Stephen Bussey

Dr. Alice Armstrong

Senior Research

August 20, 2013

Digital Music Composition: Dual Genetic Algorithms with Objective Fitness

1. **Introduction**

An emerging technique that has formed in composition of music is that of algorithmic music composition. Humans use many different tools, techniques, and processes for creating digital music, but many of these techniques are similar to the historical techniques used to develop analog music. For the purposes of this paper, rule based expert systems, Markov chain models, and genetic algorithms will be focused on, with a detailed focus on genetic algorithms. These algorithms allow for automated or semi-automated composition of music that seeks to be aesthetically pleasing to the human ear. These algorithms may either create direct music files or generate sheet music which can then be played by musicians. Along with the algorithms which have been developed, there has also been heavy statistical analysis of musical components which seeks to determine ways of automatically judging and classifying music. However, the current state of research comes with problems that may be addressed in order to produce a better and more objective algorithm. In order to analyze and understand these highly technical aspects of music composition, one must have a basic understanding of music and the components of music.

1. **Music Terms and Theory**

Pitch is one of the core concepts of music. Dragan Matic defines pitch as a “subjective feeling that the human ear hears, but [which] also has an objective value (for example, the frequency of an appropriate sound wave).”[[1]](#footnote-1) Pitches can be in relation to another note or they may be absolute with regards to the frequency of the sound wave. Pitches are distributed in the European standard in a system of twelve equally distributed semitones.[[2]](#footnote-2) This leads to a scale with seven core notes in it (A, B, C, D, E, F, G) as well as five sharp notes which are exactly one semitone above their respective note (A#, C#, D#, F#, G#). All notes together in order form the chromatic scale which is infinitely repeating in either direction. When two notes are played consecutively, the amount of vertical gap in the scale is called the melodic interval. When two notes are played at the same time, the amount of vertical gap in the scale is called the harmonic interval; multiple notes played at the same time do not always sound good together. [[3]](#footnote-3)

Rhythm is another core concept of music. A rhythm is a collection of pulses or “beats” which influences the timing of when sound is heard. These beats divide a composition into a series of measures. A measure can consist of any combination of beats, but a common styling in western music is to use four beats in a measure with the beats being subdivided. The subdivision of these beats can occur evenly (1, ½, ¼, 1/8, 1/16, etc.) or irregularly. Many algorithmic techniques for music composition choose to use the even subdivisions and cut off the subdivision at 1/8 of a beat.[[4]](#footnote-4) Pitch exists outside of time and therefore outside of any rhythm, and rhythm exists outside of any particular frequency; this separation means that notes can be composed without a rhythm and that a rhythm can be composed without any associated notes. It is the combination of pitches and rhythm into melodies (combination of notes and beats) which leads to the formation of music.

1. **Digital Music Formats**

Music can be represented in a variety of ways. Sheet music consists of a musical staff, which notes and rhythms are visually mapped against. This is a common way of giving music to a musician and is what an orchestra would play with. Digitally, music can be represented with sound files which are played with a music player to create audible sounds without the use of musicians. Two digital formats of particular interest are WAV and MIDI. The WAV format stores raw audio data in blocks which can then be played back. WAV is a lossless audio format meaning that the encoded audio is exactly the same as the input. This leads to WAV files being difficult to pull data out of without digital waveform analysis algorithms. One alternative is the MIDI format which stores song information such as “title, track names, and most importantly what instruments to use and the sequence of musical events, such as [pitches] and instrument control information needed to play back the song.”[[5]](#footnote-5) The MIDI format provides a way for players to play pitches (128 available in the specification) back without the pitches being encoded as waveforms in the MIDI file. This makes the file very easy to parse and pull data out of, without the use of digital audio waveform algorithms. Due to this simplicity in playing back and parsing, many researches have chosen to use MIDI as a file format for storing digitally composed music.[[6]](#footnote-6)

1. **Musical Statistics**

Research into the statistics of music composition has played a very important role in the development of many algorithms. Bill Manaris and fellow researchers utilized Zipf-based metrics to determine which elements of musical composition lend to evaluate and create “pleasant music.”[[7]](#footnote-7) Zipf’s Law states the frequency of occurrence of the nth ranked attribute of a metric is inversely proportional to its rank.[[8]](#footnote-8) This allows for the balance of attributes of music to be determined through statistical analysis. Some metrics of interest found by this research team include:[[9]](#footnote-9)

Pitch: The balance of pitch of music events in a given piece of music, relative to other pitches in the given piece of music.

Pitch mod 12: The balance of pitch on the twelve-note scale in a given piece of music, relative to the other pitches on the twelve-note scale.

Duration: The relative balance of durations of music events, independent of pitch, in a given piece of music.

Pitch & Duration: The relative balance of pitch-duration combinations, in a given piece of music.

Melodic Intervals: The relative balance of melodic intervals, in a given piece of music.

Harmonic Intervals: The relative balance of harmonic intervals, in a given piece of music.

Harmonic Bigrams: The relative balance of specific pairs of harmonic intervals in a given piece.

Melodic Bigrams: The relative balance of specific pairs of melodic intervals in a given piece.

The researchers in this paper are confident that the metrics above can be used to determine genre and differences between music varieties; this is crucial for developing an algorithm to evaluate musical pieces. Similar research has been performed by Michael Towsey et al., who evaluated a number of metrics which could be suitable for use in a genetic algorithm.[[10]](#footnote-10)

