Chapter 1
Evolutionary System Supporting Music Composition

Rafał Dreżewski, Przemysław Tomecki

Abstract Evolutionary algorithms are heuristic techniques for finding approximate solutions of hard optimization and adaptation problems. Due to their ability to create innovative solutions, evolutionary algorithms are very well suited for applications in the domain of art. In this paper the system supporting human composer in the process of music composition with the use of evolutionary algorithm is presented. The architecture of the system as well as implemented algorithms and results of selected experiments will be presented.

1.1 Introduction

Evolutionary algorithms (EAs) are heuristic techniques for finding approximate solutions of hard optimization and adaptation problems [2]. The evolutionary algorithms can be applied to many different problems, like multi-modal optimization, multi-objective optimization, combinatorial optimization, etc.

Because of their ability to explore huge solution spaces and to generate/propose new and original (previously not even known to experts in the field) solutions to the given problems they can be applied in domains which require innovative approaches and creativeness, for example designing of engineering components, architecture, etc. Such capabilities were supposed to belong only to the human beings but the evolutionary algorithms demonstrate that computer program can also be creative and propose new, so far unexplored or unknown, solutions and even create art. One of the domains of art in which the evolutionary algorithms can support (or compete with) humans is the process of music composition. This paper presents one of the possible approaches to support the human composer.

Department of Computer Science
AGH University of Science and Technology, Kraków, Poland
e-mail: drezew@agh.edu.pl

Almost every musician is to some extent skeptical about the concept of using machines (computers) in the process of music composition—of course what we mean here is not using electronic musical instruments or some software supporting the process of music composition by the human composer, but composing musical pieces by computer itself. Probably something like complete replacement of the human composer with some music software or hardware in the process of music composition is impossible. However, humans can easily interact with the system (especially when we use evolutionary algorithms) and guide the process of music composition.

There have already appeared several approaches to supporting music composition with the use of evolutionary algorithms. J. A. Biles is the author of GenJam—system for generating jazz solos on the basis of input data from MIDI interface [3, 4]. GenJam was generating sample bars or phrases, which then were presented to the so called "mentor" who was assigning them fitness values.

B. Budzyński in his work [5] presented different approach to music composition with the use of evolutionary algorithms. There was no input data—initial population was generated at random. There was also human factor present during evolution, but composer decided only whether to continue or to stop the evolution process—music pieces were evaluated automatically, since there existed the fitness function.

Tree-based representation for musical pieces and the appropriate mutation operator was proposed in [1].

Authors of [6] proposed Harmony Search Algorithm. Musical composition was generated on the basis of vectors of notes improvised by n musical instruments. These vectors were classified on the basis of intervals between theme and variations.

R. De Prisco and R. Zaccagnino presented the idea of bass harmonization with the use of genetic algorithm [9]. Their system took bass line as the input and on its basis generated three additional voices.

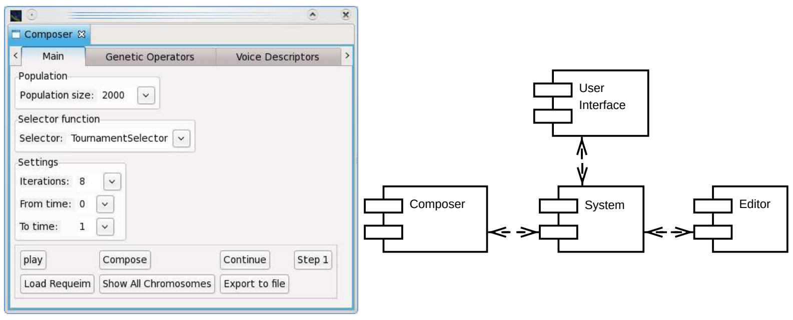
In this paper we will present evolutionary system supporting music composition—*EvoMusComp*. We will use different approach than those presented above. Firstly, in the presented system user can generate the whole new piece of music, or variation on chosen theme, with the use of evolutionary algorithm. User can interact with the system by setting values of parameters and stopping the process of evolution. The music pieces are evaluated with the use of fitness function, which is constructed from several components by the user at the beginning of evolution. In the following sections we will describe the architecture of the system, evolutionary algorithm used to generate the music and selected results of preliminary experiments.

