Yugoslav Journal of Operations Research
Volume 20 (2010), Number 1, 157-177
10.2298/YJOR1001157M

A GENETIC ALGORITHM FOR COMPOSING MUSIC

Dragan MATIĆ
Faculty of Natural Sciences
University of Banjaluka, Bosnia and Herzegovina,
matic.dragan@gmail.com

Received: September 2009 / Accepted: April 2010

Abstract: In this paper, a genetic algorithm for making music compositions is presented. Position based representation of rhythm and relative representation of pitches, based on measuring relation from starting pitch, allow for a flexible and robust way for encoding music compositions. This approach includes a pre-defined rhythm applied to initial population, giving good starting solutions. Modified genetic operators enable significantly changing scheduling of pitches and breaks, which can restore good genetic material and prevent from premature convergence in bad suboptimal solutions. Beside main principles of the algorithm and methodology of development, in this paper the analysis of solutions in general is also presented, as well as the analysis of the obtained solutions in relation to the key parameters. Some solutions are presented in the musical score.

Keywords: Music generation, evolutionary approach, combinatorial optimization, algorithm composing.

1. INTRODUCTION

Algorithms in music are used when the implementation of a set of rules or instructions can lead to adequate solutions. We can use algorithms for sound synthesis, sampling, recognition of musical works, as well as for music composition. The first three activities naturally impose algorithms as a way of solving the problem (searching the trees, series or disordered structures, and strict application of rules that describe the steps of the algorithm). In music composition, algorithms attempt to replace what so far has been considered to fall into the exclusive domain of human activity. Composing, as well as any other artistic activity includes free choice (of tones) by which a composer expresses his feelings, moods, intentions or inspiration. Proponents of algorithmic music consider that the free choice of the prescribed rules may be relatively easy to interpret as the relevant series of instructions. Most composers apply certain rules when composition, i.e. series or sets of instructions, and thus any composing process in some way can be considered as algorithm. On the other hand, the lack of human factors in the (automatic) algorithmic composition leads to the appearance of large amounts of objectively bad and useless music.

Therefore, many proponents of algorithmic composition decide to, during the execution of the algorithm, include human factors in determining the quality of the compositions. This kind of composition is called interactive composing, whereby, in a critical moment for assessing quality of composition (or its part), human opinion is involved. Sometimes, it can be shown that this approach often gives better results in comparison to the automatic composition, due to the fact that even a large number of rules and restrictions in algorithms cannot be good enough to assess the quality of the melody.

Genetic algorithms (GA) seems to be a suitable approach for generating musical compositions. Combination of genetic operators (mutation, selection and crossover) in some way simulates the innovative process (as real composing is), enabling continuous "improvement" of the obtained results.

1.1. Music terminology

This section describes the basic definitions for music terms. They do not cover complete music terminology used in this paper and some very common and less important terms are not listed.

Pitch is a basic concept in music. Pitch can be considered as a subjective feeling that the human ear hears, but also as an objective value (for example, the frequency of an appropriate sound wave). There are relative and absolute pitch determination. The relative one is based on the determination of the height in relation to some initial tone (for example, the tone of D4 is higher than the tone of C4). The absolute one is the objective and constant value (for example, the frequency of the tone A4 is 440Hz).

Pitches are written as notes, which represent the European standard system of 12 equally distributed semitones. Semitone is the smallest practically usable space between the two tones. In 12 equally tempered scale, the standard ratio between two consecutive semitones is $\sqrt[12]{2}$. In the scale, we have seven basic pitches („c“, „d“, „e“, „f“, „g“, „a“, „b“) plus five additional („cis”, „dis“, „fis“, „gis“, „ais“). After note "b", note "c" with the frequency $2f$ comes again, where f is the frequency of the starting tone „c“.

An octave is the interval between one musical pitch and another with half or double its frequency. To distinguish each tone series, the corresponding number of notes added to the numerical indices, e.g. "C1", "C2", "c3", etc. An octave is, therefore, a series of eight tones (e.g. "c1-c2), consisting of twelve semitones.

Melody is represented by pitches arranged in a horizontal sequence, one sounding after another.

Pitch duration is also an essential part of any musical composition. The timing and length of each pitch in a melody defines that melody’s rhythm. Rhythm refers to timing, both in terms of how long sound events last and when they are scheduled to occur.

The system of organizing durations, which is now commonly used, is such that the first shorter duration of each tone is half of the previous one. Thus, the system notes the duration consists of geometrical progression with a quotient of two (whole notes, half notes, quarter, eighth, sixteenth notes, etc.). The rhythm is associated with the duration as the duration of pitches and pauses, and disposition of their occurrence. The duration of the tone and frequency of these durations in the melody defines rhythm and basic unit of measurement - bar. Usually, the music works are organized in a way that they have their own rhythm and tempo, but there are works in which the bar is not constant, as well as works that have no bars.

Each basic note can be increased for a semitone, where the prefix "-is", or symbol # is added or decreased (for a semitone - half of a degree), where the symbol ♭ is added (e.g. "CIS", "D#", "E♭", etc.). In standard diatonic scales, the increased note of "e" or "eis" is equal to the note "f", as well as the increased note of "b" is equal to „c“. Analogously, the decreased notes "f" and "c" are "e" and "b". When writing increased pitches we use the sharp sign (#), to write down decreased pitch we use the sign (♭), and to “abolish” them we use sign (♮). Also, there are other symbols for multiple increasing (decreasing).

Tonality is a system of notes in which specific hierarchical pitch relationships are based on a key "center" or tonic.

The distance between the two notes, either when they sound simultaneously or one after the other, is called the interval. They are classified to consonant intervals, sounding pleasant to the human ear, and dissonant intervals, creating a subjective feeling of tension during the hearing. In the standard European system eight intervals are defined, between eight (plus one) of basic notes in octave, unison (also called prime), second, third, fourth, fifth, sixth, seventh and octave. Unison interval is trivial, because it applies the same tone. Intervals are further classified into:

- perfect, which occur in only one size of the spacing between tones (with one exception)
- minor and major, which often occur equally in two different sizes for a half degree,
- augmented and diminished which are different from perfect intervals for a half degree.