1. **Rule Based Algorithms for Music Composition**

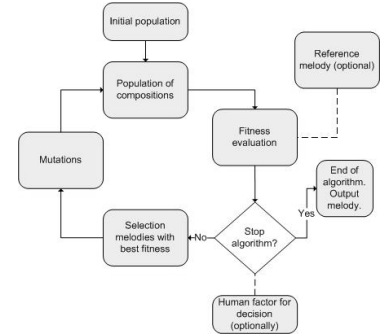
Historically, humans have employed a set of rules and processes for developing music. This set of rules has been applied to algorithms called expert systems. Expert systems are “information systems that represent expert knowledge for a particular problem area as a set of rules, and the perform inferences when new data are entered.”[[11]](#footnote-11) Expert systems seek to encode all of the rules, and heuristics, that occur in musical processes into a knowledge base which is then used in the music composition process.[[12]](#footnote-12) According to Rehberger, “A big part of composition is ‘hard work’ and inherently algorithmic whereas only a small part is considered to be genius”.[[13]](#footnote-13) This notion is confirmed by Bell who says, “Composers are following a set of rules and, despite their possible complexity, these rules can be followed by a computer to automate the composition process.”[[14]](#footnote-14) One expert music system that has been developed utilizes two levels of information and definitions. On one level, there is a set of factual information based on the processes of music. One the other hand, the user’s requirements are acted upon and the system can involve the user in the process of creating music.[[15]](#footnote-15) The involvement of a human operator hits upon a very important fact that limits the ability of rule based systems: the genius of music is lacking if only rules are followed. Genius is often a subjective feeling placed in music that cannot easily be modeled by a rule set. Another shortcoming of an expert system is that, by nature, expert systems will produce the same “optimal” solution to the same set of inputs.[[16]](#footnote-16) However, there are no solutions considered “optimal” in music composition. In order to avoid this problem, expert systems are developed to only loosely follow rules and integrate other solutions in order to avoid the same solution every time, such as the use of a human judge in the process. Another algorithm which utilizes rules in order to determine outcome is the Markov chain.

1. **Markov Model Algorithms for Music Composition**

Markov chains are “conditional probability systems where the probability of future events depends on one or more past events.”[[17]](#footnote-17) A Markov chain utilizes some known probability to a sequence of events in order to help determine the next event in the immediate future. A Markov model utilized by Herfort and Rehberger uses the immediate previous note in order to determine the next note against a set of probabilities.[[18]](#footnote-18) This is a fairly simple algorithm as the set of rules is not large, but it is extremely short-sighted by nature. This “first order” Markov chain is not capable of determining a probability based on a large sequence of pitches. To counteract this, “N-order” Markov chain models can be determined to look more deeply into the past, but the complexity of the rule set increases exponentially with added depth. Further, every aspect of music that is to be modeled would require its own Markov model. For instance, there would need to be a model for pitch, duration, etc. This leads to a potentially huge Markov model rule set where all chains are independent of each other. Thus, Markov chains are good at composing musical melodies on the micro level (such as a measure) but fail to produce quality music at the macro level.[[19]](#footnote-19) Genetic algorithms can potentially solve this problem.

1. **Genetic Algorithms for Music Composition**

Genetic algorithms are based on the modern evolutionary theory; they make use of the same processes which nature uses to select the individual fittest to its environment: natural selection and sexual reproduction.[[20]](#footnote-20) Genetic algorithms consist of several steps: Populate an initial set of chromosomes, evaluate the fitness of these chromosomes, evaluation / halting, selection / reproduction, mutation, and repopulation. This is very similar to the high level natural processes which lead to the development of a strong population. However, it is important to note that this natural process can often take place over much iteration and can therefore be very slow. Genetic algorithms are best for searching large and ill-behaved problem spaces, which perfectly describes the music composition problem space.[[21]](#footnote-21)

[[22]](#footnote-22)

**Figure 1:** Steps of Genetic Algorithm used for music composition

The chromosomes are the representation of a single problem solution instance. Chromosomes are often encoded as binary strings which allows for agnostic techniques to be used in reproduction and mutation steps. However, some researchers have chosen to use symbolic representation of structures and data in order to provide more meaningful operations in the context of the domain knowledge.[[23]](#footnote-23) Meaning operations allow for more precise reproduction and mutation stages which means that the algorithm can converge faster and with better results.[[24]](#footnote-24)

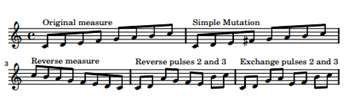
Biles utilized two levels of chromosomes in order to isolate the difference between a measure and an entire phrase, which can consist of many measures. The phrase contains pointers to individual measure chromosomes as well as an overall fitness score.[[25]](#footnote-25) The measure chromosome consists of the fitness score as well as sixteen available notes.[[26]](#footnote-26) The interesting aspect of this approach is that the fitness can be scored on multiple levels which could help lead to a more accurate set of fitness functions. Another approach is that of De Freitas, who encoded MIDI information directly into the chromosome as a string of bits. This allows for general rules to be allowed in reproduction and mutation.[[27]](#footnote-27) As mentioned prior, it is trivial to extract much information about notes and rhythm in a MIDI representation, meaning this approach is very dense with data. One last approach was taken by Ralley. His method uses a header and body where the header contains information about the starting pitch. The body of the chromosome contains the melodic intervals of each note when compared to the previous note.[[28]](#footnote-28) This approach can lead to very easy manipulation of data. Most approaches to chromosome representation are very similar or completely binary and lead to the same available data; however, the most contented aspect of genetic algorithms in music composition is the fitness function.

A fitness function measures how well a particular chromosome can perform in the task environment. An unfit chromosome will be seen as undesirable when compared to a fit chromosome. This is the natural selection aspect of a genetic algorithm which takes place because “not all chromosomes are identical, some may be more likely to survive than others”.[[29]](#footnote-29) The fitness function is critical for a successful genetic algorithm because a “genetic algorithm can only produce melodies as good as the knowledge encoded in the fitness function.”[[30]](#footnote-30) Due to this fact, research is currently heavily oriented towards identifying the different possibilities with regards to a successful fitness function.

There are many different approaches to take with a fitness function and most research varies on how the fitness of a chromosome is calculated. For instance, Biles took advantage of human power by utilizing humans as the fitness mechanism in a genetic algorithm. A human mentor powered the fitness function by stating whether a measure was good or bad in the context of a melody. The count of this at the end of the melody is the fitness of that melody.[[31]](#footnote-31) This approach takes advantage of the fact that the human ear is capable of very quickly determining whether a pitch sounds good or bad in context. This is not a trivial task to do algorithmically.[[32]](#footnote-32) While this can lead to an effective evaluation of a chromosome population, the downside is that this takes an incredible amount of time and can become infeasible to do on a large scale. De Freitas backs this idea by stating that, “fitness computation in most evolutionary-based systems for art and must requires aesthetic judgments, which are not easy to model and implement in the form of an algorithm.”[[33]](#footnote-33) While some researchers can see past the speed and time disadvantages, many are looking for ways to completely automate the process and remove the human element.

