

# A MEMETIC ALGORITHM FOR AUTOMATED MUSIC COMPOSITION

by

DEREK WELLS

Advisor

DR. HALA EL AARAG

A senior research proposal submitted in partial fulfillment of the requirements  
for the degree of Bachelor of Science  
in the Department of Mathematics and Computer Science  
in the College of Arts and Science  
at Stetson University  
DeLand, Florida

Fall Term  
2010

## **ACKNOWLEDGMENTS**

The author would like to take this opportunity to thank Dr. El Aarag for all of her help in putting this proposal together.

## TABLE OF CONTENTS

ACKNOWLEDGMENTS.....	ii
TABLE OF CONTENTS.....	iii
LIST OF FIGURES.....	iv
LIST OF TABLES.....	v
ABSTRACT.....	1
I. INTRODUCTION.....	2
II. RELATED WORK.....	3
II.I Applied Memetic Algorithms.....	4
II.II Algorithmic Music Composition.....	5
III. GENERAL MEMETIC ALGORITHMS.....	6
IV. MUSIC COMPOSITION.....	8
V. A MEMETIC ALGORITHM FOR COMPOSITION.....	9
V.I Initialization.....	10
V.II Stopping Conditions.....	12
V.III Local Search.....	12
V.III Crossover and Mutation.....	14
V.IV Population Regeneration.....	15
V.V Evaluation.....	15
VI. CONCLUSION.....	16
VII. REFERENCES.....	17

## LIST OF FIGURES

Figure 1: Memetic Algorithm.....	11
Figure 2: Local Search Function.....	15

## LIST OF TABLES

Table 1: Note Value Representations.....	13
Table 2: A Chromosome's Note and Duration Values.....	13
Table 3: A Chromosome's Converted Note and Duration Values.....	13

## **ABSTRACT**

Music has been an integral part of society for hundreds of years. Since the advent and rise of computers and computer technologies, musicians have utilized these advances in their craft, be it in electronic music, or the notation of musical scores. The field of memetic computing is a new and developing one. Research is being done in applying memetic algorithms to real world problems. This research will offer a novel approach for a memetic algorithm to produce quality musical compositions. At the time of writing this paper, there were no proposals found in any literature for the use of memetic algorithms to compose music.

## I. INTRODUCTION

Music composition systems have been around for many years, especially in the form of expert systems. As computer technology advances, composition systems have become increasingly more powerful and complex. For computer-assisted music composition, the majority of the software that falls under this umbrella acts mostly as a tool to assist with visualizing composition, such as PatchWork or OpenMusic, and not as software that composes music directly [ASS99]. There are expert systems and genetic algorithms in existence that self-create works of music that are based on the analysis of a user's input or the creation of the music from scratch [ARI 05]. An increasingly popular method of composition using algorithms is that of an interactive genetic algorithm [NAK09][UNE01]. This involves using a genetic algorithm to produce sections of music, which are then represented back to the user in either graphical or aural form, allowing the user to evaluate the 'goodness' of the piece, until a predetermined number of 'good' sections are available. Genetic algorithms are synonymous with evolutionary algorithms; these are algorithms which are developed and designed using natural evolution and biology as inspiration, and have been used with great success in many real world optimization applications [NGU07].

While the concept of memetics itself has been in existence for years, memetic computing is a relatively new field of study for computer scientists. 'Meme's have been defined in many various ways, coined originally by Dawkins [DAW76], and essentially can be summarized as units of social information upon which we construct our culture [REA01][NGU07]. Memetic algorithms represent the next step in working with genetic algorithms. A memetic algorithm can be seen as an amalgamation of an evolutionary algorithm and individualized improvement

procedures [NGU07]. The purpose behind memetic algorithms is to improve efficiency in optimal-solution convergence; studies have shown that memetic algorithms do this more efficiently than heuristic or genetic algorithm approaches [NGU09].