In Table 1 intervals and their size in semitones are listed.

Table 1: Overview of the intervals between the tones

Interval	Interval size	Name
unison	0	perfect
second	½	minor
	1	major
third	1½	minor
	2	major
fourth	2½	perfect
	3	augmented
fifth	3	diminished
	3½	perfect
sixth	4	minor
	4½	major
seventh	5	minor
	5½	major
octave	6	perfect

1.2. Genetic Algorithms

GAs are complex and adaptive algorithms usually used in solving robust optimization problems. Basically, they involve working with population of individuals where each individual represents a potential (optimal) solution, and each population is a subset of the total search space. Population in the iterative process is changing (old individuals are changing to new, potentially better ones).

Each individual is assigned a value called fitness, which indicates the quality of the observed individual. During the iteration process, good individuals are selected to (re)produce better ones, while applying genetic operators crossover and mutation. Old generation (in some way) is replaced by a new one. Detailed description of GA is out of this paper's scope and can be found in [7,23,30].

Some recent works in GA on various optimization problems show that GA often produces high quality solutions in a reasonable time [16-19].

In general, each individual is represented by a genetic code on some finite alphabet. In the wide use of GAs, usually binary coding is used, where genetic code consists of bit sequence. Number of individuals in the whole population is usually between 10 and 200.

The starting population is generated either randomly or by some other heuristic method where the only prerequisite to the usage of the second method is to be relatively fast.

1.3. Existing work of genetic algorithms in composing music

The first published record of the use of genetic algorithms (GA) for music composition is [11]. In the following years, GA has been widely used in this field by many researchers, and their works fall between music, mathematics and computer science. Description of all contributions in this area is out of this paper's scope and surveys can be found in [3,4,6,8,9]. A survey of the usage of different AI methods for algorithmic composition was made in [27].

Among many recent works, several directions of GA application for composing music melodies can be identified. In often cases, short and monophonic melodic fragments or motifs are composed, which typically range from one to eight or so bars in length. Some directions are:

- Making variations on existing composition or motif, [13,14,29];
- Making compositions similar to reference one, [10,22];
- Making solos or improvised melodies over or by existing templates (proposed rhythm and schedule of chords), [13,14,25];
- Considering both melody and rhythm: concurrently, [1,14,20], or separately, [28];
- Considering only melody composition without rhythm [15,29], or only rhythm generation without melody [5,12,31];

The interactive GA approach, where human opinion is used for evaluating the quality of the composition can be seen in [13,14,24,31]. One of the most famous software for generating music using interactive GA is GenJam, described in [1]. Meanwhile, various upgrades have been made on this software, last presented in [2]. The main two drawbacks associated with all interactive GAs are subjectivity and efficiency problem, referred to as "the fitness bottleneck", where the user must hear each potential solution in order to evaluate its quality.

Automatic calculating of the quality of the composition eliminates direct influence of the human factor, but involves two additional processes: a mapping of compositional rules to a numerical model, which is suitable for automatic optimization and re-mapping from the numerical optimization result to a musical. Among others, GAs using automatic calculating can be found in [10,22,25,26].

Current trends of GA applications to music are also described in [21]. In this book, some tools with computer simulations for creating and studying these systems are also presented: GenDesh, GenJam and CAMUS.

2. GA IMPLEMENTATION

Before the detailed analysis of the algorithm is performed, the aim and the basic idea should be stated:

1. The aim of the algorithm is to compose relatively short compositions (e.g. four 4/4 bars).
2. Compositions are represented by one array (of numbers) that carries information about the pitches and their duration.
3. The general input parameters determine: the length of the composition, tonality, number and range of tones allowed, the number of iterations, criteria for the completion of the algorithm, the method of interpretation of the results of the algorithm and so on.
4. The input parameters that affect the quality assessment of the composition are: the values that indicate the similarity of the composition with the referred

composition of the baseline (or reference values), the values of the intervals, the set of the "good" and "bad" tones, allowed deviation (variance) of the prescribed reference values, and weight factors that influence the importance of different assessment criteria

5. An important part of the algorithm is to establish criteria that determine the quality of the composition. These criteria are related to the evaluation of the intervals between successive tones, the deviation from the reference values and number of “bad” tones.
6. The composition search space is being searched by the principles of GAs in order to find composition which is “good enough”. It starts from the set of randomly generated individuals (compositions). This process of generating random composition is partially controlled by input parameters. Applying GA operators, from iteration to iteration, the algorithm tries to find the individual which meets the criteria to stop the iteration process. Algorithm stops either when it reaches the maximum number of iterations, or when the (best) individual is formed with good enough fitness. The quality of the individual is reversed in relation to the size of the fitness. The individual becomes "better" as its fitness (considering as number) decreases.
7. The fitness of all individuals of the population is computed in each iteration and new individuals are created by mutations of currently best ones. Then, the selection is performed among all new individuals and the individuals from the previous generation.
8. Output data from the algorithm is a composition, which, depending on the preset parameters and iteration process is considered as optimal.

The algorithm is implemented in the Java programming language.

The output of the algorithm is a music record, which can produce some of the standard musical outputs. JCreator (http://www.jcreator.com) is used for writing source code and compiling.

As the musical interface (for production audio files) JFugue (http://www.jfugue.org) Java API is used and for creation notation, Notation Musician (http://www.notation.com).

2.1. Population initialization and algorithm flow

In the algorithm the initial population is formed, containing individuals which have predefined rhythm, similar to the reference individual (distribution of beats at each individual is exactly the same as the reference, and possible "disorder" in the rhythm may arise due to breaks, generated in different places). Each individual is a complete composition. Fitness function is calculated for each individual, and population is sorted by fitness.

Usage of reference individual is optional and may be considered useful and practical if we would prefer that our composition has a distinctive rhythm (the schedule length of notes and pauses), or, as often the case, if we do not want the duration of notes to depend on random generator (it is much more likely that random generation would result in quite an irregular and awkward rhythm). In addition, the reference individual can have an impact on fitness, if predefined values (of intervals and their schedule) refer to that individual. In other cases, these predefined values can be entered independently.