Jensen utilized the Zipf-based metrics listed previously which were identified by Manaris. These metrics were applied as the fitness function in a vector-based system which applied the Zipf metrics to product small four measure melodies.[[34]](#footnote-34) The outcome of this research was favorable to the notion that it is possible to have a metric based fitness function, but problems with regards to which fitness metrics were applied and the weight of their application limited the promising melodies.[[35]](#footnote-35) Horowitz took advantage of both automated and human interaction in his fitness function in a genetic algorithm optimized for developing rhythm. Human mentors are used to assign fitness values based on the satisfaction of a set of criteria, but several objective functions are applied intermediately to week out weak rhythms.[[36]](#footnote-36) This approach allows for an optimized use of mentor time as they will not be weeding out very poor submissions manually. Other notable algorithmic fitness function criteria involve the use of melodic intervals to determine relative “acceptableness” between two pitches. Khalifa identified melodic intervals as follows: one to four half steps are acceptable, five to seven are acceptable with a resolving third note, and eight or more are never acceptable.[[37]](#footnote-37) Also, certain notes work better in the downbeat (first beat of a measure) and others work best on the upbeat (third beat of a measure).[[38]](#footnote-38) Also, if the composition imposes a restriction on the available rhythms, then a slow song may not be aesthetically pleasing under any condition without proper subdivision of notes that is not possible under constraint.[[39]](#footnote-39) Algorithmic fitness functions will never be universally correct. However, they must be consistent in how they evaluate results.[[40]](#footnote-40) A consistency among the fitness function will not provide some chromosomes with an unfair advantage over others. One aspect which has been agreed upon fairly unanimously in research papers are the aspects of mutation available for genetic algorithms.

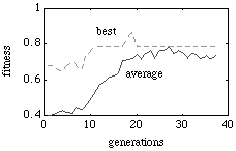
Mutation is a non-frequently occurring event where a chromosome is randomly mutated in some way at a fixed percentage in order to change an aspect of that chromosome. Mutation seeks to provide variation in a population which may not be easily achievable otherwise, leading to the elimination of local maxima in the search space which the genetic algorithm would converge on otherwise. Several techniques have been utilized such as reversing a measure, rotating the notes in a measure by a certain amount, inverting the notes of a measure, sorting a measure, or shifting the pitch of notes.[[41]](#footnote-41) Other techniques include reversing the rhythmic pulses of a measure or exchanging pulses of a measure.[[42]](#footnote-42)



**Figure 2:** Mutations used by De Freitas[[43]](#footnote-43)

One thing to note about all of these mutations is that they are very specific to the problem domain. Papadopoulos suggested that a “weak genetic operator” will lead to longer and less accurate searches. A weak genetic operator would include blindly flipping bits and blindly operating on a chromosome string without knowledge of how the chromosome is stored.

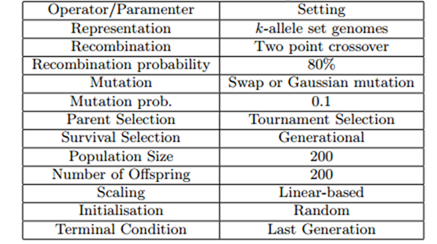
The process of selecting which chromosomes will undergo reproduction is usually done using a tournament selection algorithm. This algorithm allows for a random population polling to be taken, and then an individual is selected from the population based on a probability which scales according to the fitness. This would give a very fit chromosome a higher probability of being selected if it is present in the random population. This is a pretty common mechanism used in current research, but there is a very large problem. Ralley noticed that the best chromosomes can be easily lost in tournament selection.



**Figure 3:** Graph of the best versus average fitness of a population over generations[[44]](#footnote-44)

This can be mitigated using certain techniques such as caching the best chromosomes from generation to generation[[45]](#footnote-45), but largely is not possible to prevent in this mechanism. Another algorithm used in selection is the proportionally give a selection probability to all chromosomes, based on fitness, and to then randomly select chromosomes based on their probability.

After a set of chromosomes is selected to undergo reproduction, they go through a crossover mechanism which produces the next generation of chromosomes. A single random crossover point can be used which would split two chromosomes and splice the results to create a new one. While this may be sufficient for exploring a population in a less artistic task than that of composing music, it is not as advantageous as using an intelligent crossover selection.[[46]](#footnote-46) One way this can be overcome is to use pulse and rhythm detection of the chromosomes and then doing a one-point or two-point crossover. [[47]](#footnote-47) However, this mechanism requires that rhythm and pitches are represented by the same chromosome. Most research with regard to generating music with genetic algorithms puts less focus on crossover and more focus on the fitness function and mutations.[[48]](#footnote-48) A complete example of a genetic algorithm is presented in Table 1.



**Table 1:** Genetic Algorithm Sample Configuration[[49]](#footnote-49)

Genetic algorithms are extremely powerful for analyzing the complex search space that exists in the music composition problem domain. Many researchers have looked towards the genetic algorithm as an algorithm that can be capable of producing bits of “genius” in a composition. This genius is very hard to achieve in alternative algorithms, and so genetic algorithms are very promising. Some researchers have combined algorithms, such as Bell utilizing 5184 entry Markov chains to select an immediate predecessor note in the composition process, but then analyzing these Markov chains using genetic algorithms.[[50]](#footnote-50) This technique allows for the strength of the Markov chain (selecting a quality short term note) with the strength of genetic algorithms (analyzing a large unpredictable search space to determine the quality of the composition).

1. **Current Algorithms Disadvantages**

A huge disadvantage to using genetic algorithms is that they converge very slowly on a solution due to their nature of randomness and slow changes. However, there is a way to overcome this disadvantage, by using memetic algorithms. A memetic algorithm is a genetic algorithm which adds one additional step: a fix step where an undesirable part of a chromosome is always correct.[[51]](#footnote-51) In a musical context, an example would be fixing an out of key pitch in order to replace the pitch with a more pleasant pitch. Over fewer generations than required in a normal genetic algorithm, these small fixes provide a quicker convergence on an acceptable solution. In order for a memetic algorithm to be effective, domain knowledge must be encoded in the improvement mechanism; a random change cannot be made as a random change may not produce a higher fitness chromosome.

The algorithms above all have drawbacks to how they perform and what they produce. One limitation comes from data representation. Most researchers utilize a self-imposed limit on what pitches or rhythms can be encoded. This is to prevent an unnecessarily large search space. These assumptions mean that the digitally composed music sounds best at certain beats per minute[[52]](#footnote-52) or that the composed music will not have a full set of pitches available.

