Genetic Algorithms and the abc Music Notation Language for Rock Music Composition

Tomasz M. Oliwa
Artificial Intelligence Center
The University of Georgia, USA
oliwa@uga.edu

ABSTRACT

In this paper a music composition system based on genetic algorithms (GAs) will be presented. It can create multi-instrumental, guitar-orientated rock music using objective measures for its fitness functions. The output of this system is a song in the MIDI format. Along with this system, a unique conversion procedure from numerical values to the *abc* language (and vice versa), which allows the combination of numerical optimization with the rich expressiveness of a music description language, will be shown. The described music composition system will be further compared to other composition systems.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE Problem Solving, Control Methods, and Search — *Heuristic methods*; J.5 [**Computer Applications**]: ARTS AND HUMANITIES—*Performing arts (e.g., dance, music)*

General Terms

Experimentation

Keywords

Art and music, Genetic algorithms, Heuristics, Representations

1. INTRODUCTION

The composition of music with artificial composition systems has been implemented using various approaches and paradigms. Several Neural Network architectures like the Long-Short Term Memory (LSTM) Neural Networks [3] or Recurrent Neural Networks [11] have been used to learn (with inductive learning) from an existing dataset of music in order to create new musical pieces. This has been successful to a certain degree of expressiveness (maximum of 2 instruments, no rests and limited node representation).

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

In [9] Associative Memories have been used in combination with a context-sensitive grammar. GAs have also been utilized to compose music; a contemporary review of music created with evolutionary methods can be found in [10]. In [1] an overview about the general suitability of GAs for the composition of music is given. Taking the GA approach one has to differentiate between a semi-objective and fully objective way of the fitness evaluation. In semi-objective, interactive GAs like [13], human and computational evaluation are both used as a fitness measure for the songs. Success in [13] was limited by a small search space due to a representational restriction and thus lower quality of the resulting music; the tempo is fixed for the whole song and elementary compositional elements, such as a rest, are not implemented.

An objective fitness function is used in [8], using context-free grammar as musical rules. One important point is that previously used heuristic search algorithms like GAs with an unrestricted space usually suffer from the *fitness bottleneck* [2][10], a gigantic and mostly unusable search space. This is in particular a problem for a human-evaluation of music, because of human fatigue. There exist ways to modify existing melodies with the application of a specific noise function to create new music [7], therefore reducing the human evaluation factor. But the usage of a partly human fitness evaluation, as seen in [13], within GAs with a richer representation, makes these systems still very cumbersome due to human exhaustion, since such representations typically result in a big search space, where larger populations or longer generations are needed.

A promising GA driven approach called GenDash [10] resulted in successful live music performances of the created songs. It features a richer musical representation and uses already high-quality music as the initial population, whence new music is created. Another prominent GA music composition system is GenJam [10], which is able to improvise music in real-time.

The approach presented here is unique not only because of its GA setup and the GA individual representation described below, but also because it takes advantage of the *abc* language, a musical descriptive language in the ASCII format. In [12], the author used the *abc* language to create a music composition system based on Neural Networks and Probabilistic Finite-State Machines. Any MIDI song can be converted into an *abc* description file, and this *abc* file can be further converted to a subsequent ordering of numerical values. This process also works in the other direction, making it possible to do optimization on the integers and transforming the fittest individual of the final generation (best song)

into a MIDI song or a musical score with the intermediate step of the *abc* notation.

2. CONCEPT OF THE CREATED MUSIC

Music can be defined as the chronology of notes, and a note itself by its pitch and length. In classical music theory, a song is typically written in a specific musical scale (a subset of the set of all notes) and previous music composition systems usually strictly stick to one scale. By contrast, the composition system presented here is using three different implementations of the E-Minor scale, the most important scale in rock music.

- The *E-Minor Blues Scale*, especially focusing on the blue notes (notes which give the special blues feeling and form a triplet with their neighbor notes).
- The *E-Melodic Minor (descending) Scale*, the standard rock music scale.
- The *E-Harmonic Minor Scale*, with a chromatically raised seventh degree (compared to the E Melodic scale) to create a neo-classical sound.