The main elements of the algorithm are presented in Figure 4. Based on the initial parameters, the initial population is generated containing a total of *n* individuals. After this, an iterative process begins. Fitness is calculated in each iteration for each individual of the current population. After this, the list of individuals of the population is sorted by fitness. Based on the best individual (individual with best fitness), it is examined whether the condition is met for the end of the algorithm. If so, the algorithm stops and the corresponding best individual (composition) is pronounced as the result of execution of the algorithm. If not, the algorithm enters into the process of creating new individuals. Of all the individuals of the current population, the best individuals are chosen (namely, one-third of the total). Then, mutation operators are applied on them, thus obtaining new individuals. Each new individual is then added to the old list of individuals. After applying the mutations on selected individuals, the new list of individuals is re-sorted (by fitness). After that, duplicates (individual with the same fitness) are removed, and then the "excess" individuals are removed, in order to remain exactly *n* individuals. Iterative process is repeated until it fulfills the criteria for termination – the best individual has good enough fitness, or when the algorithm reaches the maximum number of iterations.

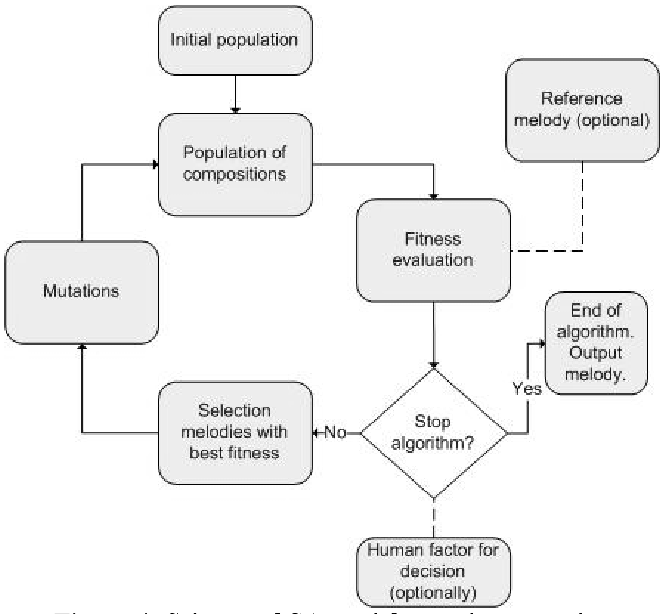
Initial population
Reference melody (optional)
Population of compositions
Fitness evaluation
Mutations
End of algorithm. Output melody.
Yes
Selection melodies with best fitness
No
Stop algorithm?
Human factor for decision (optionally)

Figure 4: Scheme of GA used for music composing

2.2. Creating an individual

The system of representation of an individual is as follows:

Let us assume that the set of allowed tones is a subset of standard diatonic set (each tone can be played with the appropriate piano key). Then, let us choose relative representation of the tones and let the total number of pitches be *n*. We should assign number 1 to the reference pitch, to the following (in height) - tone number 2, next one, 3, etc. Further, let the greatest common divisor of the durations be *k*. Let us call it „the shortest length". Also, let the whole composition consists of *m* bars, each bar of the *p* pulses. Let one pulse have *q* „shortest lengths". From here we conclude and state:

1. every bar has *pq* „shortest lengths";
2. any tone duration is of *tk* „shortest lengths", for some *t*;
3. whole composition is of *mpq* „shortest lengths";
4. a break with „shortest length" is represented by the number 0;
5. the shortest length is represented by the number of *n*+1;
6. in order to represent the whole composition, it is enough to use one array of the numbers, with the length of *mpq*, where all elements are from [0,*n*+1]. If the element is from [1,*n*], it is (real) tone with appropriate pitch, if the element is 0, it is a break, and if the element is equal to *n*+1, it means that the duration of the first preceding tone (or break) to the left is increased by one „ the shortest length";
7. each composition that satisfies these conditions can be assigned one and only one series;
8. Any series, except those that begin with the number of *n* +1 (we do not know what tone is of „shortest length") corresponds to exactly one composition.

With this system a relatively simple representation of simple compositions is achieved, while for more a complex composition this system can be used with some improvements. For example, the basic setting does not allow presentation of multiple tones at one time, and practically, for each such situation we must take more than one series. Furthermore, such a system, although theoretically possible, is not practical for representation of polyrhythmic compositions, i.e., those that have a wide range of different durations (for example, if, in addition to the usual duration of the fourths, eighths, sixteenths, also exists durations of the thirds, fifths or sixths).

Example 1.

Let us see how such a representation can be applied to the concrete composition. In Figure 1, one composition is represented by musical notation and appropriate series.

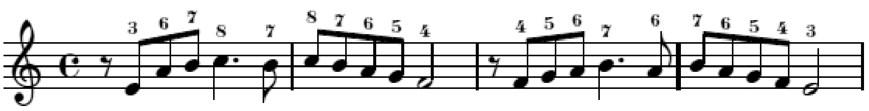

Figure 1: Representation of notes in a composition

Let the tone of C4 be selected for a reference pitch (composition is written in *a* minor). Let two octaves be available for tones. Tones that do not belong to the C major scale (i.e. *a* minor) are not considered (in this example), and for them there is no adequate representation. The numbers above the notes indicate the distance from the reference tone.

We have a total of 14 different tones and we can choose for the representation shown in Figure 2. We see that the number of zero represents the break.