This paper presents an effective memetic algorithm for music composition that builds upon existing expert system and genetic algorithm based approaches to the problem. The following paper is organized as follows. Section II presents related works in the field of applied memetic algorithms and music composition. Described are real-world applications of memetic algorithms and pre-memetic algorithmic music composition systems. Section III presents a definition of a memetic algorithm and discusses the various aspects involved in their design. Section IV explains some basic rules of composition in Western music theory. Section V presents the proposed memetic algorithm for music generation. Section VI offers a conclusion to the paper.

## **II. RELATED WORK**

In this section, review on the research done into papers and literature concerning applied memetic algorithms and music composition systems built upon artificial intelligence methods and other algorithms can be found.

## II.1 Applied Memetic Algorithms

In the last few years, research in the field of memetic algorithms has led to the development of algorithms for solving various problems. Several authors [TIN10, FER10, KON07] have focused on creating memetic algorithms for improving and optimizing the lifetime of wireless sensor networks. In [TIN10], the authors deem that the use of heuristic algorithms is too inefficient due to the trade-off between solution quality and run time. The paper also discusses the use of a genetic algorithm, which solves for near optimal solutions in acceptable time, but suffers from the dependency of needing information beforehand that is implausible to obtain. Their solution is a memetic algorithm which uses a Darwinian evolutionary algorithm combined with a Lamarckian localized improvement algorithm to find the optimal solution. The Lamarckian theory states that a parent can pass knowledge or characteristics that they obtain during their life to their offspring. Authors Lui [LUI08] and Bontoux, et. al [BON10], work with memetic algorithms in an attempt to solve large scale traveling-salesman problems. Lui [LUI08] utilizes the ‘nearest neighbor’ algorithm that is already commonly used to solve large scale traveling salesman problems, and combines it with localized search and population recombination procedures. The main focus of [BON10] is the localized search crossover procedure, with a comparison of results of their proposed crossover algorithm against four crossover techniques used in memetic algorithms. Samanlioglu, et. al [SAM08]., took the traveling-salesman problem a step further and proposed a memetic algorithm for solving multi-objective traveling salesman problems; effectively to optimize multiple objectives, such as time, distance traveled, cost, and others The authors propose an algorithm consisting of a random-keys genetic algorithm with a 2-opt local search feature. Further research in memetic algorithms was

done by Lu and Hao [LU10] in focusing on developing an algorithm for solving graph coloring problems by using an algorithm which builds upon a popular meta-heuristic algorithm currently employed in solving graph-coloring and applying a localized improvement feature.

## **II.II Algorithmic Music Composition**

Algorithmic composition for music has been around for a number of years, in the form of expert systems, heuristic algorithms, and, more recently, genetic algorithms. Pope [POP95] explains a history of fifteen years of computer-assisted composition in a paper written 15 years ago. In that time, music composition systems have moved even further. The papers of Dion [DIO06] and Chiu et al. [CHI06] propose music composition systems that work on the premise of deconstructing a user's input set of music examples and creating new works from discovered patterns and common musical structures. Nakamura et al [NAK09] proposes a system to produce music and lyrics based on the input by the user, and also utilize an interactive genetic algorithm in which each section of music produced is displayed to the user to receive input on its fitness, an idea proposed by Unehara and Onisawa [UNE01] as the optimal way to evaluate generated works. Ariza [ARI05] attempts to codify the lexicon and main descriptors of computer-assisted algorithmic composition systems. Marques et al [MAR00] propose the use of genetic and evolutionary algorithms for music composition, which are essential to the idea of developing a memetic algorithm for the same purpose. They also describe in detail the encoding process in which the chromosomes are created within the genetic algorithm. A number would be randomly generated between -128 and 127, which one can represent in a single byte, and a series of these

numbers would be placed together. Note values are represented by the values from -48 to 49, with 0 being the middle C, sustains are represented with values -99 to -49 and 48 to 99, and pauses are the values from -128 to -100 and 100 to 127. In [SHE08], the authors use a similar approach, but focus on generating a melody. The paper proposes following the MIDI standard of assigning middle C the value of 60. The authors decide to use a 2 octave range, with values ranging from 48 to 72. They also determine values for six musical durations: thirty-second notes (.125), semiquavers (.25), quavers (.5), crotchets (1), minims (2), and semibreves (4). In their algorithm, each chromosome is divided into two partitions; one for pitch, and one for duration.