Another drawback is that no algorithm presented can account for the “genius element” in the composition process. Genetic algorithms perform this task best, but only by sheer randomness. Rule-based systems are not capable of achieving the “genius element” because it is not able to be encoded in a knowledge representation. However, while there is a lack of genius overall in compositions, there is much mediocrity and relatively pleasant sounding music. While mediocrity is not usually a goal to aim for, it does mean that current research does not completely fail.

1. **Tools**

There are several tools based in the java language which can help in the process of digital music composition and with implementing genetic algorithms. Java Genetic Algorithm Package (JGAP) is a java package which provides a full implementation of common genetic algorithm patterns.[[53]](#footnote-53) JFugue is a “Java API for music programming” which allows for MIDI playback of MIDI files and special JFugue note strings which can easily be translated to and from binary strings.[[54]](#footnote-54) These packages do not provide complex statistical analysis of waveforms or MIDI files, but they do offer a starting point for development and research in the field.

1. **Literature Review Conclusion**

The current state of research in the field of digital music composition is promising although definitely not complete. Genetic algorithms appear to be most promising for future research in this area due to the flexibility and “genius” that a genetic algorithm can provide. Expert systems do not provide this flexibility, as the rule sets are encoded in their entirety in an implementation. Markov models are not able to produce long sections of coherent melodies without exponential growth in requirements. A genetic algorithm implementation can address these areas of concern, although current research is too dependent on a human operator for providing fitness, and one genetic algorithm is used for both rhythmic and pitch composition. A genetic algorithm which addresses these areas of concern could produce better end results.

1. **Primary Objective**

To compose music using genetic and memetic algorithms by concurrently and separately processing the rhythmic and pitch components subject to the following criteria:

* The level of effort shall not exceed one person-month over twelve weeks.
* The rhythmic component will be limited to the following set: whole note, half note, quarter note, eighth note, sixteenth note, whole rest, half rest, quarter rest, eighth rest, sixteenth rest.
* All music is assumed to be composed in 4/4 time (four beats to a measure).
* All music is composed without the use of tied notes.
* The pitch component will be limited to the pitches available via the MIDI specification.
* The composed music will be presented in MIDI format.
* The input music will consist of no less than thirty “walking bass lines” ranging between eight and twelve measures each.
* The input music will contain no tied notes.

The input music will feature walking bass lines in MIDI format. These inputs will be analyzed and converted into a working format using a music analysis process that will be described in detail in the solution description.

1. **Solution Overview**

The input MIDI files will be converted into an object oriented representation which is compatible with the music analysis process. The converted input will be analyzed using the music analysis process, which will produce one rhythmic music analysis vector and one pitch music analysis vector for each input.

A rhythmic genetic algorithm will execute, using a single rhythmic music analysis vector for determination of fitness. The genetic algorithm will run until a suitable fitness level is reached. The fitness level will be calculated by finding the distance between the input music analysis vector, and a vector produced for each chromosome in the fitness determination process. Once a suitable fitness target is reached, the rhythmic genetic algorithm will return an array of rhythmic measures which will be later combined with pitch measures in order to produce a MIDI output.

A pitch memetic algorithm will start concurrently with the rhythmic genetic algorithm, using a single pitch music analysis vector for determination of fitness. The memetic algorithm will have a number of notes equal to 4\*(input measures length). This will allow for a fixed number of notes to be evaluated, which should be suitable for walking bass lines. The memetic algorithm will run until a suitable fitness level is reached. The fitness level is determined using the same process as the rhythmic genetic algorithm. A memetic improvement will be used on each chromosome at each generation. Once a suitable fitness target is reached, the pitch memetic algorithm will return an array of pitch measures.

Once a set of pitch measures and rhythmic measures are produced, an output MIDI file will be produced by combining the pitch and rhythmic aspects. If there are more tones available than beats produced by the rhythmic genetic algorithm, then only the number of tones needed will be used (effectively discarding the later tones). If there are fewer tones available than needed, the tones will start to be used, in order, from the beginning of the set of tones. Once this process is completed, the total time to completion will be logged as well as the final output music in MIDI format.

1. **Music Analysis Process**

The music analysis process analyzes a set of beats and pitches in order to produce two music analysis vectors (rhythmic and pitch), which contain the values of individual music metrics. These beats and pitches are in object format and not MIDI format. Music metrics are taken from research by Manaris et al.[[55]](#footnote-55) and Towsey et al.[[56]](#footnote-56) The metrics for rhythm are as follows:

* Rhythmic Variety – Proportion of beat durations used compared to all available durations
  + Numerator – Number of distinct beat lengths
  + Denominator – Number of possible beats (10)
* Rest Density – Proportion of rest beats to note beats
  + Numerator – Silent beats length
  + Denominator – All beats length
* Syncopation – Proportion of beats having duration >= 1 and start off beat
  + Numerator – Number of syncopated beats
  + Denominator – Number of beats
* Rest Rhythmic Range – Range of rest beats divided by the maximum possible range
  + Numerator – Maximum rest beat duration / Minimum rest beat duration
  + Denominator – 16 = (4 / .25)
* Note Rhythmic Range – Range of note beats divided by the maximum possible range
  + Numerator – Maximum note beat duration / minimum note beat duration
  + Denominator - 16

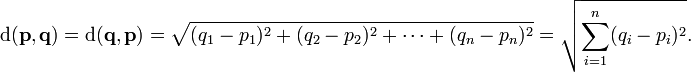
The music analysis process will produce a five length vector where each position of the vector corresponds to exactly one metric. A different set of metrics will be used for pitches:

* Pitch Variety – Proportion of distinct pitches to notes
  + Numerator - Number of distinct pitches
  + Denominator – Number of notes
* Pitch Range – Proportion of range of pitches to the desired pitch range. The pitch range for an input MIDI file will always be 1, as the desired range is the range of the input. The desired pitch range of a chromosome will be the range of the input.
  + Numerator – Highest pitch value – lowest pitch value
  + Denominator – Desired Range
* Dissonant Intervals – Sum of all intervals dissonance values. These values are presented in table 2
  + Numerator – Sum of all interval dissonances
  + Denominator – Number of notes – 1
* Diatonic Step – Proportion of intervals which step by one or two
  + Numerator – Number of diatonic steps
  + Denominator – Number of notes – 1
* Large Leaps – Proportion of intervals which step by six or more
  + Numerator – Number of large leaps
  + Denominator – Number of notes – 1
* Repeated Pitches – Proportion of intervals where both notes are the same pitch
  + Numerator – Number of repeated pitch intervals
  + Denominator – Number of notes -1

|  |  |
| --- | --- |
| Interval Leap Size | Dissonance Value |
| 0, 1, 2, 3, 4, 5, 7, 8, 9, 12 | 0.0 |
| 10 | 0.5 |
| 6, 11, >= 13 | 1.0 |

**Table 2:** Values of dissonance for given interval leaps[[57]](#footnote-57)

The music analysis process will produce a music analysis vector for the rhythmic and pitch metrics for a given input. The vectors will always be the same for a unique input. The distance between two vectors can be found by using the Euclidean distance formula:



**Formula 1:** Euclidean distance formula

This distance between two vectors will be applied in the genetic algorithms’ fitness function in order to find the error between a target vector, given as input, and the vector found for a given chromosome.

1. **Rhythmic Genetic Algorithm**

A genetic algorithm implementation will be created for this experiment. A tool like JGAP, although mature and stable, will not be used due to time constraints. The genetic algorithm will follow the standard design of a genetic algorithm given earlier in this paper. Figure 4 and Figure 5, below, present planned UML diagrams of relevant genetic algorithm classes, Chromosome and Mutation.

A rhythm chromosome consists of a fixed number of measures, equal to the number of measures in the input. A measure consists of beats which consists of duration. The duration available to a measure are established in the limiting criteria: .25, .5, 1, 2, 4 representing 1/6, 1/8 ¼ ½, 1 notes. A beat can be a rest, which is represented by a boolean flag.

A rhythm chromosome breeds with another rhythm chromosome by splitting the array of measures at a random single location and combining with the corresponding set of measures.

A rhythm chromosome can mutate on an individual measure level, as well as the entire array of measures level. On the measure level, the following mutations are possible: a beat duration can be split down to the minimum (.25); two beats can be combined to the maximum beat duration (4); two beats’ order can swap; the entire measure can reverse order; the measure can reset to beat durations (1,1,1,1); and the measure can completely randomize. On the array of measures level, the following mutations are possible: the entire array of measures can reverse order; and two measures’ order can swap. All of these mutations happen on random components of the measure and will happen at a fixed, low, percentage. This percentage is an operating point pilot which will be determined before experimentation.

The fitness of a rhythm chromosome has been presented previously. The input music analysis vector will be used in determining fitness. This provides for a fitness function that is tailored to the input instead of a known standard. The target fitness is an operating point pilot.

1. **Pitch Chromosome**

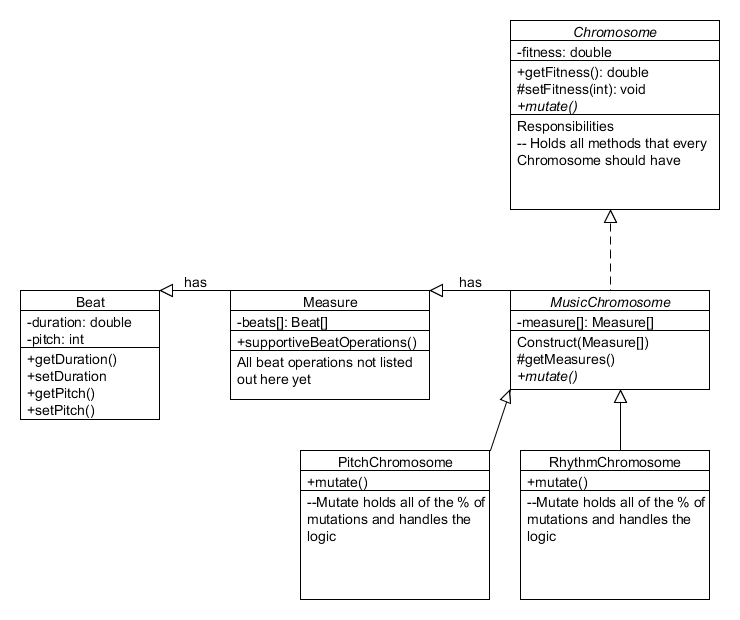
A pitch chromosome consists of a fixed number of measures, equal to the number of measures in the input. Each measure consists of four pitches. A pitch is an integer in the range of 0-127; this is based on the MIDI specification. The octave-less pitch can be found by finding the remainder of (pitch / 12). A fixed number of pitches is useful as it allows for the rhythmic genetic algorithm and pitch memetic algorithm to be executed simultaneously. If this did not happen, the rhythmic genetic algorithm would have to finish before the pitch memetic algorithm could start, which would produce an execution time equal to (Trhythm + Tpitch), rather than an improved time of (MAX(Trhythm, Tpitch)). If the execution time was not a point of concern, then the rhythmic genetic algorithm could execute in its entirety, and the pitch memetic algorithm would have the exact number and separation of beats that will be used in the final product.

A pitch chromosome breeds with another pitch chromosome by splitting the array of measures at a random single location and combining with the corresponding set of measures.

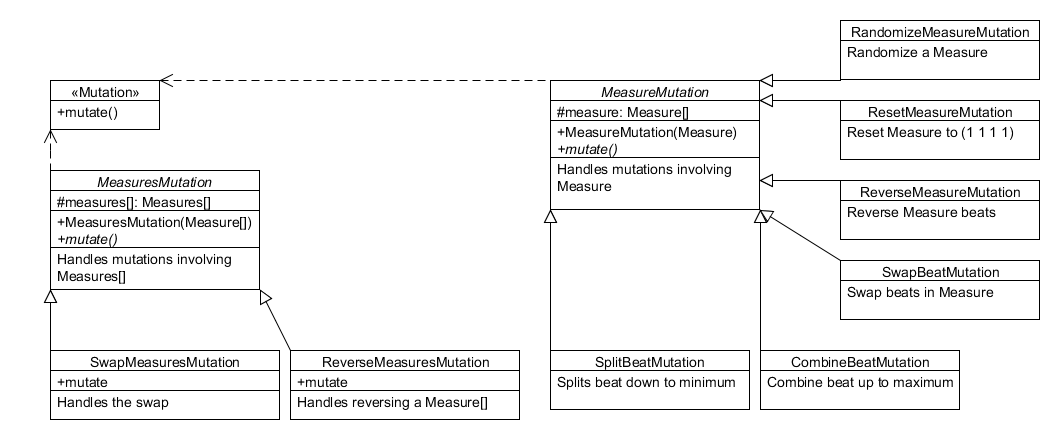
A pitch chromosome can mutate on an individual measure level, as well as the entire array of measures level. On the measure level, the following mutations are possible: two pitches’ orders can swap; the entire measure can reverse order; the measure can completely randomize; the measure’s pitches can transpose by one value up or down; and one pitch in the measure can transpose by one value up or down. On the array of measures level, the following mutations are possible: the entire array of measures can reverse order; two measures’ orders can swap. All of these mutations happen on random components of the measure and will happen at a fixed, low, percentage. This percentage is an operating point pilot which will be determined before experimentation.