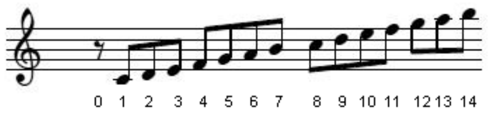

Figure 2: Coding in C major scale

It is now necessary to introduce the duration of tones. Based on the composition (Figure 1) the following facts are noted: Time signature of the composition is 4/4. Since the total number of allowed tones is 14, all the elements of array are from the interval [0,15], where zero indicates a break, and number 15 we use to add one "shortest duration" of the previous note. The greatest common divisor of all durations is eighth. Therefore, for the shortest length we use one eighth (of beat). Given that the composition has a total of four bars, in each bar we have eight of the shortest lengths, for the presentation of this composition, we need a series of length 32. Break (that is length of one eights) is represented by zero, each tone is represented by a number that represents the duration of one eighth. Any longer duration is indicated by the number of 15. Therefore, the first tone C, which occurs in the composition, lasts three eighths, and is represented by 8 15 15. The whole series is as presented in Table 2:

Table 2: Coding of the composition shown on Figure 3

Indexes and values of the elements of series																															
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0	3	6	7	8	15	15	7	8	7	6	5	4	15	15	15	0	4	5	6	7	15	15	6	7	6	5	4	3	15	15	15

Distribution of numbers and tones are shown on Figure 3.

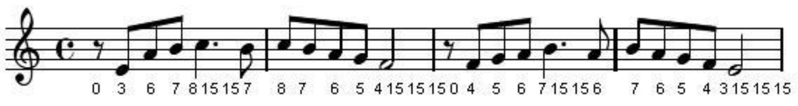

Figure 3: Coding composition with breaks and different durations

Initially, a reference individual is chosen, which determines the general parameters: size, tonality, number and a list of allowed (half) tones, the overall duration of an individual (the number of beats or bars), the shortest length (greatest common divisor of all durations), the number of the shortest lengths in one beat, as well as the distribution of beats in individuals.

Each individual (array) is generated in an arbitrary way, with two restrictions (*n* is total number of pitches):

- all elements of the series are from [0,n+1],

- *i* –th element is equal to *n*+1 (meaning prolongation) if and only if the appropriate element of the series of the reference individual is equal to *n*+1. With this feature, we hold the same rhythm for individuals.

2.3. Determining fitness

The fitness function is used to determine the criterion for comparison of quality of individuals. Determining the fitness function in the theory of GAs is often a critical point in the design of the algorithm. Here, we must take into account the additional parameter, that music is a subjective sensory event (for instance, what one person likes, may not be pleasant to others, and otherwise). Therefore, however the fitness is computed, the possibility of subjective opinions about the quality of the individual still remains. It is clear that the determination of fitness function of GA is the most important but also the most complicated single step. According to the current state of the art, a reliable and efficient way to determine the fitness function that will directly refer to the desired solution is not yet defined [33]. In most cases, a function which computes the total fitness based on different criteria is used. List of potential measurable musical elements in the composition is given in [32].

Thus, the total fitness *f* is defined as

$$f = \sum_{i=1}^{n} \lambda_i f_i \tag{1}$$

where λ_i represents the weight (influence) of the value f_i to the total fitness, and *n* is the total number of criteria. For example, for different *i*, f_i may be a ratio between the number of tones out of a given tonality and the total number of tones, the ratio between the number of dissonant intervals and all intervals, the ratio between the number (or total) appearances of some pattern in relation to the total number of notes, density of tones etc. Parameters λ_i give appropriate weight to the value.

In [22], a more general approach is used, where fitness is calculated from one to another bar, and the total fitness is the sum of those values. This approach is also used in algorithm presented in this paper.

Therefore, the total fitness is calculated as

$$f = \sum_{j=1}^{k} \sum_{i=1}^{n} \lambda_{ij} f_i \tag{2}$$

where λ_{ij} is weighted factor of value f_i u *j*-th bar, *n* is the total number of criteria, and *k* is the total number of bars.

As we have a reference individual (or reference values), determination of fitness is (not entirely) related to the assessment of how our individual „looks like" the reference one. In addition, given that all semitones from the observed interval are allowed, it is possible that, while generating individuals we get „good“ intervals, but with tones that do not belong to the desired tonality. It is therefore necessary that the final fitness value is affected by the number of tones out of tonality. The quality of an individual is inversely proportional to the number. Therefore, tones out of a given tonality are allowed, but the individual is still better as it has less of those tones.

The similarity with the reference individual is determined on the basis of the defined "distance" of an individual to the reference one. The distance is calculated bar by bar. Roughly speaking, the distance between individuals, and appropriate bars is based on the number and type of "good" intervals, as well as their distribution by bars. In the case that the reference individual is not used, the parameters that affect the comparison must be "manually" defined. From the mathematical perspective, the similarity is based on determining the arithmetic mean value of the intervals in the bar and the corresponding variance of the two compositions, for each bar. After that, differences between the corresponding values are considered, which are then gathered together (with possibly some weight multiplication factors). The process of determining the fitness is as follows:

Determining the (number) values of each note. According to the system of representing the composition, each note corresponds to the appropriate number, i.e. the distance from the reference note.

Determination and evaluation of the interval (in bar). Interval consists of two consecutive notes (breaks are skipped). If we observe the appropriate series, all intervals are the subtractions between the two consecutive elements which are different from zero and the total length (which does not denote a note, but the extension period). Thus, in relation to the total number of notes, there is one interval less, in the first bar. Intervals that are "on the border" between two bars are tied for the second tone of the interval, i.e., the second of the two bars. The rule for evaluation of the intervals is carried out by the "quality" of intervals, giving the lower value to the „better“ intervals. Table 3 gives two proposals for evaluating intervals. It should be noted that, due to the functioning of the algorithm (computing fitness function), the lower value of the interval actually says that that interval is „better“. Examples of evaluation of intervals are given in the last two columns of Table 3.

Table 3: Proposals for evaluation of intervals

Categories of intervals	Intervals	Values (proposals)	
		I proposal	II proposal
perfect consonants	unison, perfect fourth, perfect fifth, octave	1	1
imperfect consonants	minor and major thirds and sixths	2	3
seconds	minor and major seconds	3	1
sevenths	minor and major sevenths	3	3
intervals greater than octave	all intervals greater than octave	5	5

Determine the arithmetic mean and variance. Arithmetic mean and variance are calculated for each bar. Arithmetic mean is the average value of the interval values that are present in the bar.

$$a = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (3)$$

where x_i is value of i-th interval, n is the total number of intervals (in the bar). For example, according to data from Table 3, if all intervals in the bar are perfect consonants, the arithmetic mean is equal to 1.

