Music Segmentation With Genetic Algorithms

Brigitte Rafael*, Stefan Oertl*, Michael Affenzeller† and Stefan Wagner†
*Re-Compose GmbH
Vienna, Austria,
{brigitte.rafael, stefan.oertl}@re-compose.com
†Heuristic and Evolutionary Algorithms Laboratory
School of Informatics, Communications and Media
Upper Austria University of Applied Sciences,
Hagenberg, Austria,
{michael.affenzeller, stefan.wagner}@heuristiclab.com

***Abstract*—Music segmentation is a key issue in music information retrieval (MIR) as it provides an insight into the structure of a composition. Based on structural information, several tasks related to MIR such as searching and browsing large music collections, visualizing musical structure, lyric alignment, and music summarization can be further improved. Various approaches are available to achieve an appropriate segmentation of a given composition. The authors of this paper present an approach to appliy genetic algorithms for a solution to the segmentation problem.**

***Keywords*-music information retrieval; music segmentation; pattern recognition; heuristic optimization; genetic algorithms;**

I. Introduction

Music segmentation targets at the identification of boundaries between structurally relevant parts of a composition [1]–[4] to enable or improve MIR-related tasks such as searching and browsing large music collections, visualizing musical structure, music summarization [5], and lyric alignment [6]. The most common approach aims at detecting these boundaries with the aid of a novelty score which is described in [1] and [6]. Methods based on that score are limited to compositions following certain rules and principles as they require the existence of domain knowledge (extraopus). However, the authors' method is not based on this kind of a priori knowledge but focuses on the information provided within the piece itself (intraopus). In consequence, it can be applied to a broader musical spectrum. Music is analyzed based on its self-similarity and repetitions are used to detect segments. The algorithm then clusters similar segments to create segment groups (i.e., collections of several nonoverlapping segments that fulfill a given similarity condition).

This paper presents the idea of applying a genetic algorithm to achieve an automatic segmentation of a composition. In the second section, the reader is introduced to the basic characteristics of genetic algorithms and their suitability for the segmentation problem. The third section describes the mapping of music features to the components of a genetic algorithm. A discussion of the results and an outlook on future work concludes the paper.

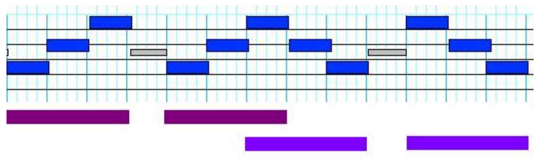

Fig. 1. Ambiguity in the clustering process

II. Genetic Algorithms

Genetic algorithms belong to the field of evolutionary computing. They are useful for problems with a very large solution space where it is not possible to evaluate all solutions and to obtain the optimal solution within reasonable time [7]. Furthermore, genetic algorithms introduce diversity and avoid getting stuck in a local optimum, therefore they are particularly suitable for problem domains with complex fitness landscapes.

Genetic algorithms simulate the biologic process of evolution using techniques like selection, crossover, and mutation. Each candidate solution is represented as an individual within a population. By evolving the population using selection, crossover, and mutation the average fitness of the population increases over generations. The best individual of the population evolves towards the global optimal solution.

In a first step, a genetic algorithm initializes a population randomly. All individuals of the population are evaluated by calculating their fitness values. To evolve the next generation the algorithm chooses above-average parents and combines them using the crossover operator. The individuals created this way can be mutated according to a mutation rate and are then inserted into the new generation. Various selection methods can be applied to the parents as well as to the resulting children. The genetic algorithm stops when a fixed number of generations has been created, a certain fitness value has been exceeded, or stagnation of the population's fitness has been reached. Details about genetic algorithms and evolutionary computing can be found in [8], [9].

A. Using Genetic Algorithms for Music Segmentation

A segmentation of a composition consists of nonoverlapping segments that represent the internal structure of the music.

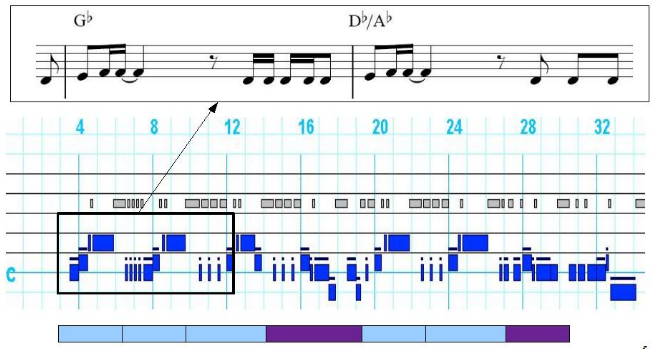
Fig. 2. Graphical representation of a MIDI track

