule-based system. In: Proceedings of the AISB 1999 Symposium on Musical Creativity (1999)
13. Towsey, M., Brown, A., Wright, S., Diederich, J.: Towards melodic extension using genetic algorithms, pp. 54–64. International Forum of Educational Technology & Society (2001)