### **III. GENERAL MEMETIC ALGORITHMS**

Memetic algorithms are a fairly recent development in the field of computer science. A memetic algorithm is a combination of a genetic algorithm with a localized improvement feature. Studies have shown that memetic algorithms tend to find high-quality solutions more efficiently than straightforward genetic algorithms or heuristic algorithms [NGU09]. Digalakis and Margaritis [DIG04] performed a comparison on various local search techniques for memetic algorithms, and came to the conclusion that population size is perhaps the most important parameter to control for efficiency in the algorithm. Another important factor in developing memetic algorithms is the balance in frequency and intensity of the evolution of the population and the individualized learning [NGU07]. In all problem specific instances of memetic algorithms researched, there can be found four main components; the initial generation of a population by use of a genetic algorithm, a localized search for evaluation and improvement of

individuals in the population, the crossover and mutation operations of parents to produce unique offspring, and the regeneration of the population to introduce the offspring and remove weak individuals [TIN10, LU10, FER10, LIU08, SAM08, BON10, KON07, DIG04, NGU07, NGU09]. The steps of a memetic algorithm are described in [TIN10]: The typically genetic algorithm [FER 10] produces an initial population whose individuals are termed ‘chromosomes’. The local search chooses two of these chromosomes to act as parents, usually by selecting on the basis of ‘fitness’ values. The crossover operation then creates offspring by using the data from the two parents and passing the best qualities on. These offspring are sometimes mutated and enhanced to improve their fitness for propagation. A survival procedure determines whether the offspring are fitter to survive than the existing members of the population; if so, the offspring are entered into the population as a replacement for the least fit individuals. This cycle continues until an optimal, or acceptable, solution is found [TIN10]. In [LIU08], the author gives more detail on parent selection methods: fitness-proportionate, elitism, and tournament selection. Fitness-proportionate simply weights more ‘fit’ solutions heavier, leading them to be more likely to be chosen to produce offspring. Tournament selection works by using the fitness-proportionate values and systematically weighting solutions against each other in a tournament format, until the two remaining ‘best’ parents are left. These are left to crossover and produce offspring. In the elitism format, the top N solutions are propagated to the next generation of the population. The remaining solutions are generated using the tournament selection method. In [LU10], the author makes a claim that the crossover component to memetic algorithms is typically the most important, as it produces the new and improved individuals to add to the population. The authors also differentiate between the two classifications of crossover operators; the assignment operator, and the partition operator. They propose that the partition operator acts

superiorly to the assignment operator, because the partitions allow for the differentiation of ‘good’ and ‘bad’ properties to pass to offspring. Their paper also explains that an offspring should be inserted into the population as a replacement for the ‘worst’ parent, and should not be identical, or too closely similar, to an existing individual in the population.

#### **IV. MUSIC COMPOSITION**

In accordance with Western music theory, a musical scale is a series of notes that start with a root, or tonic, note, and ascend in order by following a specific pattern in relation to the appropriate scale. The twelve notes in Western music are; A, A#, B, C, C#, D, D#, E, F, F#, G, and G#. These notes circle back onto themselves. G# would then move back to A. A half step is simply the distance, or interval, between a note and its immediate successor. A whole step is two half steps. Therefore, to build, for example, a C major scale, you begin with the tonic note, C. The notes then follow a pattern of whole and half steps as such; whole, whole, half, whole, whole, whole, half. We then end up with the series of notes C, D, E, F, G, A, B. These are the notes of a C major scale. Major scales always follow the same pattern of steps. There are, however, a number of other scales one can build using different patterns. For the sake of the first generation of this memetic algorithm, only major scales will be used.

The duration of notes are typically referred to as note values, but for the sake of clarity in this paper, they will be referred to as durations to prevent confusion. The various durations in music theory are as follows; a longa (four notes long), a breve (two notes long), a semibreve (one note long), a minim (half a note), a crotchet (quarter of a note), quaver (eighth), semiquaver

(sixteenth), demisemiquaver (thirty-second), hemidemisemiquaver (sixty-fourth), and quasihemidemisemiquaver (hundred twenty-eighth). These durations refer to how long the note is played for. This paper will deal with only whole notes (semibreve) to thirty-second notes (demisemi-).