1.2 Evolutionary System Supporting Music Composition

In this section we will describe the architecture of the evolutionary system supporting music composition (EvoMusComp) as well as the genetic algorithm used for music generation.

1.2.1 Architecture of the System

The implemented evolutionary system supporting music composition is based on the Java technology. The eclipse RCP platform was used to create user interface (see Fig. 1.1a).


Population
Population size: 2000
Selector function
Selector: TournamentSelector
Settings
Iterations: 8
From time: 0
To time: 1

(a) (b)

Fig. 1.1: GUI (a) and basic components (b) of the EvoMusComp

Basic components of the system are presented in the Fig. 1.1b. *System* component is responsible for merging information and controlling the information flow between other components. *User Interface* component is responsible for interaction with the user. *Composer* component is responsible for composition process and the evolutionary algorithm is a part of this component.

1.2.2 Genetic algorithm

The general structure of the evolutionary algorithm used in the system is based on the well known J. Holland's and D. Goldberg's genetic algorithm [7].

In the classical genetic algorithm the binary representation is used in order to represent the encoded solution. In the genetic algorithm used in the presented system such representation is adapted to the music notes description.

The attributes of the *Genotype* class, that represents the genotype of individuals include:

- *name*—*String* type—represents the name of the note in the specific octave;
- *octave*—*integer* type—represents the octave that the note is in. In the algorithm it takes values from the range 3 to 7, depending on the key signature;
- *noteKind*—*double* type—represents the duration of the note. Acceptable values are: 1.0 (whole note), 0.5 (half note), etc.

In many research papers in the area of music composition, the structure of the chromosome is based on the single voice line. In the implemented system, the chromosome is built from *n*-elements voice array. This construction allows to compose not only one voice melodies, but also the whole phrase, or even whole piece of polyphonic music.

There are two implemented genetic operators: crossover and mutation. Below we will describe them in more details because they are quite different from the standard operators used in genetic algorithms.

The recombination operator was implemented as one-point crossover. When we try to exchange genes from the one chromosome with the genes from the other chromosome there may appear some problems. We can face the situation when the new chromosomes, created after the crossover, can be invalid.

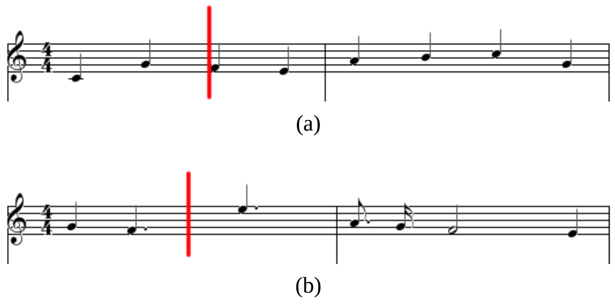


Fig. 1.2: First (a) and second (b) chromosome before crossover

Such situation is illustrated in the Fig. 1.2. The example shows that there is no note in the second chromosome in the place where the random point is placed before performing crossover. The sum of the notes' duration in the new chromosome would be longer or shorter than allowed.

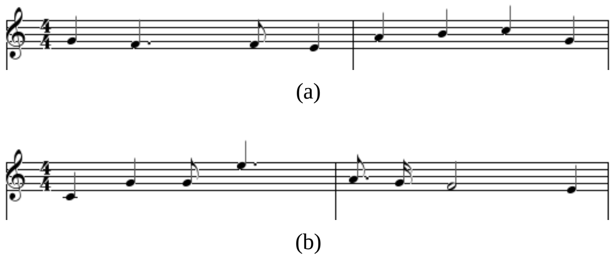


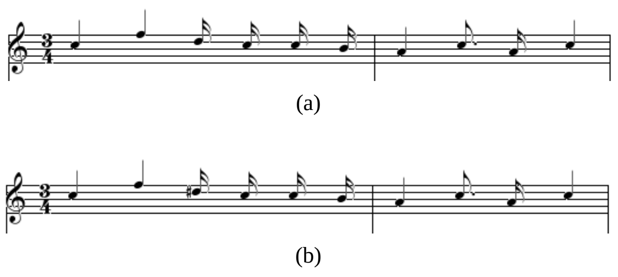
Fig. 1.3: First (a) and second (b) chromosome after crossover

Such error has to be fixed by changing the border note's values—it is shown in the Fig. 1.3.

