Design and Comparison of Different Evolution Strategies for Feature Selection and Consolidation in Music Classification

I. Vatolkin, *Graduate Student Member, IEEE*, W. Theimer, *Senior Member, IEEE*, and G. Rudolph

***Abstract*— Music classification is a complex problem which has gained high relevance for organizing large music collections. Different parameters concerning feature extraction, selection, processing and classification have a strong impact on the categorization quality. Since it is very difficult to design a deterministic approach which provides the efficient parameter tuning, we haven chosen a heuristic approach. In our work we apply and compare different evolution strategies for the optimization of feature selection and consolidation using three pre-defined personal user categories. Concepts of local search operators with domain-specific knowledge and self-adaptation are examined. Several suggestions based on an empirical study are discussed and ideas for future work are given.**

I. Introduction

Portable devices have emerged as one of the preferred platforms for music listening during the past years. Most of the music is available in digital formats and the memory capacity is doubling every year. This leads to the situation that a user can now carry his or her complete music collection in a mobile device. The audio quality even in very small devices can be excellent. But the user interaction poses severe challenges due to the small physical size of mobile terminals: The display size is constrained and the input posibilities via keypads or touchscreens are limited.

In large music collections of 10000 or more music tracks it is very time consuming and suboptimal to manage *textual menu lists* such as artist or track lists with thousands of entries on a device the size of your palm. This is one motivation for our work to organize large music collections more intuitively. Another observation is that users have a very *personal* taste of music. The pre-defined music categories such as genres or sub-genres might help, but do not take personal listening preferences into account. We argue that music listeners should be able to define their *personal categories* of music by giving examples and let an intelligent music classification system rank the complete database according to these user preferences. In our approach we investigate how far the music taste of a person can be approximated from the audio content of a small number of music examples, prototypes that perfectly match a personal category and also counter-examples that are the complete "opposite" of this personal music category.

In the following section we introduce the music classifcation problem in more detail and define the performance criteria for optimal classifiers.

I. Vatolkin and G. Rudolph are with the Department of Computer Science, Technical University of Dortmund, Germany, W. Theimer is with Research In Motion, Bochum, Germany (email: igor.vatolkin@cs.tu-dortmund.de, wolfgang.theimer@ieee.org, guenter.rudolph@tu-dortmund.de).

The training of the classifier depends on many parameters that cannot be computed analytically. In order to achieve an optimal classifier performance we introduce evolutionary algorithms in section III. The parameters of the feature extraction, processing and classification are adjusted based on a fitness function that optimizes the classifier performance. In the first two subsections the evolution strategy is outlined and our modifications are described. In the third subsection the experiments are sketched.

In section IV the empirical results of our optimization algorithm are presented. The numeric results of the evolution strategy are shown and general findings are discussed. The paper ends with a concluding section and gives an outlook.

II. Music Classification Overview

A. Music Feature Extraction

A set of musical features is extracted from all music tracks. The N raw features $f_i, i \in [1, N]$ describe different characteristics of music, i.e., timbre, harmony, melody, rhythm, time and structural properties of music [19]. We used a set of up to 33 features defined in [17]. Since some of the features are represented by multidimensional vectors, the total number of used scalar values is 97. They are extracted in a sequence of M time windows for one track of music. The set of computed features for one time window forms a feature vector $\mathbf{f} = (f_1, f_2, ...f_N)^T$. Concatenation of the feature vectors for each time window creates a N x M feature matrix $\mathbf{F}$, where the number of columns is equal to the number of time windows (see Eq. 1):

$$\mathbf{F} = \begin{pmatrix} f_1(t_1) & f_1(t_2) & \dots & f_1(t_M) \\ f_2(t_1) & f_2(t_2) & \dots & f_2(t_M) \\ \vdots & \vdots & \ddots & \vdots \\ f_N(t_1) & f_N(t_2) & \dots & f_N(t_M) \end{pmatrix} \quad (1)$$

The timbre features and some harmony features are computed for 23 ms time windows (512 samples at a sampling rate of 22.05 kHz) with no overlap. But there are also more complex features that require longer time windows for the computation, like e.g. the fundamental frequency / pitch estimation or the melody analysis. Rhythm features are analyzed in time windows of several seconds providing sufficient signal statistics for a reliable estimate. Structural features of music are not yet taken into account, but typically are computed in even longer time intervals (in the order of 30 s) [9]. For details about the mathematical definitions of features refer to [10], [15], [18], [16].