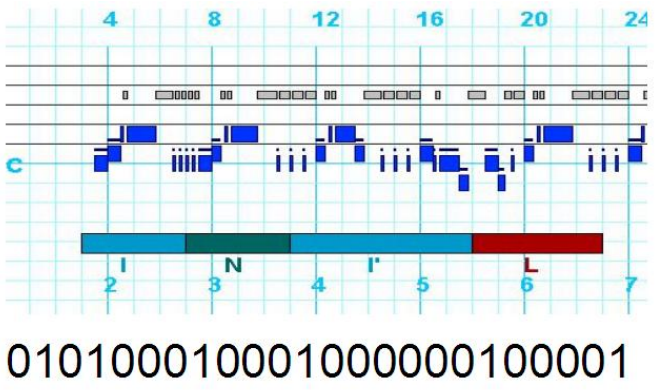

Fig. 3. Encoding of a segment as a bit vector

Similar segments form segment groups through clustering. There are several reasons why music segmentation is not a trivial problem:

- Segments can vary in duration and in the number of notes they contain.
- Segments within a segment group usually are not exact repetitions but approximate ones.
- Distances between segments do not have to be regular; gaps of undefined duration may exist.
- Clustering similar segments may lead to ambiguities if overlapping segments are detected and one of them has to be chosen (see Fig. 1).
- Single notes between detected segments pose a problem if they cannot be unambiguously assigned to neither of the neighbouring segments. A decision has then to be made if they should form their own segments or if they belong to any of the existing ones.

As segments can start at any arbitrary position of the composition and can have any duration ranging from one beat up to the duration of the composition, the runtime complexity depends on the duration of the composition. An evaluation of all possible segment combinations leads to an explosion of runtime because the runtime increases exponentially for longer compositions. Therefore it is not possible to evaluate all potential segmentations but a solution of sufficient quality has to be found in reasonable time. Given all these circumstances, the problem domain of music segmentation turns out to be highly suited for applying genetic algorithms.

III. Implementation of the Genetic Algorithm

The success of a genetic algorithm mainly depends on a suitable problem representation. Candidate solutions have to be encoded to represent individuals in the evolving population. The encoding has to enable the application of an evaluation function as well as crossover and mutation operators (all these parameters are problem-dependent). [10] describes the importance of problem representations for good results of genetic algorithms.

Since the music data is represented in the MIDI (Musical Instrument Digital Interface) format, information is available for each track separately. In consequence, the optimal solution for each track can be found independently.

Fig. 2 illustrates an extract from the notes of one track. The upper part shows a graphical representation of notes as it might be familiar to the reader. The lower part is similar to the common representation. It also contains five staff lines and an additional line for Middle C. Notes are displayed as blue boxes and the box widths show note durations. Lines above notes indicate an increment of the pitch value by one semitone. Rests are represented as grey boxes. Vertical lines correspond to the vertical lines in the upper picture and represent bar changes. The colored rectangles below the notes give a sample segmentation. Segments of the same color belong to the same segment groups.

A. *Problem Encoding*

The authors have decided to use simple bit vector individuals allowing the application of existing operators. To encode a segmentation of a track one beat of the track is mapped to one bit of the individual. The genome size thus depends on the beat length of the track. Bits of value 1 indicate the start of a new segment. A sample encoding can be seen in Fig. 3. A new segment starts with each 1 in the bit vector, apart from the first one. This results from the precondition that a segment must have a minimum duration of one bar (which equals four beats in this case). As a consequence of the mapping explained above, segments can start on full beats only. However, this is not considered a problem because segment changes are unlikely to appear between full beats.

To produce the inital population segments are created randomly by assigning 0 or 1 to each bit in an individual. Since segments must be of a minimum length, the probability of 0 is higher than the probability of 1. Various probability ratios are used for different individuals leading to a wider variation of segment durations.

B. *Evaluation*

An evaluation function is necessary to calculate the fitness of an individual. Several features contribute to the evaluation function increasing or decreasing the fitness of an individual. Increasing features include

- the number of identical segments
- the number of similar segments
- the average similarity of segments within one segment group
- the total note coverage (how many notes are covered by all segments)
- the number of segments starting at full bars

Features decreasing the fitness of an individual are

- the number of segment boundaries that clip notes
- different segment durations
- irregular distances between segments
- the number of different segment groups

All these features are weighed according to their importance and summed up to result in the fitness value of an individual. The parameters for weighing the factors of the evaluation function have been tested by music experts to find the optimal values.