There are three variants of mutation, which can be chosen by the user:

- *basic*—value of a note is changed half tone up or down;
- *extend*—two neighboring notes are joined;
- *split*—one note is split into two notes.

The probability of applying the mutation operator is also set by the user. The *basic* mutation of sample chromosome is shown in the Fig. 1.4.


(a)

(b)

Fig. 1.4: Chromosome before (a) and after (b) mutation

There are three selection methods implemented in the system:

- roulette selection;
- tournament selection;
- ranking selection.

As the result of applying of each of the above mechanisms the next generation consists of the same number of chromosomes (the number of the individuals in the population is constant).

There are three basic components of the fitness function, in which user can:

- define preferred kind of note and distance between neighboring notes;
- define harmony functions;
- define the chromosome that will serve as a reference point for the adaptation function.

With the use of the first and the second component we can compose a completely new piece of music, but with the use of the third one, the user is able to compose a music variation on a given theme.

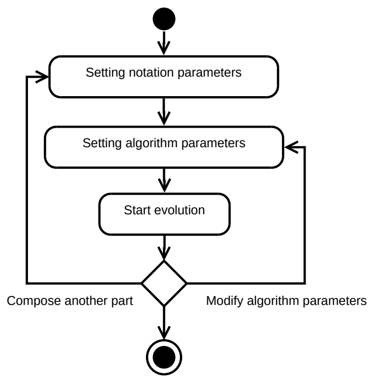

Setting notation parameters
Setting algorithm parameters
Start evolution
Compose another part
Modify algorithm parameters

Fig. 1.5: Process of music composing with the use of EvoMusComp

1.2.3 The Process of Music Composition

The process of music composing with the use of implemented system is illustrated in the Fig. 1.5. First, we define parameters connected with the music notation: measure count and key signature. Then parameters of the evolutionary algorithm are set. Evolutionary algorithm can be interrupted by the user and parameters can be modified interactively.

1.3 Experimental Results

In this section we will present the results of experiments carried out with the use of system presented in the previous section. The experiments were conducted in order to verify whether the system generates sensible results (music pieces) and which operators (and to what extent) give more satisfying results.

The results of this experiments are very promising. With the use of implemented system user can create an original piece of music. With the given melody or music fragment (theme) one can also compose the variation.

We can also mix settings of the system. At the beginning we can start composing using some basic parameters describing specific kind of note and preferred distance between notes. Then we can take satisfying results as a basis chromosome, which will be used in the next part of the music composition process as a reference point. If we want to extend our composition with an extra voice that wasn't specified at the beginning and restart the process of evolution, we can do it easily. So the implemented system is quite flexible and can serve as a composer of completely new pieces of music or variations on the given theme. Also the user can interact with the system during the process of music composition.

During experiments different aspects were taken into consideration. One of them was the question whether the algorithm is able to generate reasonable results when it runs completely without the user intervention (changing values of parameters) for a given number of steps. Such experiments were conducted with all three selection mechanisms implemented. The results are presented in the Fig. 1.6. The whole experiment was carried out with the parameters' values shown in the Table 1.1 (column *selection mechanism*).