The fitness of a pitch chromosome has been presented already. The input music analysis vector will be used in determining fitness. This provides for a fitness function that is tailored to the input instead of a known standard.

Every chromosome in every generation of the pitch memetic algorithm will undergo a variation which will improve on a single unwanted feature of the chromosome. The first pitch which does not fit into the key of the rest of its measure will be transposed by one pitch up or down. This will correct the pitch and make it fit in with the rest of a measure. This memetic improvement will happen with a 100% probability on every chromosome in every generation. The goal of this improvement is to more quickly converge onto the target fitness.



**Figure 4:** Chromosome classes of the solution

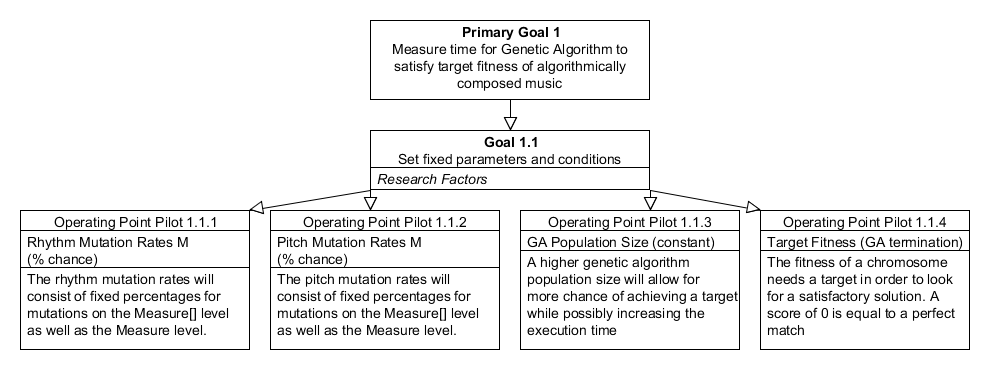


**Figure 5:** Mutation classes of the solution

1. **Hypotheses and Goals**

Null Hypothesis 1: The music analysis process generates different music analysis vectors for different pieces of music.

Null Hypothesis 2: The genetic algorithm produces a result satisfying the input music analysis vectors within ten minutes, at least 75% of the time.



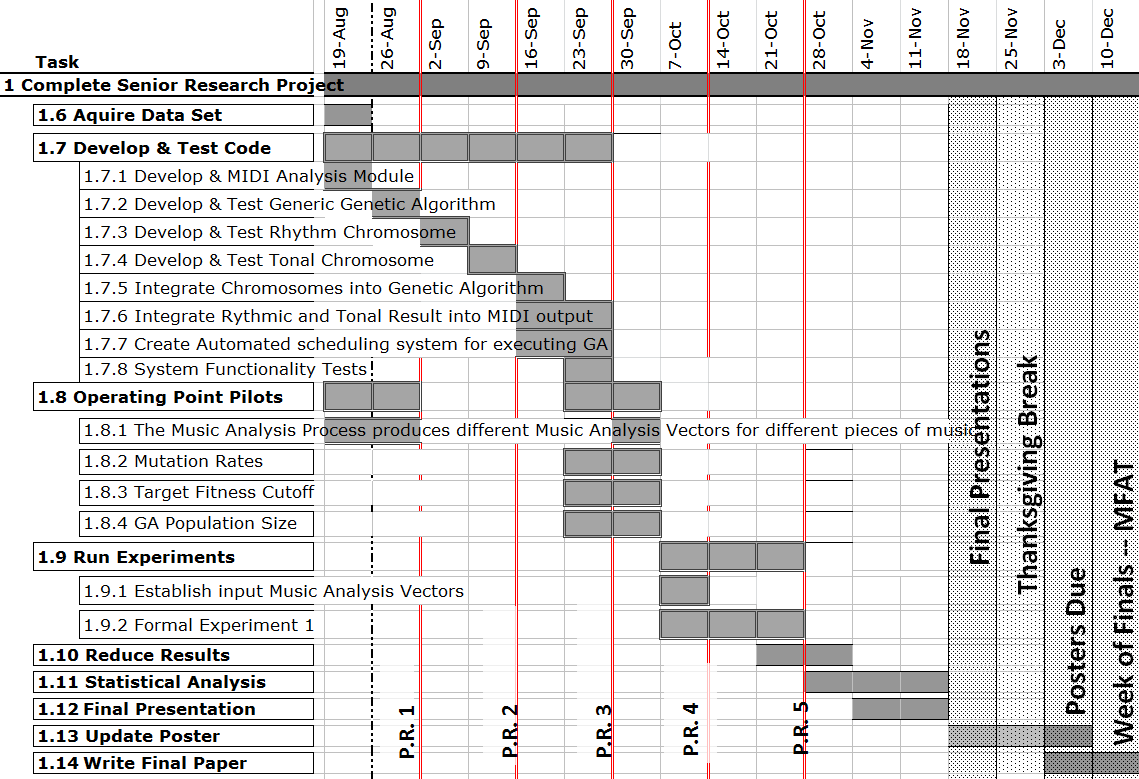
**Figure 6:** Goal Tree with Operating Point Pilots

Figure 6 consists of the goals and operating point pilots required for experimentation. The operating point pilots are critical in order to successfully tune and optimize the genetic algorithm. The mutation rates should be low, but not too low. If the rates are too high, the good aspects of a chromosome could be quickly lost, leading to a slow convergence onto a solution.