978-1-4244-2959-2/09/$25.00

B. Reduction and Transformation of Features

The feature matrix **F** must be converted to one or more classifier input vectors. Methods operating on rows of the feature matrix normally seek to reduce the number of features. Principal component analysis [2] and linear discriminant analysis [7] are statistical methods to reduce the data dimensionality. In correlation-based feature selection highly correlated features are not submitted to the classifier. One can start with an empty group of features adding one-by-one a new feature which is least correlated with the existing feature set. Another possibility is to start with the full set of features and to take away the mostly correlated feature one-by-one until the desired number of features is reached [6], [5]. Sequential forward selection starts with an empty group of features identifying new features best improving classification performance in each iteration step.

Another group of methods operates on the columns of the feature matrix. The information of tatum and beat times in a music piece can be used for subsampling the feature matrix. Tatum times measure the perceived duration of the shortest note [16]. If tatum reduction is applied, only the features from time windows at tatum times or exactly in the middle between adjacent tatum times are saved for further classification since notes are changing on that time scale. Longer partitions are built by dividing a track in parts of equal length (e.g. 5 seconds). Each partition provides a single classifier input vector.

Finally, the pruned feature matrix is converted to one or several classifier input vectors. Each feature vector is mapped to the corresponding ground truth. The ground truth is a user-specified rating how similar a music track is to the current music category and ranges between 0 (no similarity at all) and 1 (matches perfectly with the music category). In our experiments we use a first-order Gaussian model, saving mean value and standard deviation over time from each feature. Alternatively more complex Gaussian models using larger number of Gaussian bell-shaped curves or methods which calculate the optimum number of Gaussians can be applied [13].

C. Classification of Music Categories

The ultimate target of our system is that the music classifier learns the user's musical taste and assigns the same similarity values to music tracks as the human user. In order to train the classifier and test the performance different users are asked to rank $L = 1139$ music tracks by similarities to their personal music category (reference input) $s_{i,ref}(n) \in [0, 1]$, for i-th song and n-th category. Each user has to identify 10 music tracks as prototypes for one personal music category n ($s_{i,ref}(n) = 1$) and 10 tracks as counter-examples of the music category ($s_{i,ref}(n) = 0$). These 20 songs are used for training. It is not realistic to assume that users rate much more music tracks to define a music category since this is a high effort. Only for evaluation purposes our test users also rank the remaining 1119 tracks of music by assigning one of four discrete similarity values to them: $s_{i,ref}(n) = 1$ - music fits perfectly to music category (but is not used as positive example for training), $s_{i,ref}(n) = \frac{2}{3}$ - good, but not perfect fit with music category, $s_{i,ref}(n) = \frac{1}{3}$ - weak similarity with category, $s_{i,ref}(n) = 0$ - music does not fit at all to personal music category (but is not used as counter-example for training). We used an even number of similarity values in order to not allow users to choose an average rating thus forcing them to make a decision. Providing finer or even continuous ranking values easily leads to confusion: People most likely do not remember their previous choice for similar songs.

The output of the classifier $s_i(n)$ is compared to $s_{i,ref}(n)$. An optimal classifier would perfectly approximate the reference similarities and the deviations between $s_{i,ref}(n)$ and $s_i(n)$ would be zero for each music track. In reality there are differences and as measure for the classifier performance the normalized mean squared error is chosen:

$$E^2 = \frac{1}{E^2_{max}} \frac{1}{L} \sum_{i=1}^{L} (s_i(n) - s_{i,ref}(n))^2 \qquad (2)$$

$$\text{with } E^2_{max} = \frac{1}{L} \sum_{i=1}^{L} e^2_{i,max}(n) \text{ and}$$

$$e_{i,max}(n) = \begin{cases} 1 & : \; s_{i,ref}(n) = 0 \vee s_{i,ref}(n) = 1 \\ \frac{2}{3} & : \; s_{i,ref}(n) = \frac{1}{3} \vee s_{i,ref}(n) = \frac{2}{3} \end{cases}$$

The normalization of the error by the divisor E^2_{max} is needed since the maximum possible error with regard to a value $s_{i,ref}(n)$ is $\max(s_{i,ref}(n), 1 - s_{i,ref}(n))$.