Time signatures, or meter, are a representation of how many beats are in each measure, and what note duration constitutes one beat. The most commonly used time signature in Western music is that of 4/4. This means there are 4 beats in a measure (top numeral), and each beat is made of a quarter note, or a crotchet. As stated, 4/4 is the most commonly used time signature, and is referred to as ‘common time,’ and therefore, our memetic algorithm will work within this meter.

## V. A MEMETIC ALGORITHM FOR COMPOSITION

```
population P = initialization();
while(!stopping_conditions)
{
    chromosome C1, C2 = local_search(P);
    offspring O = crossover(C1, C2);
    mutate(O);
    P = regeneratePopulation(O);
    evaluate(O);
}
```

*Figure 1: Memetic Algorithm*

## V.I Initialization

This procedure is to generate an initial population from which to proceed with. It will generate  $N$  members of a population, all with random genetic information. In this algorithm, a member of the population, hereafter chromosome, will consist of a series of values that indicate the notes and duration of those notes for one musical bar. Middle C, considered the ‘middle’ note in music composition and is found in the middle of a piano or keyboard, will be designated as the value of 60, in keeping with the MIDI protocol standards, and the values will range within 2 octaves; the octave leading to middle C, the octave beginning at middle C, and the C above this octave. The integer values from 48 to 72 will represent all of the natural, sharp and flat notes within these octave ranges (see Table 1). Also, a 0 will indicate a rest, or no note being played. For randomization, each note will be equally likely to be chosen. The values for the duration of the note will be represented as follows; thirty-second notes (1), semiquavers (2), quavers (4), crotchets (8), minims (16), and semibreves (32). For randomization, durations will be assigned a weighted probability, with quavers, crotchets, and minims being more common than the other durations. In the event of a dotted note, which indicates the note be held again half the length of the original duration, the value will reflect 1.5x the original value assigned (e.g. a dotted quaver will be represented as 6). A chromosome will consist of two sets of values; one to hold the values of the notes, and one to hold the values of the durations (see Table 2). This algorithm, as stated earlier, will assume 4/4 time. This means that there will be the equivalent of a semibreve, or whole note, within each measure. The set of values for duration, therefore, must summate to be less than or equal to 32 in order to be fit. The number of values in the set for duration will

equal the number of values in the set for pitch, assuring that each pitch will have a duration assigned to it (E.g. A set of 4 note values will indicate a set of 4 note durations).

*Table 1. Note value representations*

<b>C</b>	<b>C#</b>	<b>D</b>	<b>D#</b>	<b>E</b>	<b>F</b>	<b>F#</b>	<b>G</b>	<b>G#</b>	<b>A</b>	<b>A#</b>	<b>B</b>
48	49	50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69	70	71
72											

*Table 2. A chromosome's note values and duration values.*

52	55	56	52	50	58
8	8	4	2	2	8

*Table 3. A chromosome's converted note values and duration values.*

E	G	G#	E	D	A#
1/4	1/4	1/8	1/16	1/16	1/4

## **V.II Stopping Conditions**

The stopping conditions chosen for the initial implementation will be whenever every chromosome conforms to a specified key (Cmaj, Amin, etc) and the melody within any given chromosome holds within one octave.