1. **Experimentation**

For each input music, genetic algorithms will be run, in parallel, for rhythmic and pitch components. The time to completion will be monitored for each. The results of the genetic algorithms will be combined into an output MIDI file. The output MIDI file and time to completion will be recorded. This process will be run a minimum of five times for each input MIDI file, leading to a minimum of 150 experiments being performed.

1. **Schedule**



**Figure 7:** Project Schedule

The schedule in Figure 7 consists of a number of tasks which must be worked on and completed on the dates specified. Progress Reports (P.R.) are where certain tasks are reviewed and critiqued, in order to ensure that the project tasks are on time.

1. **Conclusion**

Genetic algorithms appear to be the best class of algorithms suited for creative and automated digital music composition. The pitfalls of current research are that the implementations are using a single genetic algorithm and human operators are being used for fitness evaluation. By correcting these pitfalls with dual genetic and completely objective algorithms, better quality music may be composed without the involvement of a human operator.

Works Cited

Beer, Michael. "Mathematics and Music: Relating Science to Arts?" *Mathematical Spectrum*41.1 (2008): 36-42. Web. 12 Feb. 2013. <http://www.michael.beer.name/file\_download/1/mathandmusic.pdf>.

Bell, Chip. "Algorithmic Music Composition Using Dynamic Markov Chains and Genetic Algorithms." Journal of Computing Sciences in Colleges 27.2 (2011): 99-107.ACM Digital Library. Dec. 2011. Web. 11 Feb. 2013.

Biles, John. "GenJam: A Genetic Algorithm for Generating Jazz Solos." *In Proceedings of the International Computer Music Conference* (1994): n. pag. *CiteSeerX*. Web. 12 Feb. 2013. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.55.6146&rep=rep1&type=pdf>.

De Freitas, Alan R.R., and Frederica Gadelha Guimaraes. "Originality and Diversity in the Artificial Evolution of Melodies." Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation (2011): 419-26. ACM Digital Library. Web. 11 Feb. 2013.

Horner, A., and Goldberg, 1991. "Genetic Algorithms and Computer-Assisted Music Composition," in proceedings of the 1991 International Computer Music Conference, pp. 479-482.

Horowitz, Damon. "Generating Rhythms with Genetic Algorithms." *ICMC Proceedings*(1994): 142-43. Web. 12 Feb. 2013. <http://quod.lib.umich.edu/cgi/p/pod/dod-idx/generating-rhythms-with-genetic-algorithms.pdf?c=icmc;idno=bbp2372.1994.035>.

Jensen, Johannes H., and Pauline C. Haddow. "Evolutionary Music Composition Based on Zipf's Law." Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation (2011): 41-42. ACM Digital Library. Web. 11 Feb. 2013.

*JGAP: Java Genetic Algorithms Package*. Computer software. *JGAP*. Vers. 3.6.2. Source Forge, n.d. Web. 12 Feb. 2013. <http://jgap.sourceforge.net/>.

*JFugue: Java API for Music Composition*. Computer software. *JFugue*. Vers. 4.0.3. Web. 12 Feb. 2013. <http://www.jfugue.org/>.

Khalifa, Y., and M. Basel Al-Mourad. "Autonomous Evolutionary Music Composer."*Proceeding GECCO '06 Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* (2006): 1873-874. *ACM Digital Library*. Web. 11 Feb. 2013.

Rehberger, Leopold. “Composing music using AI algorithms.” GRIN Publishing GmbH. 2004. <<http://www.grin.com/en/e-book/108991/composing-music-using-ai-algorithms>>.

Manaris, Bill, Dallas Vaughan, Christopher Wagner, Juan Romero, and Robert B. Davis. "Evolutionary Music and the Zipf-mandelbrot Law: Developing Fitness Functions for Pleasant Music." Proceeding EvoWorkshops'03 Proceedings of the 2003 International Conference on Applications of Evolutionary Computing (2003): 522-34. ACM Digital Library. 2003. Web. 11 Feb. 2013.

MATIĆ, Dragan. "Yugoslav Journal of Operations Research." *A Genetic Algorithm for Composing Music*. N.p., n.d. Web. 12 Feb. 2013. <http://www.doiserbia.nb.rs/img/doi/0354-0243/2010/0354-02431001157M.pdf>.

"MIDI File Format." *The Sonic Spot*. N.p., n.d. Web. 12 Feb. 2013. <http://www.sonicspot.com/guide/midifiles.html>.

Papadopoulos, George, and Geraint Wiggins. "A Genetic Algorithm for the Generation of Jazz Melodies." (n.d.): n. pag. Web. 12 Feb. 2013. <<http://www.cs.cmu.edu/afs/cs.cmu.edu/project/music/web/files/Genetic%20Algorithm%20for%20Generation%20of%20Jazz%20Melodies.pdf>>.

Ralley, David. "Genetic Algorithms as a Tool for Melodic Development." *Proceedings of the 1995 International Computer Music Conference* (1995): n. pag. Web. 12 Feb. 2013. <http://minuet.dance.ohio-state.edu/ralley2/icmc/icmc.html>.

Towsey, Michael, Andrew Brown, Susan Wright, and Joachim Diederich. "Towards Melodic Extension Using Genetic Algorithms." Educational Technology & Society 4.2 (2011): n. pag. Educational Technology & Society. International Forum of Educational Technology & Society, Apr. 2001. Web. 11 Feb. 2013.

Wells, Derek, and Hala ElAarag. "A Novel Approach for Automated Music Composition Using Memetic Algorithms." Proceedings of the 49th Annual Southeast Regional Conference (2011): 155-59. ACM Digital Library. Web. 11 Feb. 2013.