Table 1.1: Parameters' values used in three types of experiments

	Selection mechanism	**Time of computations**	**Mutation probability**
Number of iterations (steps)	24	24	11
Number of measures	8	8	8
Genetic operators	crossover, mutation	crossover	crossover, mutation
Number of voices	1	1	1
Fitness function components	1	1	1

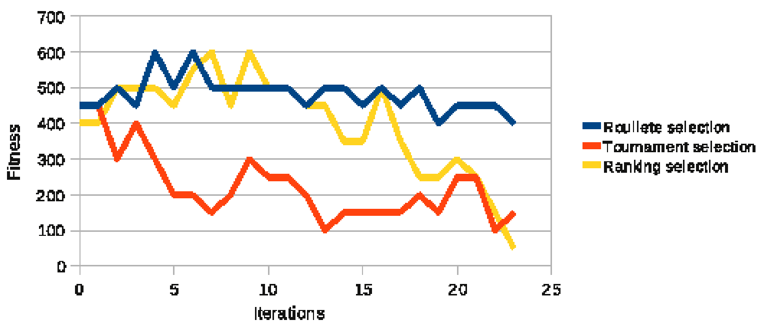


Fig. 1.6: Fitness of the best individual in consecutive steps of the algorithm for three types of selection mechanisms used and without user interaction (average values from 30 experiments)

The results show that when there is no user interaction the greater number of steps for which the evolutionary algorithm is run do not necessary increase the value of the fitness function when tournament selection is used. In the case of the other two selection mechanisms there is a little progress only during the first 10 steps. These results show that interaction with the user is necessary—the best way of music composing with the use of evolutionary algorithm is to run it for a few steps, then stop the algorithm, modify the values of some parameters and resume the run of algorithm.

The number of individuals present in the population has usually two contradictory effects. Better results are obtained when the population is larger, but it of course slows down computations. The influence of the number of individuals in the population on the time of computations is presented in the Fig. 1.7. Values of parameters used during these experiments are presented in the Table 1.1 (column *time of computations*).

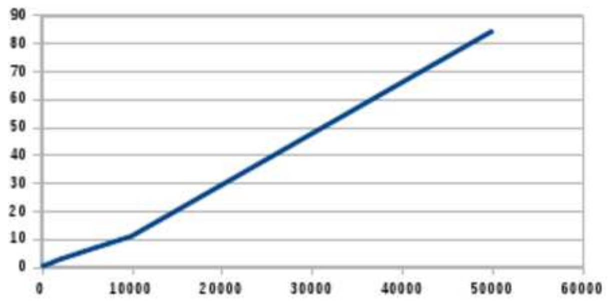


Fig. 1.7: Time of computations (in seconds) of 24 iterations versus number of individuals in the population. Average values from 30 experiments

The influence of mutation probability on the quality of obtained results is shown in the Fig. 1.8. Values of parameters used during these experiments are presented in the Table 1.1 (column *mutation probability*). It can be observed that mutation is very important operator in the presented system. Without mutation there is stagnation in the population. Aggressive mutation ($p = 0.2$) causes that better solutions appear in the population.

The sample of the “real” results obtained with the use of presented system can be seen in the Fig. 1.9. Presented piece of music is the generated variation on the “Requiem for dream” theme with an extra voice that was also composed with the use of genetic algorithm during the first stage of composing process in the two voices context.

1.4 Summary and Conclusions

In this paper we have presented the system that can support the user (composer) during the process of music composition. The system can compose completely new pieces of music or generate variations on the given theme.

User can interact with the evolutionary algorithm during initial phase, by setting the values of parameters and by constructing fitness function from the given set of

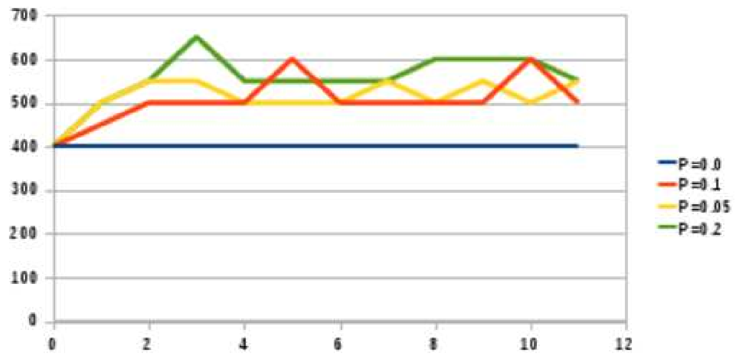


Fig. 1.8: Value of best individual's fitness for different probabilities of mutation versus number of iterations. Average values from 30 experiments

components. User can also interrupt the process of evolution at the selected moment, change the values of some parameters and continue the evolution. As presented experiments show the obtained results are much better when user interacts with the system during its run.