To define the similarity between two segments the music sequences within the segments are aligned using dynamic programming (compare [11]) and a similarity score is calculated. If the score exceeds a defined threshold, the segments belong to the same segment group.

C. Parameters

Each test run evolved 3000 generations with a population size ranging from 100 to 500 individuals. The track chosen as an example with a duration of 312 beats resulted in a bit vector length of 312 bits. As a selection operator, tournament selection with size 3 and 4 was tested as well as roulette selection. A simple single point crossover was applied for recombination. The mutation rate for bit flip mutation ranged from 0.01 to 0.1. For all test cases 1-elitism was employed. Table I gives the settings of some sample test runs as well as the total and average best results.

IV. Results

Comparing the settings described above tournament selection turned out to be more successfull than roulette selection. Higher mutation rates caused an early stagnation of the best fitness within a population, thus resulting in lower fitness values (see Fig. 4). Figures 5 and 6 show the progress of the best, average, and worst fitness within a population during its evolution process of sample test runs using tournament and roulette selection, respectively.

Figures 7, 8 and 9 show example segmentations in different phases of the evolution. While the segments seem to be random in the beginning, clear segments and segment groups can be detected in later phases.

V. Conclusion

In this paper the reader has been introduced to the problem domain of music segmentation as well as to genetic algorithms and their application to the segmentation problem. Some good results have been achieved with the first test settings but more test cases have to follow. New settings will be introduced and existing ones tested further. The parameters for the evaluation function need fine-tuning for optimal results.

The present algorithm performs a segmentation for each track separately. Future work will also approach the combination of segmentations of different tracks to form one global segmentation. Smaller segments found by the genetic algorithm will be further structured hierarchically to form longer segments until the longest structural parts of a composition are reached (e.g., chorus and verse). Building such a hierarchy might also be done using a genetic algorithm.

Currently the genetic algorithm is only applied to segmenting MIDI data. The feasibility to adapt the approach for segmenting audio data will be examined during future research.

References

[1] E. Peiszer, "Automatic audio segmentation: Segment boundary and structure detection in popular music," Master's thesis, Vienna University of Technology, Vienna, Austria, 2007.

[2] M. Levy, K. Noland, and M. Sandler, "A comparison of timbral and harmonic music segmentation algorithms," in *Proceedings of the Acoustics, Speech, and Signal Processing*, vol. 4, 2007, pp. 1433–1436.

[3] K. Jensen, "Multiple scale music segmentation using rhythm, timbre, and harmony," *EURASIP Journal on Applied Signal Processing*, vol. 2007, no. 1, 2007.

[4] M. Mueller and S. Ewert, "Joint structure analysis with applications to music annotation and synchronization," in *Proceedings of the 9th International Conference on Music Information Retrieval*, 2008, pp. 389–394.

[5] M. Cooper and J. Foote, "Summarizing popular music via structural similarity analysis," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003, pp. 127–130.

[6] K. Lee and M. Cremer, "Segmentation-based lyrics-audio alignment using dynamic programming," in *Proceedings of the 9th International Conference on Music Information Retrieval*, 2008, pp. 395–400.

[7] Z. Michalewicz and B. Fogel, *How to Solve It: Modern Heuristics*. Springer, 2000.

[8] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley Longman, 1989.

[9] M. Affenzeller, S. Winkler, S. Wagner, and A. Beham, *Genetic Algorithms and Genetic Programming Modern Concepts and Practical Applications*. CRC Press, 2009.

[10] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1992.

[11] T. Jehan, "Hierarchical multi-class self similarities," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2005, pp. 311–314.

TABLE I
COMPARISON OF DIFFERENT PARAMETER SETTINGS AND THEIR RESULTS

Selection operator	Population size	Mutation rate	Best fitness	Average fitness	Evaluations
Tournament (3)	100	0.01	138.75	128.79	297000
Tournament (3)	500	0.01	142.63	135.90	1497000
Tournament (4)	500	0.01	145.51	137.43	1497000
Tournament (3)	100	0.10	67.47	57.30	297000
Roulette	100	0.01	115.58	109.32	297000
Roulette	500	0.01	119.93	111.85	1497000
Roulette	100	0.10	62.65	51.97	297000