The purpose of classification algorithms is to relate music songs to given categories, allowing each song to be member of several categories. In our work we use supervised learning techniques for music classification, which learn from given labeled data (ground truth). For details and an overview of classifiers, see [3], [2].

The *divide-and-conquer* algorithm builds decision trees which consider only one most important feature in each node. The importance of every feature is estimated on the basis of an entropy measure. Several enhancements of divide-and-conquer lead to an algorithm named C4.5 [12]. It typically constructs very compact decision trees.

The experiments described later deploy C4.5 decision trees. We decided to use this classifier for several reasons. At first it is a complex method with integrated feature selection technique with sufficient performance for different data mining tasks. The runtime is more efficient in comparison to support vector machines which are often a first choice but require longer times for the transformation into a higher-dimensional space. The fact that the feature dimensions are not changing during the classification helps to interpret the meaning and impact of the low-level features. The parametrization of C4.5 itself is very simple and no tuning of the classifier itself is required. Besides, feature design and processing are essential for a classification task (as underlined in [8] for music categorization), and the optimization of a classifier itself is a very hard problem

which can be successful only if the input feature data has been prepared very thoroughly.

III. Algorithm Design and Test Plan

A. ES Design

We apply a (1+1)-ES for the feature selection and the choice of partition size, which defines an interval from a music piece for feature consolidation. The individual representation I for ES consists of a binary vector $\mathbf{m}$ with $m_i = 1$ for all selected features and an integer T_p for the partition size in ms:

$$\begin{aligned} I &= \{\mathbf{m}, T_p\} \\ \text{with } \mathbf{m} &= m_1 m_2 ... m_N, \;\; m_i \in \{0,1\} \\ \text{and } T_p &\in [500, 30000] \cap \mathbb{Z} \end{aligned} \tag{3}$$

The optimization problem can be described as:

$$\text{minimize } E^2(\mathbf{m}, T_p) \text{ with } T_p \in [500, 30000] \tag{4}$$

Since we take 33 features into account, the number of all possible individual representations is $2^{33} \cdot 25501$ and would require a very large amount of time for the analysis of all possible configurations.

For the first ES designed in [20] we implemented two mutation operators. The first one flipped each bit in $\mathbf{m}$ with a probabilty $1/N$. The second one changed T_p according to the integer mutation operator introduced in [14]. The expected mean mutation step size was adjusted by an exponential function S_g which started with an expected step size equal to 15000 and ended in the 100th generation with an expected step size equal to 1 ($g \in [1, 100]$ is the current generation number):

$$S_g = 15000 \cdot 0.96^g + \frac{1 - 15000 \cdot 0.96^g}{100} \cdot g \tag{5}$$

Another possibility to adjust the expected mutation step size is to use self-adaptation according to the $1/5$-rule [1]. Here the number of successful mutations N_s is counted during five generations and the new expected mean mutation step size S_g is calculated from the previous step size S_{g-5}:

$$\begin{aligned} \text{if } \; g = 5k, k \in \mathbb{N} : S_g &= \begin{cases} \alpha \cdot S_{g-5} & : \quad N_s > 1 \\ S_{g-5} & : \quad N_s = 1 \\ \frac{1}{\alpha} \cdot S_{g-5} & : \quad N_s = 0 \end{cases} \\ \text{else } : \; S_g &= S_{g-1} \end{aligned} \tag{6}$$

We used a recommended value $\alpha = 1.224$. In comparison to the ES with a mutation adjustment by the exponential function here the new expected mean mutation step size was changed only after each 5th generation.