## **V.III Local Search**

The local search feature is that which defines a memetic algorithm apart from a genetic or evolutionary one. Its purpose is to enhance the quality of the solution by causing improvements in the individual chromosomes. The localized search for this algorithm will traverse through each chromosome in the population, and will improve the chromosome by taking the first note value in the series of note values that does not belong to the specified key, and shifting it up or down to the nearest value that does. It will also check to ensure that the summation of values for duration is equal to 32. In the event that it is greater than 32, a check for dotted values will take place. If one is found, it will be reassigned as an un-dotted duration. If not, the largest value for duration will be found, and the next largest will be substituted in its place. If the summation is less than 32, the difference will be found and assigned as the duration of a rest at the end of the measure. In the event that each note does belong to the specified key, it will swap two of the note values and two of the duration values within itself at random. It will also assign a fitness value to each chromosome; one point for each note out of the key, one point if the duration does not equal 32, and one point if the notes within the chromosome are not within one octave. Therefore, higher

values will equate with lower fitness. The two chromosomes with the lowest values are assigned as parents. In the event of a tie, they are chosen at random from the eligible chromosomes. The pseudocode can be found in Figure 2.

```
for each chromosome
{
    if summation of duration values != 32
    {
        if summation > 32
        {
            if dotted value exists
                undot value;
            else reduce largest value;
        }
        else
        {
            add new duration = to difference;
            add rest note value for this duration;
        }
    }
    for each note value
    {
        if note value does not belong in key
            shift value into key;
        else if all values in key
        {
            randomly swap two note values;
            randomly swap two duration values;
        }
    }
}
```

*Figure 2. Local Search function*

### *Fitness function*

$$f(P_k) = N + d(P_k) + o(P_k); \quad (1)$$

$P$  = population of chromosomes

$P_k$  = chromosome  $k$  in population

$N$  = number of notes out of key

$d(P_k) = 0$  when duration = 32, else = 1

$o(P_k) = 0$  when all notes within 1 octave, else = 1

### **V.III Crossover and Mutation**

The purpose of the crossover and mutation procedures is to generate new and improved individuals to repopulate the solution with. To do this, the procedure uses the results of the local search in finding the parents by comparing their fitness values.

The process for crossover in this memetic algorithm will be as follows. The child will inherit at random the duration values of either parent 1 or parent 2. Then the child will fill the number of available note values as determined by the number of duration values by randomly inheriting notes from both its parents. This way, the genetic information of each parent can be found in the child.

The process for mutation will follow as such after the crossover has finished. The duration values of the resulting child will be rearranged at random, so as to create a wholly new set of durations.

Once both procedures have completed, a unique individual will be created from the genetic information of both parents.

#### **V.IV Population Regeneration**

Repopulation occurs after the creation of a new offspring. The process goes as follows. The new offspring has its fitness value calculated, just as every other member of the population during the localized search. Then this fitness value is compared to all the members of the existing population. If the fitness value of the new individual is lower than the worst fitness value of an existing member (highest value equates with worst fitness), then the least fit member is removed from the population, and is replaced with the new individual.

#### **V.V Evaluation**

The evaluation procedure is used to determine whether the stopping conditions have been achieved. Each chromosome is evaluated individually to determine if it satisfies the stopping condition. If all chromosomes are found to satisfy the conditions, the algorithm completes, and the chromosomes are returned to the user. If any chromosome fails to satisfy the conditions, the algorithm continues to perform.

## **VI. CONCLUSION**

The research into the uses and designs of memetic algorithms is in its infancy. There are numerous real-world applications already being solved by the development of memetic computing. Algorithmic music composition is also a field that is growing. Many artificial intelligence techniques are already being employed in research to advance the field. This proposal is an attempt to put forward a novel use for memetic algorithms for algorithmic music composition, a use that, at the time of this paper, has no published research. The belief is that this research, by improving upon the evolutionary and genetic algorithms already implemented in music composition, can help to find better solutions.

## VII. REFERENCES

[ARI05] Ariza, C. "Navigating the Landscape of Computer-Aided Algorithmic Composition Systems: A Definition, Seven Descriptors, and a Lexicon of Systems and Research." In Proceedings of the International Computer Music Conference. San Francisco: International Computer Music Association. (2005) 765-772.

[ASS99] Assayag, G., Rueda, C., Laurson, M., Agon, C., and Delerue, O. Computer-Assisted Composition at IRCAM: From PatchWork to OpenMusic. *Comput. Music J.* 23, 3 (Sep. 1999), 59-72.

[BON10] Bontoux, B., Artigues, C., and Feillet, D. 2010. A Memetic Algorithm with a large neighborhood crossover operator for the Generalized Traveling Salesman Problem. *Comput. Oper. Res.* 37, 11 (Nov. 2010), 1844-1852.