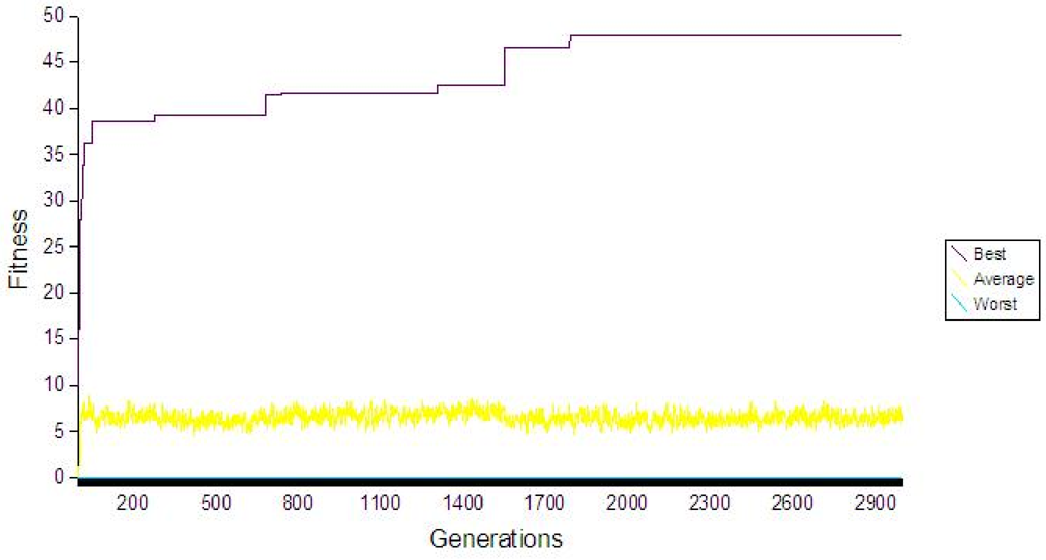

Fig. 4. Tournament selection (size 3) with a high mutation rate

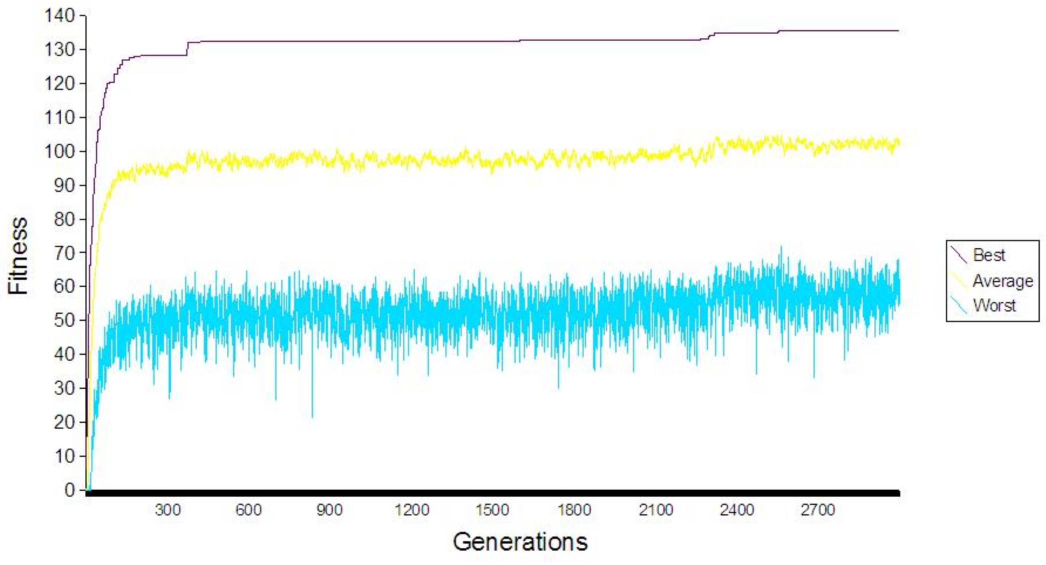

Fig. 5. Tournament selection (size 4)