For the comparison of algorithms below we use the abbreviations FA-ES (*functionally adjusted ES*) and SA-ES (*self-adaptation ES*).

B. Hybrid ES Design

A hybrid ES evaluates a local neighborhood during the evolutionary loop. At first the operator(s) for a local search should be designed. We considered to utilize the correlation between features. It is obvious, that some subsets of the computed music features may correlate more strongly than other subsets, so that some of features may be removed from the classifier input. A correlation coefficient between feature vectors $\mathbf{f}_i$ and $\mathbf{f}_j$ is at first averaged over the complete song, where $\mathbf{f}_i(k)$ is a vector of values of feature i for all time windows of song k:

$$\rho(\mathbf{f}_i, \mathbf{f}_j) = \frac{1}{L} \sum_{k=1}^{L} \frac{Cov\left(\mathbf{f}_i(k), \mathbf{f}_j(k)\right)}{\sqrt{Cov\left(\mathbf{f}_i(k), \mathbf{f}_i(k)\right) Cov\left(\mathbf{f}_j(k), \mathbf{f}_j(k)\right)}} \tag{7}$$

Now we distinguish between the features $\mathbf{f}_\mu$, which are marked for selection in the current individual ($\forall \mu : m_\mu = 1$) and the features $\mathbf{f}_\lambda$, which are marked as not selected for music classification ($\forall \lambda : m_\lambda = 0$). A *correlation rank* $R(\mathbf{f}_i)$ describes the averaged correlation of feature i to the features which are currently selected (M is the number of currently selected features):

$$R(\mathbf{f}_i) = \frac{1}{M} \sum_{\mu=1}^{M} \rho(\mathbf{f}_i, \mathbf{f}_\mu) \tag{8}$$

The first local search operator LS^+ sorts the not selected features according to $R(\mathbf{f}_\lambda)$ and adds a feature with the smallest rank to the set of selected features. In other words, we add a feature which at least correlates with the features which have been already used for music classification. The second local search operator LS^- reduces the number of the selected features sorting them according to $R(\mathbf{f}_\mu)$ and disables the selection of a feature with the highest rank, i.e. the feature which is mostly correlated with the others.

The Variable Neighborhood Search (VNS) procedure is described as following:

```
01 // Choose randomly current LS operator
02 LSadd = true;
03 LSsub = true;
04 while (LSadd OR LSsub) {
05   // Apply current LS operator
06   if (application successful) {
07     // Save the new individual
08     // Activate another LS operator
09   } else {
10     // Deactivate current LS operator
11     // Switch current LS operator
12   }
13 }
```

The goal is to switch between the two different LS operators as long as they provide better fitness values. In the line 01 it is chosen randomly if LS^+ or LS^- will be applied at first. Lines 02 and 03 activate the application of the both operators. The VNS loop continues as long as at least one operator is permitted for application. If both operators have

not been successful, the stop criterion becomes active. In the while loop the current operator is applied. If it has been successful (line 06), the new individual is saved and the other LS operator is activated. If the current operator has not been successful (line 09), it is deactivated (the local optimum is reached) and the other LS operator is switched to be the current operator. If the both local optima are reached one after another, the loop terminates.

Another consideration about the design of a hybrid ES is the VNS application frequency. One possibility is to apply VNS for each generation, creating a child individual with mutation operators and then performing the VNS procedure. However if a mutation creates a worse individual, several evaluations may be necessary during VNS to achieve the better fitness than the parent individual, if it is possible at all. Therefore another hybrid strategy applies VNS only after a successful mutation. We call the ES algorithm with the first hybridization paradigm AEG Hybrid (*after each generation*) and the second one ASM Hybrid (*after successful mutation*).

The last proposal is to consider the convergence behavior of the different ES runs. A non-hybrid ES converges fast at the beginning since the strong mutation operator allows the quick detection of promising regions. On the other side, the application of VNS permits the better search of local optima. Therefore it may be valuable to start with a non-hybrid ES and switch to a hybrid ES after some number of generations. We distinguish between 100%, 50% and 25% hybrid runs. 100% runs apply VNS from the first generation onwards. 50% runs start it after the 50th generation of the non-hybrid ES (we limit the generation number to 100 for non-hybrid runs) and 25% runs apply VNS after the 75th generation of the non-hybrid ES. Since a convergence behavior of FA-ES and SA-ES was not very different after the first performed experiments, we have decided to run the 50% and 25% hybrid runs only after FA-ES runs.