[CHI06] Chiu, Shih-Chuan, Shan, Man-Kwan, Computer Music Composition Based on Discovered Music Patterns. Proc. of 2006 IEEE International Conference on Systems, Man, and Cybernetics, Taipei (2006)

[DAW76] Dawkins, R. *The Selfish Gene*, Oxford University Press: 1976

- [DIG04] J.G. Digalakis and K.G. Margaritis, Performance Comparison of Memetic Algorithms, *Journal of Applied Mathematics and Computation*, Elsevier Science, 158 (25): 237-252, 2004.
- [DIO06] Dion, John A. Automated Music Composition: An Expert Systems Approach. *Rivier College Online Academic Journal*, Vol. 2, 1 (Spring 2006)
- [FER10] Ferentinos, K. P. and Tsiligiridis, T. A. 2010. A memetic algorithm for optimal dynamic design of wireless sensor networks. *Comput. Commun.* 33, 2 (Feb. 2010), 250-258.
- [KON07] Konstantinidis, A., Yang, K., Chen, H., and Zhang, Q. 2007. Energy-aware topology control for wireless sensor networks using memetic algorithms. *Comput. Commun.* 30, 14-15 (Oct. 2007), 2753-2764.
- [LIU08] Lui, Y.-H. A Memetic Algorithm for the Probabilistic Traveling Salesman Problem. In: *Procs. 2008 IEEE Congress on Evolutionary Computation (CEC 2008)*, Hong Kong.(2008).
- [LU10] Zhipeng Lu, Jin-Kao Hao, A memetic algorithm for graph coloring, *European Journal of Operational Research*, Volume 203, Issue 1, 16 May 2010, pgs. 241-250
- [MAR00] M. Marques, V. Oliveira, S. Vieira, and A. C. Rosa, "Music composition using genetic evolutionary algorithms," in: *Proc. of 2000 Congress on Evolutionary Computation (CEC-2000)*, vol. 1, 2000, pp. 714–719.

[NAK09] Nakamura, Chisa, Onisawa, Takehisa. Music/Lyrics Composition System Considering User's Image and Music Genre. Proc. of 2009 IEEE International Conference on Systems, Man, and Cybernetics, San Antonio (2009)

[NGU09] Nguyen, Q. H., Ong, Y., and Lim, M. H. 2009. A probabilistic memetic framework. Trans. Evol. Comp 13, 3 (Jun. 2009), 604-623.

[NGU07] Q. H. Nguyen, Y. S. Ong, and N. Krasnogor, "A Study on the Design Issues of Memetic Algorithm", *IEEE Congress on Evolutionary Computation*, 2007.

[POP95] Pope, S. T. Fifteen years of computer-assisted composition. Computer Music Journal, 1995.

[REA01] Reader, S.M. The nature of the memetic beast. *Darwinizing Culture: The Status of Memetics as a Science* (book review), Trends in Cognitive Sciences, Vol. 5, p.365-366, Elsevier: 2001

[TIN10] Ting, C. and Liao, C. 2010. A memetic algorithm for extending wireless sensor network lifetime. Inf. Sci. 180, 24 (Dec. 2010), 4818-4833.

[SAM08] Samanlioglu, F., Ferrell, W. G., and Kurz, M. E. 2008. A memetic random-key genetic algorithm for a symmetric multi-objective traveling salesman problem. *Comput. Ind. Eng.* 55, 2 (Sep. 2008), 439-449.

[SHE08] Sheikholharam, P. and Teshnehlal, M. 2008. Music Composition Using Combination of Genetic Algorithms and Kohonen Grammar. In *Proceedings of the 2008 international Symposium on Computational intelligence and Design - Volume 01*(October 17 - 18, 2008). ISCID. IEEE Computer Society, Washington, DC, 255-260.

[UNE01] Unehara M, Onisawa T, Composition of Music Using Human Evaluation. Proc. of 2001 IEEE International Conference on Fuzzy Systems, Melbourne (2001)