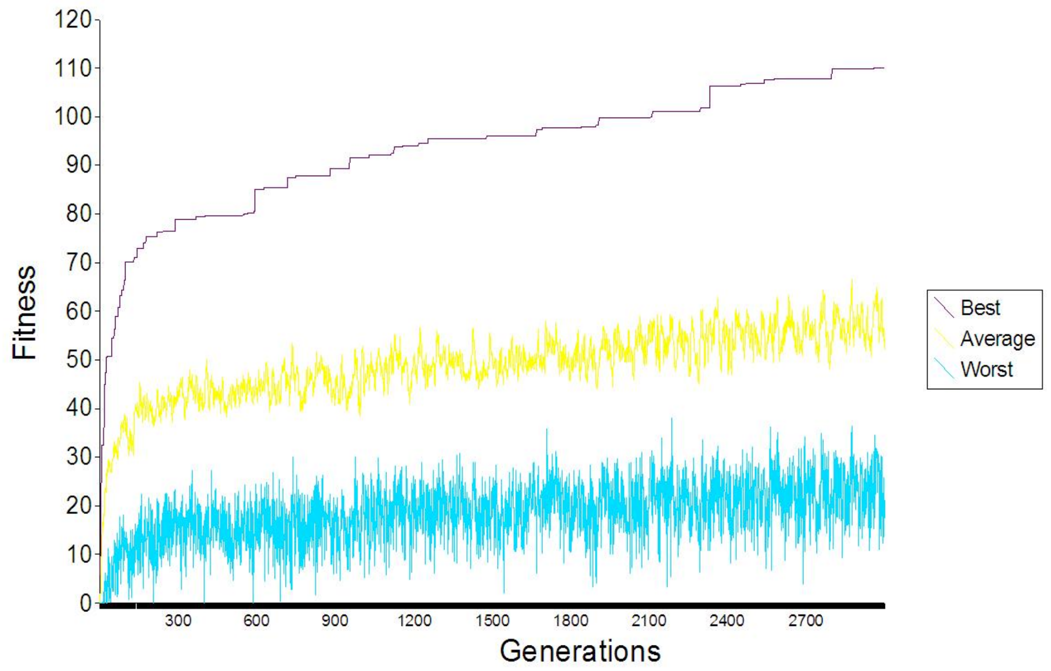

Fig. 6. Roulette selection

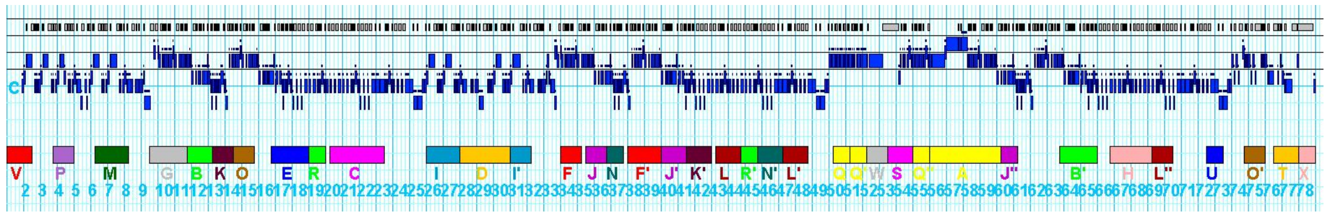

Fig. 7. Segmentation in an early generation

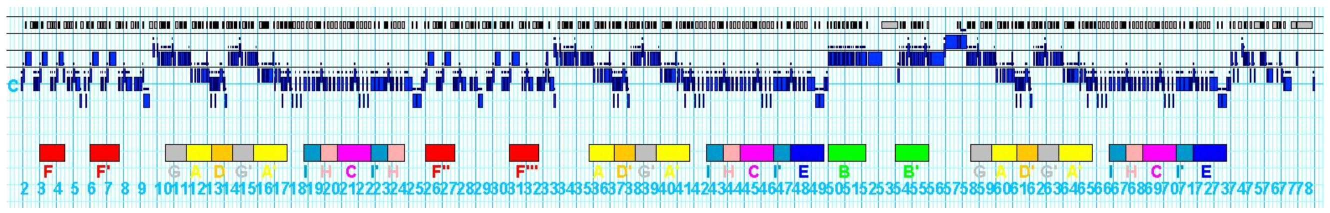

Fig. 8. Segmentation in a later generation

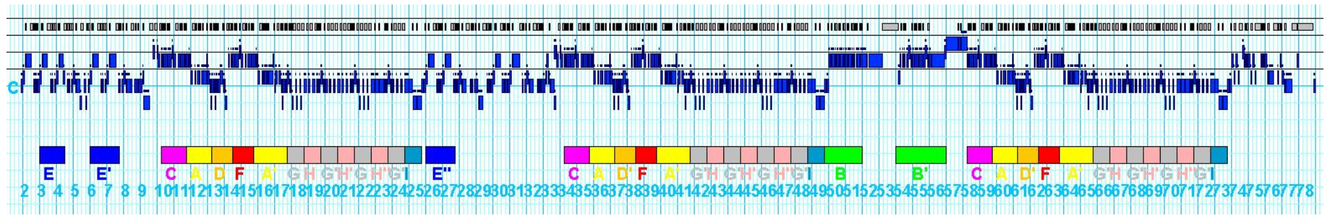

Fig. 9. Segmentation towards the last generation