The developed GA system is able to create songs consisting of up to 4 different instruments, and the quantity and kind of instruments can also vary from song to song.

- A *lead guitar*, which expresses instrumental virtuosity through different lead guitar techniques such as tapping or fast triplets.
- A *rhythm guitar*, responsible for the background rhythmicity of the created songs with a strong riffing through a power chord based structure.
- A *rock organ/piano* to broaden the spectrum of possible background rhythmicity or melody.
- The *drums*, which provide the beat structure of music with a pushing and pulsing effect through the song.

3. GENETIC ALGORITHMS

3.1 Genotype/Phenotype

Genetic Algorithms (GAs) are a form of a heuristic search, which are based on the Darwinian ideas of evolution of individuals through natural *selection* and *mutation*, leading to the "survival of the fittest" individuals (finding the best solution in the search space). It would go beyond the scope of this section to mention all the important contributors (like Bremermann or Fraser) to the invention and development of GAs, but for good reference about GAs the works of Holland [6] and Goldberg [5] can be consulted. Every individual is a dual-entity [4], consisting of its *genotype* and *phenotype*. In the presented system an individual has a low-level *representation*, on its *genotype* level; it is an arbitrary but fixed multi-dimensional array of genes, each associated with an allele of bounded-integer range.

The *phenotype*, encoded through these *genotypic* traits, is a musical score, a song, consisting of the notes for each of the different instruments used. The GA composition system is written in C++ and uses the GAlib[1], a C++ library of genetic algorithm components.

[1]http://lancet.mit.edu/ga/

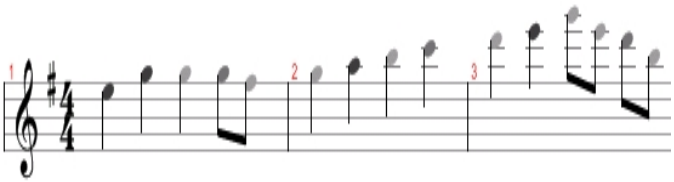

Figure 1: Beginning of "Claret and Oysters" in a musical score

```
E2 G2 G2 G      (9,10), (11,10), (11,10), (11,8)
F G2 A2 B2      (10,8), (11,10), (12,10), (12,10)
c2 d2 e2 g      (14,10), (15,10), (16,10), (18,8)
e d B           (16,8), (15,8), (12,8)
```

Figure 2: Beginning of "Claret and Oysters" in the *abc* language (left) and the integer representation (right). A node is defined by (pitch,length).

3.2 Representation

Before describing the structure of the GA individuals, the *abc* language will be introduced to explain the mapping of *genotype*, the intermediate *abc* notation of the individual and the resulting *phenotype*.

3.3 *abc* standard notation

abc[2] is a language which is used to notate music in the ASCII format. This language features a rich set of symbols and is able to express complex musical structures including but not restricted to notes, note-lengths, triplets, accidentals, chords, grace notes and accents. A song notated in the *abc* language can be converted into the MIDI format by programs[3]. The GA music composition system is able to:

- Compute the *abc* language symbols encoded as integer values in the genome alleles.
- Use transform functions which assign every integer a distinct symbol or symbol string from the *abc* language at the end of the evolutionary process.
- Thus transforming the best individual into an *abc* file.

With the *abc* language as an intermediate step, this procedure allows combining the power of GAs with a very rich musical representation.

To give an intuitive and illustrative example of what this transformation looks like, the beginning of the song "Claret and Oysters"[4] is shown in Figure 1 as musical score and in Figure 2 in the corresponding *abc* language and an internal integer representation.

3.4 Structure of the individuals

An individual consists of k-allele set genomes (where k corresponds to the number of previously specified instruments) which hold integer values. This can be seen in Figure 3.