Preliminary experiments are quite promising—the system is able to generate completely new pieces of music or variations on the given theme, as it was presented in the previous sections. Future work will be focused on adding an expert system component which will play the role of human expert during music composition.

Acknowledgments

This research has been partially supported by the Polish Ministry of Science and Higher Education under grant N N516 5000 39.

References

1. Ando D, Dahlsted P, Nordahl MG, Iba H (2007) Interactive gp with tree representation of classical music pieces. In: [8], pp 577–584
2. Bäck T, Fogel D, Michalewicz Z (eds) (1997) Handbook of Evolutionary Computation. IOP Publishing and Oxford University Press
3. Biles JA (1994) Genjam: A genetic algorithm for generating jazz solos. In: Proceedings of the International Computer Music Conference
4. Biles JA (2002) Genjam: evolution of a jazz improviser. In: Creative evolutionary systems, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 165–

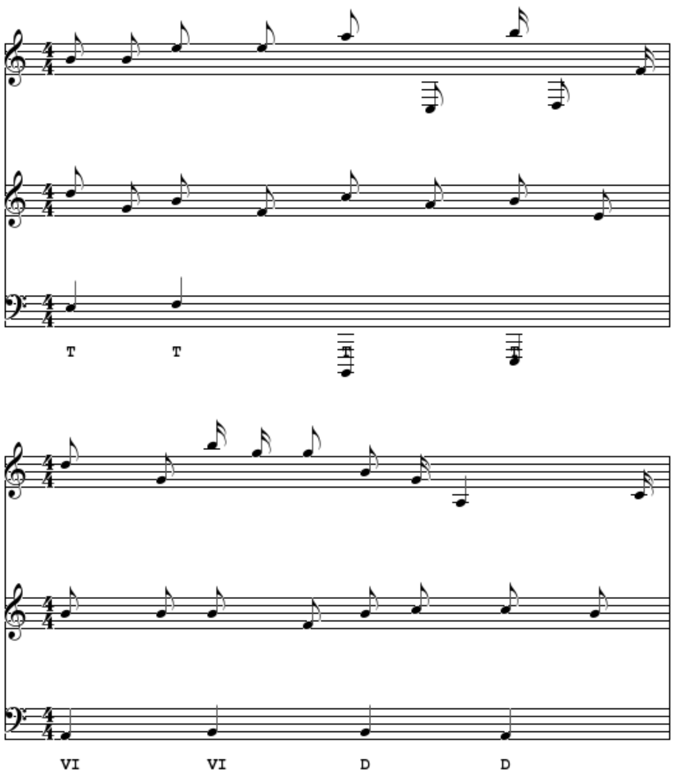


Fig. 1.9: “Requiem for dream” variation generated by EvoMusComp

187
5. Budzyński B (2002) Algorytmy genetyczne w rozwiązywaniu problemów – generowanie muzyki (Genetic algorithms in problems solving—music generation). Zeszyt Naukowy Sztuczna Inteligencja (1), in polish
6. Geem ZW, Choi JY (2007) Music composition using harmony search algorithm. In: [8], pp 593–600
7. Goldberg DE (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, Massachusetts, USA
8. M Giacobini, et al (ed) (2007) Applications of Evolutinary Computing, EvoWorkshops 2007: EvoCoMnet, EvoFIN, EvoIASP,EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog, Valencia, Spain, April11-13, 2007, Proceedings, LNCS, vol 4448, Springer
9. Roberto De Prisco RZ (2009) An evolutionary music composer algorithm for bass harmonization. In: M Giacobini, et al (ed) EvoWorkshops, Springer, LNCS, vol 5484, pp 567–572