Variance is calculated as the mean of sum of squares of all deviations in the interval, from arithmetic mean, given by formula:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - a)^2 \tag{4}$$

From this formula we see that the variance is greater when we have more „different“ types of intervals.

Therefore, these values are calculated for each bar of the reference and observed individuals. Information about the similarities between these two individuals are given by formulas,

$$f_1 = \sum_{i=1}^{m}\zeta_i(\mu_i - a_i) \tag{5}$$

$$f_2 = \sum_{i=1}^{m}\eta_i(\sigma_i^2 - b_i^2) \tag{6}$$

where ζ_{i} is influence of the difference of arithmetic means in i-th bar, μ_{i} is the arithmetic mean of i- th bar of reference melody (or predefined value if reference individual is not used), a_i is arithmetic mean of i- th bar of arbitrary individual, η_i is influence of i –th deviation, σ_i^2 variance of i-th bar of reference individual, (again if we do not use it, it is predefined value), and b_i^2 variance of i-th bar of arbitrary composition. The number m is the total number of bars. An opportunity for (manually) setting the values of ζ_i and η_i for any bar, gives the possibility of „balancing“ intervals in the melody.

For example, at the beginning and at the end of the composition, lower values can be given to these numbers, and greater in the middle, which means that at the beginning and the end we emphasize the similarity, with the reference individual (or pre-defined values). In the examples presented in this paper, all values weight factors are equal to one.

Total similarity is defined as $f = \alpha f_1 + \beta f_2$.

α and β are global weighted factors. In the examples in this paper, both factors are equal to one.

It is clear that if the reference and arbitrary individual are the same, the value of f will be zero. The opposite is not true, the value of f can be zero if individuals are not equal. It justifies that the usage of reference individual is optional. What this information suggests, then, is that individuals, from bar to bar, have a similar (or same) distribution of intervals with the same given value.

Furthermore, in the algorithm an additional factor that affects the fitness is considered: the number of tones that are outside of the prescribed tonality. In general, this algorithm uses a set of “bed” tones, where the total number of „bad“ tones are counted. Breaks are ignored (considered as "good" tones). Thus we get the value:

$$g = \gamma \frac{1}{bl} \tag{7}$$

where bl is the total number of „bad“ tones. γ is weighted factor , in this solution, it is equal to one.

Thus, the total fitness of an individual is calculated by the formula:

$$TotalFitnes = f + g = \alpha \sum_{i=1}^{m} \zeta_i (\mu_i - a_i) + \beta \sum_{i=1}^{m} \eta_i (\sigma_i^2 - b_i^2) + \gamma \frac{1}{bl} \tag{8}$$

2.4. Genetic operators

In the algorithm three types of mutation and selection are used, while crossover omitted. The reasons for the lack of crossover operator are:

- the algorithm is to generate relatively short compositions and it makes no sense to crossover so short pieces;
- using three types of mutations and good balancing parameters that affect the fitness attained adequate results (not always, but in many cases algorithm generated individual with fitness equal to zero) and crossover (or any other operator) cannot further optimize already the optimal solution;
- Obtained best individuals represent good "samples" to create a new larger (longer) composition and the upgrade of this algorithm should go in the direction of the crossing over whole individuals within these longer compositions. This idea is out of the scope of this paper;
- Since the goal is not to develop a fast algorithm, but one which can identify an individual which is good enough, for each generation the possibility of generating a huge number of individuals has been left, of which a very large number of these are abandoned. In this manner, we prefer exploitation of the set of all individuals rather than optimization.

According to the models used in the literature, three different mutations are implemented. Probability (relative to other mutations) of occurrence for each mutation is determined. Furthermore, there is a choice on what individuals (and how many times) mutation will be applied. Since there is no crossover operator, the idea is to apply mutations on better individuals multiple times. In this way, the good individuals are “striving” to become better. On the other hand, it is possible that the application of mutation does not change fitness at all (although the individual changes), so it is possible that different individuals with the same fitness appear. This problem is solved by the appropriate selection.

Mutation 1: Changing tone for an octave. This mutation potentially reduces the number of "large intervals", i.e. those that are larger than one octave, which in a standard algorithm setting are given very high value (they are considered as „bad“ intervals).

Mutation 2: Changing one tone. This mutation allows the "correction" of the fitness of the old individual, in the case when the tone which is not in a "harmonious" relationship with its neighbors changes. In this case, with substitution to some other tone, there is a chance to improve fitness. According to the functioning of the selection, "distortion" of fitness (getting worse in the new individual) does not affect the overall quality, because in this case the old individuals will survive.

Mutation 3: Swapping two consecutive notes. The index of the note is chosen randomly and the note swaps with the neighboring note. This mutation can improve fitness by changing and potentially correcting the "surrounding intervals".

Selection plays an important role, given that a large number of new individuals is generated in each iteration. The elimination selection is used (individuals who have low fitness are removed), along with additional elements: before removing poor individuals, potential duplicates are removed, and of all individuals who have the same fitness (this can occur by applying appropriate mutations) only one copy is left. Furthermore, if the defined number of iterations, runs the best individual that has no satisfactory fitness (not good enough), then it is also removed, and the second one becomes the current best individual. Experiments show that this phenomenon usually happens in the case of an "unfortunate" definition of extremely poor initial population, where individuals are so bad that the mutations can not sufficiently improve them. On the other hand, the objective of the algorithm justifies and allows these effects and so it is not considered as error in the algorithm, but rather as "poor inspiration" of the random generator. Elitist strategy is not applied directly (with no pre-defined number of individuals that are going into the next generation), but the assumption is that the mutation operators can not decrease or increase the fitness in such a way that old outstanding individuals do not survive at least until the next generation. (Each mutation can change only two intervals.).

3. EXPERIMENTAL RESULTS

In this section the compositions obtained by variations of parameters are presented. By an analysis of the parameters and the obtained composition the conclusion is that results can be categorized into classes of „similar“ compositions.