There are 5 possible kinds of genomes:

[2]http://www.walshaw.plus.com/abc/
[3]http://abc.sourceforge.net/abcMIDI/
[4]Song "Claret and Oysters" by Andy Anderson

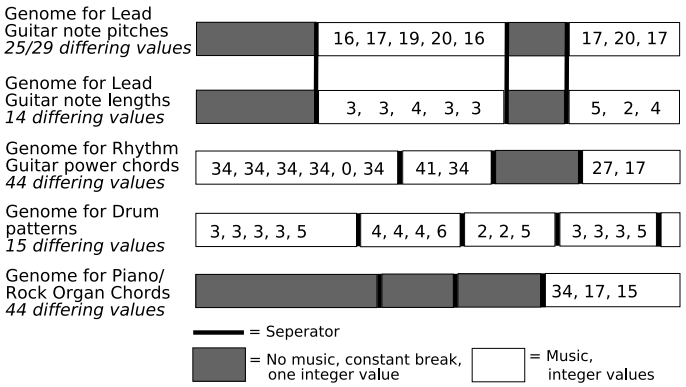

Figure 3: A possible individual with five genomes is presented here. The integer values inside the genomes can be seen. Also the *separators* are shown, which (arbitrary but fixed) split up a genome in different parts for different fitness functions.

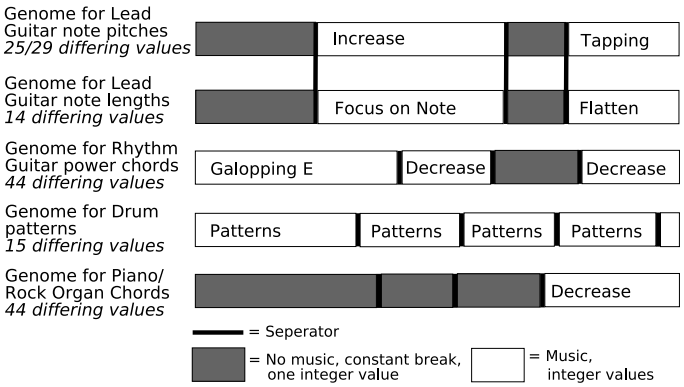

Figure 4: A possible individual with five genomes, which hold the corresponding musical definitions achieved through the fitness functions.

1. *Genome 1*: Allele set integer genome for Lead Guitar note pitches with 25 or 29 values. Seven notes from the E Melodic Minor (E ♯F G A H C D) or E Harmonic Minor (raised seventh note for a three half steps interval between the sixth and the seventh note) Scale are included. The notes are present in four octaves, leading to 24 notes. Additionally, four blue notes can be used. The last value is the pause operator, which inserts a pause instead of a note in the song.

2. *Genome 2*: Allele set integer genome for Lead Guitar note lengths with 14 values. Seven different lengths are used with a slightly uneven distribution (thus same speed for different values) that favors faster lengths.

3. *Genome 3*: Allele set integer genome for Rhythm Guitar power chords with 44 values. The values correspond to 11 different power chords inside the E Minor Scale, each power chord with four possible lengths. A power chord is a genderless (no Major/Minor notes) chord, consisting of a base tone, the next lower fifth tone and the base tone one octave higher. The majority of rock songs uses power chords as underlying chords.

4. *Genome 4*: Allele set integer genome for Drum patterns with 15 values. There are 12 different drum patterns, which are basic drum beat constructs, and three different speed definitions for these patterns.

5. *Genome 5*: Allele set integer genome for Piano/Rock Organ Chords with 44 values. The piano is using the same chord based representation as the *Rhythm* guitar.

Furthermore, at the beginning of the GA run, a *shape* of a song is randomly created. *Shape* is an arbitrary but fixed division of the song broken into *segments* by *separators*. Each of the *segments* has an arbitrary but fixed length. After that, each of these *segments* is assigned a fitness function from the set of fitness functions of the respective genome. This assignment will stay throughout the whole GA run. The individual from Figure 3 with its fitness function assignment can be seen in Figure 4.

Every individual in the whole run has the same *shape* and therefore the same *separator* segmentation; but every time the GA is started initially, a different shape is created. Thus it is guaranteed that a completely new structured song will be made every time the GA begins. This allows for uniqueness in the number of instruments used and structure of each newly created song. Also, for each *segment* of the song, either a random decision is made as to which of the three scales to follow, or the system can be adjusted in a way to have a fixed scale. For example a fixed E-Harmonic Minor scale for every *segment* allows the creation of baroque-like music.