1. MATIĆ, Dragan. "Yugoslav Journal of Operations Research." A Genetic Algorithm for Composing Music. N.p., n.d. Web. 12 Feb. 2013, 158. [↑](#footnote-ref-1)
2. Matic, 158. [↑](#footnote-ref-2)
3. Beer, Michael. "Mathematics and Music: Relating Science to Arts?" Mathematical Spectrum41.1 (2008): 36-42. Web. 12 Feb. 2013, 2 [↑](#footnote-ref-3)
4. Biles, John. "GenJam: A Genetic Algorithm for Generating Jazz Solos." In Proceedings of the International Computer Music Conference (1994): n. pag. CiteSeerX. Web. 12 Feb. 2013, 6. [↑](#footnote-ref-4)
5. "MIDI File Format." The Sonic Spot. N.p., n.d. Web. 12 Feb. 2013. [↑](#footnote-ref-5)
6. Horner, A., and Goldberg, 1991. "Genetic Algorithms and Computer-Assisted Music Composition," in proceedings of the 1991 International Computer Music Conference, 479-482. [↑](#footnote-ref-6)
7. Manaris, Bill et al. "Evolutionary Music and the Zipf-mandelbrot Law: Developing Fitness Functions for Pleasant Music." Proceeding EvoWorkshops'03 Proceedings of the 2003 International Conference on Applications of Evolutionary Computing (2003): 522-34. ACM Digital Library. 2003. Web. 11 Feb. 2013, 1. [↑](#footnote-ref-7)
8. Ibid., 2. [↑](#footnote-ref-8)
9. Manaris., 3. [↑](#footnote-ref-9)
10. Towsey, Michael et al., "Towards Melodic Extension Using Genetic Algorithms." Educational Technology & Society. International Forum of Educational Technology & Society, Apr. 2001. Web. 11 Feb. 2013. [↑](#footnote-ref-10)
11. Raynor, W. J. The International Dictionary of Artificial Intelligence. Glenlake Publishing, pp. 100 [↑](#footnote-ref-11)
12. Rehberger, Leopold. “Composing music using AI algorithms.” GRIN Publishing GmbH. 2004, 4.2.1. [↑](#footnote-ref-12)
13. Ibid., 4.4. [↑](#footnote-ref-13)
14. Bell, Chip. "Algorithmic Music Composition Using Dynamic Markov Chains and Genetic Algorithms." Journal of Computing Sciences in Colleges 27.2 (2011): 99-107.ACM Digital Library. Dec. 2011. Web. 11 Feb. 2013, 99. [↑](#footnote-ref-14)
15. Camurri 119 [↑](#footnote-ref-15)
16. Rehberger, 4.2.2. [↑](#footnote-ref-16)
17. Rehberger, 4.1.1. [↑](#footnote-ref-17)
18. Ibid., 4.1.1. [↑](#footnote-ref-18)
19. Ibid., 4.1.1. [↑](#footnote-ref-19)
20. Ibid., 4.4. [↑](#footnote-ref-20)
21. Biles, 1. [↑](#footnote-ref-21)
22. Matic, 163. [↑](#footnote-ref-22)
23. Papadopoulos, George, and Geraint Wiggins. "A Genetic Algorithm for the Generation of Jazz Melodies." (n.d.): n. pag. Web. 12 Feb. 2013, 1. [↑](#footnote-ref-23)
24. Ibid., 1. [↑](#footnote-ref-24)
25. Biles, 2. [↑](#footnote-ref-25)
26. Ibid., 3. [↑](#footnote-ref-26)
27. De Freitas, Alan R.R., and Frederica Gadelha Guimaraes. "Originality and Diversity in the Artificial Evolution of Melodies." Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation (2011): 419-26. ACM Digital Library. Web. 11 Feb. 2013, 419. [↑](#footnote-ref-27)
28. Ralley, David. "Genetic Algorithms as a Tool for Melodic Development." Proceedings of the 1995 International Computer Music Conference (1995): n. pag. Web. 12 Feb. 2013. [↑](#footnote-ref-28)
29. Rehberger, 4.4. [↑](#footnote-ref-29)
30. Rehberger, 4.4. [↑](#footnote-ref-30)
31. Biles, 2. [↑](#footnote-ref-31)
32. Biles, 2. [↑](#footnote-ref-32)
33. De Freitas, 419. [↑](#footnote-ref-33)
34. Jensen, Johannes H., and Pauline C. Haddow. "Evolutionary Music Composition Based on Zipf's Law." Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation (2011): 41-42. ACM Digital Library. Web. 11 Feb. 2013, 41. [↑](#footnote-ref-34)
35. Ibid., 42. [↑](#footnote-ref-35)
36. Horowitz, Damon. "Generating Rhythms with Genetic Algorithms." ICMC Proceedings(1994): 142-43. Web. 12 Feb. 2013, 142. [↑](#footnote-ref-36)
37. Khalifa, Y., and M. Basel Al-Mourad. "Autonomous Evolutionary Music Composer."Proceeding GECCO '06 Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (2006): 1873-874. ACM Digital Library. Web. 11 Feb. 2013, 1873. [↑](#footnote-ref-37)
38. Papadopoulos, 1. [↑](#footnote-ref-38)
39. Ibid., 1. [↑](#footnote-ref-39)
40. Ibid., 1. [↑](#footnote-ref-40)
41. Biles, 5. [↑](#footnote-ref-41)
42. De Freitas, 422. [↑](#footnote-ref-42)
43. Ibid., 422. [↑](#footnote-ref-43)
44. Ralley. [↑](#footnote-ref-44)
45. Ibid. [↑](#footnote-ref-45)
46. De Freitas, 421. [↑](#footnote-ref-46)
47. Ibid., 421. [↑](#footnote-ref-47)
48. Papadoupolos, 1. [↑](#footnote-ref-48)
49. Khalifa, 1873. [↑](#footnote-ref-49)
50. Bell, 100. [↑](#footnote-ref-50)
51. Wells, Derek, and Hala ElAarag. "A Novel Approach for Automated Music Composition Using Memetic Algorithms." Proceedings of the 49th Annual Southeast Regional Conference (2011): 155-59. ACM Digital Library. Web. 11 Feb. 2013. [↑](#footnote-ref-51)
52. Biles, 6. [↑](#footnote-ref-52)
53. JGAP: Java Genetic Algorithms Package. Computer software. JGAP. Vers. 3.6.2. Source Forge, n.d. Web. 12 Feb. 2013. [↑](#footnote-ref-53)
54. JFugue: Java API for Music Composition. Computer software. JFugue. Vers. 4.0.3. Web. 12 Feb. 2013. [↑](#footnote-ref-54)
55. Manaris., 3. [↑](#footnote-ref-55)
56. Towsey., 57. [↑](#footnote-ref-56)
57. Towsey., 57. [↑](#footnote-ref-57)