Some compositions obtained by GA can be downloaded from http://www.pmfbl.org/matematika/zaposleni/dmatic/files/music.html.

Some of these compositions, especially the "mainstream" ones, sound pleasant. Comparison of the quality of the compositions can be done only for those represented by the same mathematical model. As it is hard to define the function which naturally determines the quality of a composition, there are large numbers of mathematical models, that are incompatible and (mathematically stated) incomplete. Since this model uses characteristics of various different models, direct comparison is not possible.

Tests have shown that the combination of a large number of different parameters can significantly affect the quality and the concept of melody. For example, giving lower value to minor and major seconds (compared to the others) we get the composition of which the successive tones (or intervals) are relatively close. Mostly, the situation when the lowest value is assigned to the perfect consonants is tested (they are considered as best intervals). In the opinion of the author, in this case, the best solutions are obtained.

In this solution, the author opted for the following limitations:

1. Tones are taken from two octaves. There is a possibility of defining a set of "good" or „bad“ tones. By default, the algorithm declares tones from G major scale as good, while the tones out of G major scale are considered as bad. This does not mean that they are completely excluded, but only that their appearance spoils the overall fitness.
2. Perfect consonants are given lower value than the other intervals. Interval values are identical to the values from Table 3 (I proposal)
3. It is chosen that the composition consists of four bars and a total of 32 shortest lengths. Thus, each bar has 8 shortest lengths.
4. The reference arithmetic means and deviations of each individual bar are defined. Depending on the defined means and deviations, we get different distribution of consonant and dissonant intervals. We get quite a nice solution when we require more perfect consonants in the first and fourth bars, while we allow freedom for the appearance of other intervals in the middle bars.
5. The algorithm was tested for a population size of several dozen (mostly 30) of individuals. It turned out that for obtaining good (and often optimal) solutions 100 generations are enough.
6. The solutions are series of tones with different durations, with rather frequent breaks. Generally, the algorithm seeks to produce breaks, because that reduces the potential bad intervals; the bed intervals have a greater impact on decreasing the quality of the individual, than the good ones have on increasing that quality. Hence, the obtained individuals sound more like good improvisations than melodic composition. Ultimately, they are too short in order to form a longer melody. Given that, the author has decided to present the results arranged in the basic arrangement, where the generated individuals are associated with slightly adjusted elementary chords and rhythm of drums.

3.1. Examples of „mainstream“ compositions

This section presents a combination of parameters which determine the best composition.

The interval values are shown in Table 4. The interval is „better“ as its value diminishes.

Table 4: Concrete values of the intervals in the mainstream compositions

Intervals	Values
unisons, perfect fourths and fifths, octaves	1
minor and major thirds and sixths	2
minor and major seconds and sevenths	3
intervals greater than octave and augmented fourths	5

From data from Table 4 we conclude:

- The perfect consonant intervals are the best, and
- Thirds and sixths are good enough that the probability that they will appear is relatively high
- seconds and sevenths are not welcome, and are likely to occur less than consonant intervals
- intervals larger than one octave are extremely undesirable.

For a set of "good" tones we declare the set of tones belonging to G major scale. Tones out of G major scale are considered as bad.

Since the composition consists of four bars, we define four reference values for the arithmetic mean of the interval and variance. The values are shown in Table 5. The algorithm combines data from Table 4 and Table 5 and so estimates the quality of the intervals.

Table 5: Reference values for arithmetic mean and variance

Reference values	Bars			
	I bar	II bar	III bar	IV bar
Arithmetic mean	1	1	1	1
Variance	0	0.2	0.2	0

Based on data from Table 5, we can conclude:

- Perfect consonant intervals are required for all four bars,
- Any deviation will happen before in the second and third bar, rather than in the first and fourth.

It should be repeated that such preferences do not exclude the occurrence of other intervals, but only reduces the probability of their occurrence.

All weighting factors that affect fitness are the same unit.

Figures 5-8 shows four individuals obtained under these conditions.

Figure 5: The first individual. Almost all intervals are perfect consonants

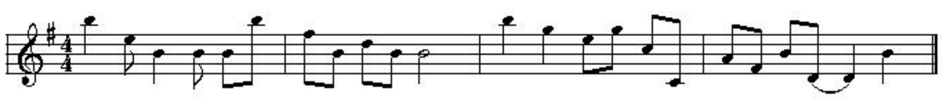

Figure 6: The second individual. Appearance of thirds and sixths

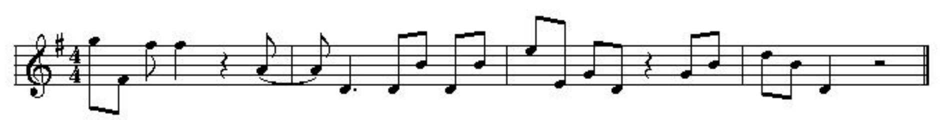

Figure 7: The third individual. A greater number of thirds and sixths in the second, third and fourth bar

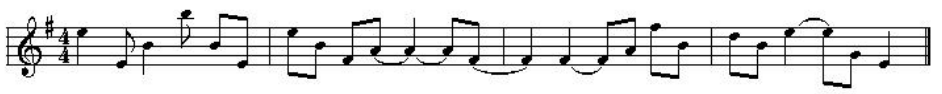

Figure 8: The fourth individual. Again, we have mostly perfect consonants

3.2. Special individuals

In this section we can see how the changing values of the intervals, as well as reference values for the mean of the interval and variance, can "manage" the composing process.

Individual 1.

Perfect consonant are the most desirable (table of interval values is identical to Table 4), and for the reference values we requested that the entire composition consists of the intervals with a value of 1 (Table 6).

Table 6: Reference values for arithmetic mean and variance

Reference values	I bar	II bar	III bar	IV bar
Arithmetic mean	1	1	1	1
Variance	0	0	0	0

Other parameters are the same as in the previous example.