3.5 Fitness Functions

In general, the fitness function evaluation works as follows:

1. Each *segment* of a genome has its assigned fitness function (as defined by the *shape*) and will be evaluated by it.

2. A *segment* will be awarded with as many points as it contains values which fulfill the fitness constraints.

3. This gives a measure for the maximization of the individuals, whose fitness value is defined by the combined fitness values of their *segments*.

What follows is an assortment of different types of fitness functions. Along with them, examples of *segments* and their raw fitness values according to these functions are given to help with the replicability of the results.

3.6 Genome 1 - Lead Guitar note pitches

3.6.1 Ascending Tone scale

This function creates ascending tone scales, meaning that every tone (from a random starting tone) should be higher than its predecessor. The individual gets a reward for every pair of notes which fulfill this constraint and a higher reward if the distance between the notes is just one note pitch. This technique is one of the standard lead guitar solo techniques and is sometimes called ”walking up the scale” and can be seen in Figure 5. Other ”walking up the scale” functions give rewards for an increasing order of k notes, which is dropped

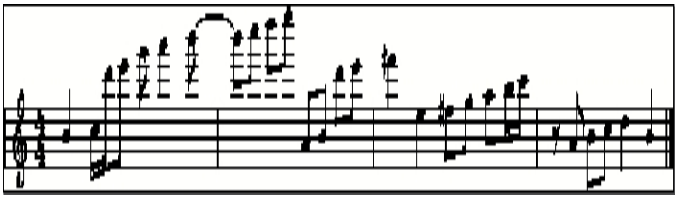

Figure 5: Ascending Tone scale for Lead Guitar

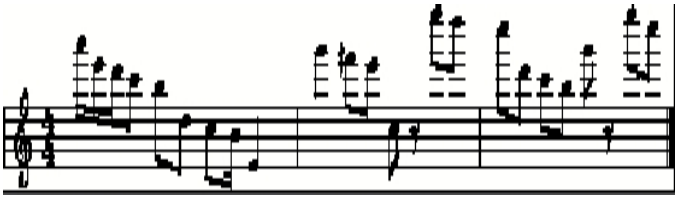

Figure 6: Descending Tone scale for Lead Guitar

back to the k - m note and increased again, resulting in a slower climbing with temporal repositions.

To provide an example how *segments* following this fitness function would look like, consider the following two *segments* (written in their musical notation):

E ♯F G A, ♯F G A H, G A H C
E ♯F G ***C****, ♯F G* ***D*** *H, G A H* ***A***

The values for k and m have been randomly assigned to $k = 4$ and $m = 3$, with the constraint that $k \geq m$. The first *segment* gets the maximum score of 12 (which equals the number of notes in the *segment*), since it perfectly matches the description. The second *segment* however would get a score of 9 and therefore three points below the optimal *segment*, since three note values do not fulfill the assigned fitness function.

In this and in all following *segment* examples, the **bold** notes are notes which do not fulfill the fitness function constraints.

3.6.2 Descending Tone scale

The descending tone scale is an analog of the ascending tone scale, but with pitches getting lower with the same rewarding procedure, also known as "walking down the scale"; an example can be seen in Figure 6. Other "walking down the scale" functions with an usage of slower descent with temporal reposition are also implemented, with analog definitions and constraints for k and m as in the *Ascending Tone scale*. The following two *segments* illustrate this:

H A G, A G ♯F, G ♯F E
H A ***E****,* ***D*** ***C*** *♯F, G ♯F* ***C***

If the constraint values have been randomly assigned to $k = 3$ and $m = 2$, the first *segment* gets the maximum score of nine (since all nine notes fulfill the fitness function requirements). The second *segment* gets a score of five, since four of the notes do not fulfill the necessary fitness constraints.

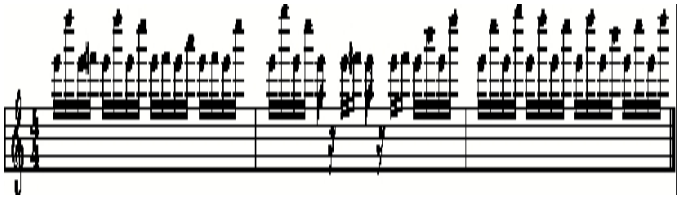

Figure 7: Tapping for Lead Guitar

3.6.3 Tapping

Tapping is a virtuoso guitar playing technique; it is a fast-played alternation of a low base note and different subsequent notes, which are all higher or lower than the base note. To achieve this technique every k-note is forced to be the base note and all the other notes are forced to be higher (or lower) than the base note. Different tapping functions are implemented, with a varying size of k and with the arbitrary but fixed base note in both a lower and a higher octave. A resulting score is illustrated in Figure 7. Two *segments* are presented, which will be evaluated on the tapping fitness function:

G A H, G A H, G H C, G A H
G A H, ***E*** *A H, G* ***G*** *C, G A H*

If the constraint values have been randomly assigned to $k = 3$, the base note to G, and the ordering to be ascending, the first *segment* gets the maximum score of 12 (all 12 notes fulfill the fitness function requirements). The second *segment* gets a score of ten, because one of the base notes is a wrong note and one of the higher notes is not higher than the base note.

3.6.4 Flatten

The flatten function reduces the range of possible values within the genome. This creates a fluent melody which is more suitable for slower passages; this suitability gives the fitness function allocator a small bias to assign lower lengths here. There are two methods of flattening. The first one uses a random base note for every *segment* and counts the notes that stay in a certain range above the base note. The other one uses the standard deviation of the note values, and the lower it is, the more flat the melody becomes. The result of this method is shown in Figure 8. An example for the application of the first flatten function are the following two *segments*:

A, G, A, H, C, C, ♯F
G, H, ***E****,* ***D****, G, A, ♯F*

If the base note is A, and the range to two, the first *segment* gets the maximum score of seven (the notes stay within the range of 2 around the base note). The second *segment* gets a score of five, two of the notes are outside of the range.

3.7 Genome 2 - Lead Guitar note lengths

3.7.1 Note Length

Besides using the implemented flatten function (as described above) to shape a pleasant progression of tone lengths, the note length function can also produce slow sections with

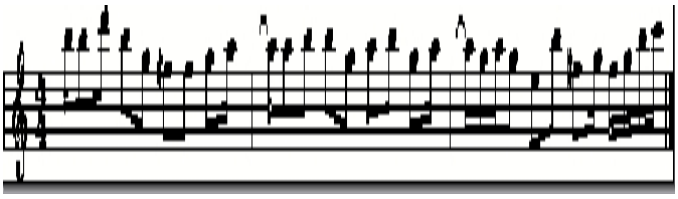

Figure 8: Flatten for Lead Guitar

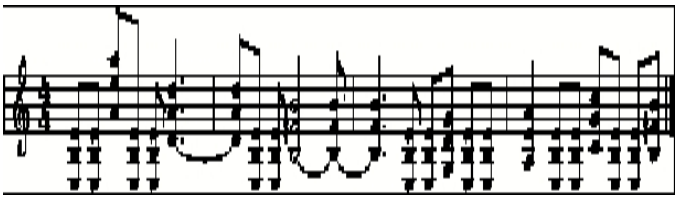

Figure 9: Heavy E based Rhythm Guitar

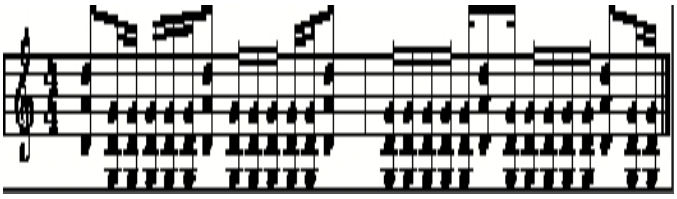

Figure 10: Another Heavy E based Rhythm Guitar

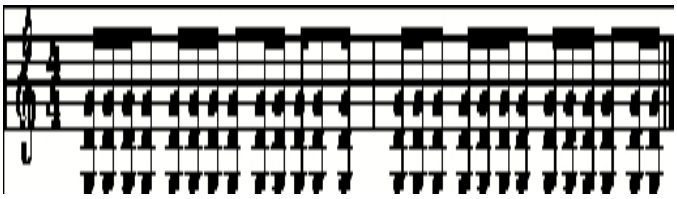

Figure 11: Galloping Rhythm Guitar

k slow tone lengths of a kind interrupted by another faster tone length. This means that after every k fast tone lengths, another slower tone length follows. The two following *segments* are evaluated by the fitness function for slow sections:

2 2 2 4, 2 2 2 5
2 2 **3** 4, **3** 2 2 5

If $k = 4$, the slow tone length = 2 and the numerical value denotes a faster speed of a note, the first *segment* gets the maximum score of eight. The second *segment* gets a score of six, two of the note lengths do not equal the slow notes. In addition, a speed function is implemented which rewards fast tone lengths if the corresponding *segment* in the pitch chromosome is optimized for tapping or ascending/descending tone scales, this will help to shape faster lead guitar parts in the melody.

3.8 Genome 3 - Rhythm Guitar power chords

3.8.1 Ascending Power chords

The *Rhythm Guitar power chord's* function echoes the *Ascending Tone scale's* function of the the lead guitar section, but the tone lengths do not change between the notes because of the different representations. For the rhythm guitar, lengths and pitches of the chords are encoded together.

3.8.2 Descending Power chords

These are analogous to the power chord's ascending function.

3.8.3 Heavy E based Rhythm

This rhythm has, as its name implies, a strong emphasis on the usage of the E chord. This chord is very important and popular because it is the lowest possible chord which can be played on a standard tuned guitar and therefore it produces a strong, driving sound. In addition, it is the base chord of the E-minor scale. A random value k is chosen to determine how often the E chord has to be played before another randomly chosen chord is inserted. For every note that fits into the scheme the *segment* gets a reward. Figure 9 and Figure 10 show two different realizations of this rhythm.

Two *segments* are provided to illustrate the application of this fitness function:

E E E E C, E E E E G
E ***a C*** *E a, E E E E* ***E***

Here, $k = 4$ and the base chord is E. The first *segment* gets the maximum score of ten, but the second *segment* is only assigned a score of seven, because the chords do not equal the base chord, or the last chord equals the base chord.

3.8.4 Galloping Rhythm

The galloping rhythm is a widely used style in rock music. It is related to the heavy E rhythm but the base chord does not have to be an E. The filling chord is equal to the base chord but can have a different (fixed) speed; this can be seen in Figure 11.

Two *segments* show the usage of this fitness function:

a a a A, a a a A
a a ***E*** *A, a a* ***D D***

Here, $k = 3$ and the base chord is a. A lowercase 'a' denotes the chord A played one measure faster. The first *segment* gets the maximum score of eight. The score for the second *segment* is 5, because the chords do not equal the base chord.

3.9 Genome 4 - Drum patterns

3.9.1 Drum Blocks

The drum track is built of short, predefined patterns. The fitness function rewards a *section* where a pattern is repeated up to four times, and is then followed by a filling pattern. In addition, the function assures that a small number of pauses occur throughout the whole chromosome.

3.10 Genome 5 - Piano / Rock Organ Chords

The chords specified for the piano / rock organ are optimized in the same way as the chords for the rhythm guitar. A restriction forbids the use of a "walking the scale" that simultaneously opposes the direction of the guitar. There is also a bias to use either the piano / rock organ or the rhythm guitar to create the background rhythm.

Table 1: GA Setup

Operator/Paramenter	Setting
Representation	k-allele set genomes
Recombination	Two point crossover
Recombination probability	80%
Mutation	Swap or Gaussian mutation
Mutation prob.	0.1
Parent Selection	Tournament Selection
Survival Selection	Generational
Population Size	200
Number of Offspring	200
Scaling	Linear-based
Initialisation	Random
Terminal Condition	Last Generation

3.11 Operators

The configuration of the operators and parameters of the GA for music composition can be seen in Table 1. With this configuration, a complete song generation on the genotype level (which includes both the setup of the *shape* and the GA run to find the best individual) takes approximately 11 seconds on a contemporary computer for about half a minute of music.