Under these conditions, in 47th iteration, the algorithm determined the melody (shown in Figure 9) as the best result. The fitness of this composition is zero (optimal), because, in addition to all the intervals being optimal, composition does not contain any tone out of G major scale.

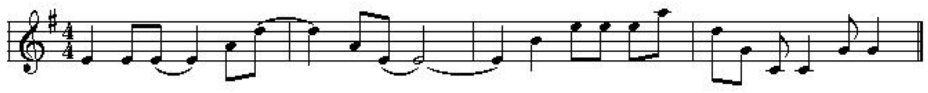

Figure 9: All intervals are unisons, perfect fourths and fifths.

Individual 2. In this example, seconds (minor and major) are declared as the best intervals. Interval values are shown in Table 7. Variances are equal to those of Table 6 (We do not allow deviations from the reference value). This indicates that the algorithm will seek to put all the intervals to those who have a value of 1.

Table 7: Seconds are best intervals

Intervals	Values
unisons, perfect fourths and fifths	2
sevenths and augmented fourths	4
minor and major thirds and sixths	3
minor and major seconds	1
all intervals greater than octave	5

In the 100th iteration, the algorithm brought out the melody shown in Figure 10. We see two interesting things: The algorithm aims to delete tones (composition contains a long break) and the „bad“ tone of Cis retained, which does not belong to G major scale. Therefore, the fitness of this composition is greater than zero and the algorithm is not terminated in earlier iterations (it performed the maximum number of iterations which was a criterion to stop the algorithm). The occurrence of the tone Cis affects the "deterioration" of fitness. Hence, we conclude that this individual could mutate into a "better" one only if mutation changed the tone Cis to C (any other tone would undermine the interval). The probability that this will happen is very small. Therefore, it is assumed that in additional number of iterations the fitness of that individual will not be better. Another possibility is that the individual "dies of young age", and the algorithm finds the optimal solution based on other individuals.

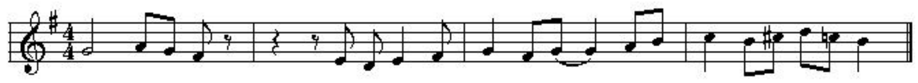

Figure 10: All intervals are minor and major seconds

Individual 3: For reference values we demand that the entire composition consists of thirds, sixths or octave. Interval values are shown in Table 8, and reference data are again the same as in Table 6

Table 8: Best intervals are thirds, sixths and octaves

Intervals	Values
unisons, perfect fourths and fifths	3
minor and major thirds and sixths, octave	1
minor and major seconds and sevenths	3
all intervals greater than octave and augmented fourths	5

In the 50th iteration, the algorithm gave the composition shown in Figure 15. We can see that all the intervals are thirds, sixths or octaves and there are no tones out of G major scale. This means that the fitness of this individual is zero.

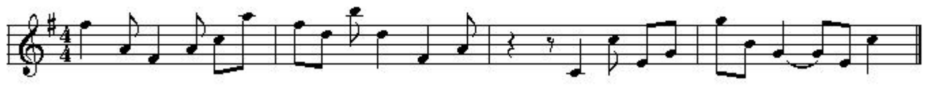

Figure 11: All intervals are thirds, sixths or octaves

Individual 4, 5 and 6 show that by increasing the allowed variance, step by step, we lose control over the tones.

Individual 4: If we favor minor and major seconds, and allow a relatively small variance (10%), the algorithm, (after some less successful attempts) brought out an individual shown in Figure 12. We see that allowing deviations reduces the probability that breaks will appear. Given that, fitness is not optimal, the algorithm was carried out "to the end", i.e. made a maximum 100 iterations.

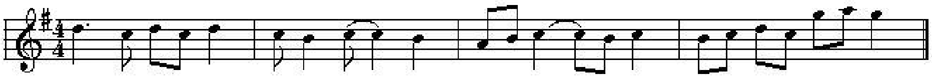

Figure 12: Greater deviance decreases probability that break will appear

Individual 5: If we allow a slightly larger deviation (we can consider deviance up to 30%), the algorithm results in the composition which is still „kept under control", although deviation allows greater freedom in the distribution of intervals. Still, a large number of the preferred intervals (large and small seconds) exists. The composition is shown in Figure 13.

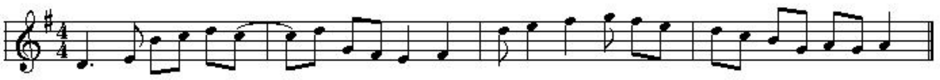

Figure 13: Greater deviance allows more freedom in intervals

Individual 6: If we allow a large deviation (practically we remove restrictions), keeping the values of the other parameters, we are given the composition that makes no sense at all. Here is listed only as a marginal case, which further justifies the control of parameters. The composition is shown in Figure 14.

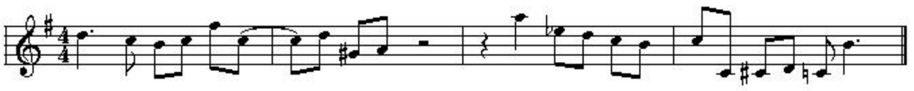

Figure 14: Deviance caused by large variance

4. CONCLUSION

A genetic approach for generating music compositions is presented in this paper. Results that can be obtained by the algorithm meet some objective criteria of "beautiful" compositions: they contain intervals that are pleasant to the human ear, the rhythm is meaningful, and, with a slight adjustment to the appropriate arrangement, the compositions sound unusual, but pleasant.

From a practical point of view, this algorithm gives the possibility to control the various parameters that affect the quality and form of the composition. The existence of reference individuals (or pre-defined parameters) improves the process of selecting and obtaining a relatively rhythmic and harmonious composition.

By coding the composition by an array of tones and breaks (with additional information about the length), an effective and quick control of the composition, tones and its rhythm is provided. This coding system enables the application of appropriate mathematical functions to tones, intervals and other "musical" parameters. It gives numerical values that can perform arithmetic and logical operations necessary for the operation of any algorithm.

This research can be extended in several ways. It would be interesting to implement some other metaheuristic for comparison or hybridization with GA. By adjusting parameters in an appropriate way, it can be investigated how presented GA could generate compositions that all belongs to one particular music gender.

REFERENCES

[1] Biles, J.A., "GenJam: A genetic algorithm for generating jazz solos", *In ICMC Proceedings 1994, The Computer Music Association*, 1994.

[2] Biles, J.A., "Improvizing with genetic algorithms: GenJam*", Evolutionary Computer Music* (Eduardo Reck Miranda and John Al Biles (Eds)), Springer, 2007.

[3] Biles, J.A., Evolutionary Computation for Musical Tasks, *Evolutionary Computer Music* (Eduardo Reck Miranda and John Al Biles (Eds)), Springer, 2007.

[4] Brown, A.R., "Opportunities for Evolutionary Music Composition", *Australasian Computer Music Conference*, Melbourne: ACMA, (2002) 27-34.

[5] Burton, A.R., *A Hybrid Neuro-Genetic Pattern Evolution System Applied to Musical Composition. PhD Thesis*, University of Surrey, School of Electronic Engineering, 1998.

[6] Burton, A.R., and Vladimirova, T., "Generation of musical sequences with genetic techniques", *Computer Music Journal* 23, (4) (1999) 59–73.

[7] Filipović, V., "*Selection and migration operators and Web services in parallel evolutionary algorithms" PhD thesis*, University of Belgrade, Faculty of Mathematics, 2006, (in Serbian)

[8] Gartland-Jones, A., "Can a Genetic Algorithm Think Like a Composer?", *Generative Art*, 2002.

[9] Gartland-Jones, A., and Copley, P., "The Suitability of Genetic Algorithms for Musical Composition", *Contemporary Music Review*, 22 (3) (2003) 43–55.

[10] Hochreiter, R., "Audible Convergence for Optimal Base Melody Extension with Statistical Genre-Specific Interval Distance Evaluation", *Lecture Notes in Computer Science,* 3907 (2006) 712-716.

[11] Horner, A., and Goldberg, D.E., "Genetic algorithms and computer-assisted music composition", *Proceedings of the 1991 International Computer Music Conference*, 1991, 479—482.

[12] Horowitz, D., "Generating rhythms with genetic algorithms", *Proceedings of the 1994 International Computer Music Conference, ICMA, San Francisc,*. 1994.

[13] Jacob, B.L., "Composing With Genetic Algorithms", *Proc. of the 1994 International Computer Music Conference*, 1995, 452-455.

[14] Jacob, B.L., "Algorithmic Composition as a Model of Creativity", *Organised Sound: (1)(3)*. Cambridge: Cambridge University Press, (1996) 157-165.

[15] Johanson, B., and Poli, R., Gp-music: An interactive genetic programming system for music generation with automated fitness raters, *Proceedings of the 3rd International Conference on Genetic Programming, GP'98*, MIT Press, Cambridge, MA, (1998).

[16] Kovacevic, J., "Hybrid genetic algorithm for solving the low-autocorrelation binary sequence problem", *Yugoslav Journal of Operations Research,* 2009.

[17] Kratica, J., Kovačević-Vujčić, V., and Čangalović, M., "Computing strong metric dimension of some special classes of graphs by genetic algorithms", *Yugoslav Journal of Operations Research*, 2005.

[18] Kratica, J., Kovačević-Vujčić, V., and Čangalović, M., "Computing the metric dimension of graphs by genetic algorithms", *Computational Optimization and Applications,* 36 (2009) 2149-2159.

[19] Kratica, J., Čangalović, M., and Kovačević-Vujčić, V., "Computing minimal doubly resolving sets of graphs", *Computers & Operations Research*, 2009.

[20] Marques, M., Oliveira, V., Vieira, S., and Rosa, A.C., "Music composition using genetic evolutionary algorithms", *Proceedings of the IEEE Conference on Evolutionary Computation 2000*. IEEE Press, New York, NY, 2000.

[21] Miranda, E.,R.,and Biles, J.A., (editors), *Evolutionary Computer Music*, Springer 2007.

[22] Mišljenčević, N., "Musical Notes", University of Zagreb, Faculty of Electrical Engineering and Computing, 2007, (in Croatian).

[23] Mitchell, M., *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, Massachusetts, 1999.

[24] Moroni, A., Manzolli, J., Von Zuben, F., and Gudwin, R., "Vox populi: An interactive evolutionary system for algorithmic music composition", *Leonardo Music Journal*, 10 (2000) 49-54.

[25] Özcan, E., and Erçal, T., *A Genetic Algorithm for Generating Improvized Music*, *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 4926, 2008.

[26] Papadopoulos, G., and Wiggins, G., "A Genetic Algorithm for the Generation of Jazz Melodies", *In: STeP 1998, Jyväskylä, Finland,*1998.

[27] Papadopoulos, G.; Wiggins, G., "AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospects", *In AISB Symposium on Musical Creativity*, 1999.

[28] Prerau, M., "On the possibilities of an analytic synthesis system", *Proceedings of the European Conference on Artificial Life 2001 Workshop: Artificial Life Models for Musical Applications,* Prague, Czech Republic, 2001.

[29] Ralley, D., "Genetic algorithm as a tool for melodic development", *Proceedings of the 1995 International Computer Music Conference, ICMA*, San Francisco, 1995.

[30] Stanimirović, Z., "Genetic algorithms for solving some NP-hard hub location problems", Ph.D. thesis, University of Belgrade, Faculty of Mathematics, 2007, (in Serbian).

[31] Tokui, N., and Iba, H., "Music composition with interactive evolutionary computation", *GA2000, Proceedings of the Third International Conference on Generative Art, Milan, Italy*, 2000.

[32] Towsey, M., Brown, A., Wright, S., and Diederich, J., "Towards Melodic Extension Using Genetic Algorithms", *Educational Technology & Society,* 4 (2) 2001.

[33] Wiggins, G., Papadopoulos, G., Phon-amnuaisuk, S., and Tuson, A., "Evolutionary Methods for Musical Composition", *Proc. of the CASYS98 Workshop on Anticipation, Music&Cognition,* 1998.