4. RESULTING MUSIC

4.1 Composed music

It should be noted that the GA music composition system is given much freedom, not only by the rich representation, but also by the randomization of the number and size of the *segments* in the genomes that results in the assignment of different fitness functions from the set of all fitness functions. Furthermore many fitness functions, such as the *Ascending Tone scale*, feature arbitrary but fixed setups. No best song of one run sounds the same as any best song from a previous run. While minimizing the restrictions and constraints of the music creation by both ideas of musical theory as well as rock music instrument techniques, music is created which exhibits these ideas, but still remains unique.

Five different songs, created by the music composition GA, have been uploaded to [5] and [6], and the readers of this paper are encouraged to listen to those songs to get an idea of the quality of the composed music.

5. GRAPHS

Depending on the structure and segmentation of the different genomes, differing convergence properties can be observed. In Figure 12, a fitness evaluation for a whole song for a lead instrument with 5 *segments* is presented. The second and the last *segments* get a worse score than the remaining *segments*. It should be noted that in this run, the GA found an optimum for the *segments* after a short time (usually 25 to 30 generations). Then the diversity in the *segment* is lost and the musical *segment* converges. In contrast to that, in Figure 13, a single song *segment* for the piano is shown. Unlike Figure 12, the assigned fitness for the piano is constantly improving up till the end of the evolution.

[5]http://www.uni-koblenz.de/~zophar/garock/
[6]http://www.geocities.com/darkstarxy/index.html

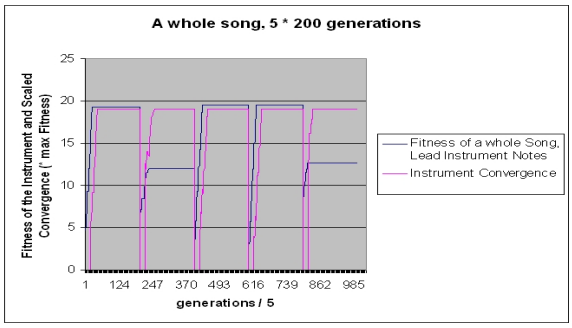

Figure 12: Fitness of a whole song for the lead instrument. There are 5 different *segments* in the song, each following its own fitness function

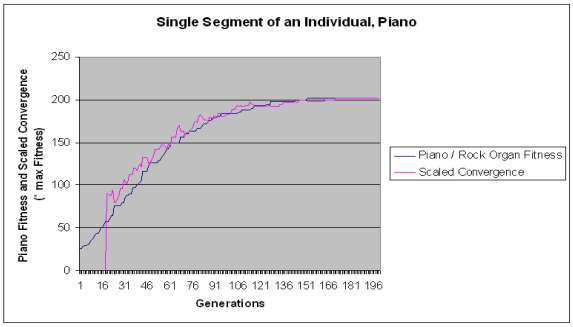

Figure 13: Fitness of a song *segment*, the instrument is piano

6. COMPARISON WITH OTHER MUSIC COMPOSITION SYSTEMS

6.1 Comparison with a Neural Networks (NN) approach

With a NN, a music composition system is limited to learn via inductive learning from an existing database of songs in order to create new songs. The problem with a RNN approach is that long term dependencies of music cannot be represented in practice [11], thus leading to non-coherent music. The results of [3] look much more promising, but only 2 instruments are used and the presented system is restricted to 13 notes for the lead instrument and 12 notes for the rhythm instrument. Also no pauses are taken into consideration, making the composed music more coherent than [11], but still the lacking expressive power of a rich representation.

6.2 Comparison with an objective fitness function approach

Competitive results were obtained with GenJam [10] and GenDash [10]. GenJam, having a slightly different job definition (jazz music improvisation), features a good diversity of melodies and musical ideas, but limits itself to one instrument and requires additional information from the accompanist. GenJam [10], likewise offers rich representation possibilities and diversity in music, but the system itself needs already composed musical data in the initial population. The *motifs* in [8] share some resemblance to the presented *seg-*

ments. However, the *motif* combination is only done by note transition resolutions of last and first notes and the possibility of composition with multiple instruments is not discussed.

6.3 Comparison with a subjective fitness function approach

The approach in [13] is to create music which reflects users' subjectivity towards it. A user can get "cheerful" or "sad" music out of the system by repeatedly evaluating created music. The use of a virus operator inside the GA is interesting, but this partly objective, partly subjective fitness approach "steers" the GA into the "right" direction, leaving less work for the objective evaluation. Furthermore, this program uses a fixed tempo for the whole song and is not able to use a pause, which restricts the search space. Also, no melodic complexity with tempo changes or instrument interaction can be created.

7. CONCLUSION AND FUTURE WORK

The presented GA music composition system can create music which competes with or surpasses current approaches in terms of diversity, virtuosity, and variety of instruments. This is due to the representation of individuals as k-allele set genomes, the usage of *segments* with arbitrary but fixed fitness functions and the corresponding *abc* language translations of the integer representations. Another advantage of the presented system compared to other approaches is that it is able to create an arbitrary long song structure with a diverse multi-instrumental cast. The possibilities of MIDI and the *abc* language allow a very rich representation of music. The usage of integer values, which correspond to notes in *abc* musical notation allows an easy handling and processing of musical data. Another advantage of the usage of the *abc* language as an intermediate step is that the final song is human-readable and also human-editable in any text editor without difficulties.

With little work, the system can be extended to include more musical scales, more instruments, and more fitness functions. Research on this system is currently being conducted in order to implement inter-instrumental fitness functions, where rhythm and lead instruments as well as drums can be optimized for a coherent cooperation between themselves. Although it is an exaggeration to say that the presented system creates genuine new musical ideas (due to the encoded knowledge of music in the formulaic fitness functions), it is a step towards the creation of these. Further research will implement repetitions with slight variations, a common theme in contemporary rock melodies. Research also continues in the creation of music with a greater unity and with a general theme throughout.

8. REFERENCES

[1] G. Andrew and P. Copley. The suitability of genetic algorithms for musical composition. *Contemporary Music Review*, 22(3):43–55, 2003.

[2] J. A. Biles. Genjam: A genetic algorithm for generating jazz solos, June 15 1994.

[3] D. Eck and J. Schmidhuber. Finding temporal structure in music: blues improvisation with lstm recurrent networks. In *Neural Networks for Signal Processing XII, Proceedings of the 2002 IEEE Workshop*, pages 747–756. IEEE, 2002.

[4] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.

[5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, January 1989.

[6] J. H. Holland. *Adaptation in natural artificial systems*. University of Michigan Press, Ann Arbor, 1975.

[7] Y.-W. Jeon, I.-K. Lee, and J.-C. Yoon. Generating and modifying melody using editable noise function. In *CMMR*, volume 3902 of *Lecture Notes in Computer Science*, pages 164–168. Springer, 2005.

[8] Y. M. A. Khalifa, B. K. Khan, J. Begovic, A. Wisdom, and A. M. Wheeler. Evolutionary music composer integrating formal grammar. In *GECCO '07: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, pages 2519–2526, New York, NY, USA, 2007. ACM.

[9] T. Kohonen. A self-learning musical grammar, or "associative memory of the second kind". In *IJCNN*, volume I, pages I–1–I–6, Washington DC, 1989. IEEE.

[10] E. R. Miranda and J. A. Biles, editors. *Evolutionary Computer Music*. Springer, Mar. 2007.

[11] M. C. Mozer. Neural network music composition by prediction: exploring the benefits of psychoacoustic constraints and multi-scale processing. pages 227–260, 1999.

[12] T. Oliwa and M. Wagner. Composing music with neural networks and probabilistic finite-state machines. In *Applications of Evolutionary Computing, EvoWorkshops2008: EvoCOMNET, EvoFIN, EvoHOT, EvoIASP, EvoMUSART, EvoNUM, EvoSTOC, EvoTHEORY, EvoTransLog*, LNCS. Springer Verlag, 26-28 Mar. 2008.

[13] M. Unehara and T. Onisawa. Construction of music composition system with interactive genetic algorithm. In *Proceedings of 6th Asian Design International Conference*, pages 84–89, 2003.