Thus, we have designed eight different ES algorithms to be compared: FA-ES, SA-ES (non-hybrid) and 100% AEG Hybrid, 50% AEG Hybrid, 25% AEG Hybrid, 100% ASM Hybrid, 50% ASM Hybrid and 25% ASM Hybrid.

C. Experimental Setup

The problem instances are three personal music categories ED, IV and WT from [20]. However we reduced the number of music songs to 75 in the test set because of the large evaluation run times. The comparison of experiments with 176 test songs (2 songs from each CD of our collection) and 75 test songs (one song per artist) showed, that the change in the problem complexity and optimal parameters was marginal.

Prior to the optimization we reduced the number of time windows with a tatum pruning method. Here only the feature values from the time windows exactly in the middle between tatum times were processed. We limited the evaluation number for the eight ES algorithms to 100. Each experiment setup was run 10 times for the calculation of the median and standard deviation.

IV. Empirical Results

A. Experimental Outcome and Interpretation

Table I summarizes the results after 100 evaluations for each algorithm. Figures 1, 2 and 3 depict the process of fitness change during the evaluations plotting the median values over 10 runs for each 10th generation.

TABLE I
OPTIMIZATION RESULTS AFTER 100 EVALUATIONS, NORMALIZED MEAN SQUARED ERROR $E^2 \cdot 10^2$ (MEDIAN AND MEAN DEVIATION FROM MEDIAN)

	ED	IV	WT
FA	17.20 ± 00.69	05.87 ± 00.28	04.65 ± 00.38
SA	17.27 ± 00.71	06.80 ± 00.60	04.63 ± 00.23
AEG 100%	17.79 ± 01.13	08.13 ± 00.48	04.86 ± 00.72
AEG 50%	17.09 ± 00.83	06.03 ± 00.24	04.85 ± 00.39
AEG 25%	17.80 ± 00.87	05.87 ± 00.30	04.79 ± 00.48
ASM 100%	$\mathbf{16.46 \pm 00.71}$	05.87 ± 00.48	$\mathbf{04.57 \pm 00.46}$
ASM 50%	16.78 ± 00.47	05.89 ± 00.27	04.65 ± 00.36
ASM 25%	17.84 ± 01.00	$\mathbf{05.83 \pm 00.29}$	04.74 ± 00.51

The used personal categories are clearly different in their complexity. The characteristics of the best individuals ever found during 80 experiments per each problem (8 algorithms $\times$ 10 runs) are given in the Table II. ED seems to be the toughest classification problem, whereas IV and WT are easier to classify.

TABLE II
DETAILS OF THE BEST INDIVIDUALS FOUND

	ED	IV	WT
$E^2 \cdot 10^2$	15.80	05.45	04.04
T_p	1235	1483	23969
N	16	12	13
found by	ASM 100	FA	FA

Several trends can be observed. At first, ASM Hybrid outperforms AEG Hybrid in almost all cases. The application of variable neighborhood search after a mutation with a worse outcome requires a number of evaluations and cannot guarantee a better outcome than the parent individual.

If we compare the non-hybrid algorithms, self-adaptation ES is worse than FA-ES. However it outperforms FA-ES slightly for WT category.

The 100% ASM Hybrid performs well for all personal music categories achieving the best results for categories ED and WT. Since the advantage of 100% ASM Hybrid for ED category was larger, the choice of a hybrid strategy for more complex categories may be recommended. For an easier music classification task a simple non-hybrid ES may be considerable. More experiments with different categories of different categories are required.

The runs with hybrid methods started after 50 or 75 FA-ES generations were not successful for the most configurations. Although there are some differences between the algorithms on the right subplots of Figures 1-3, they are rather small.

For ED, where 100% ASM Hybrid performed very well in general, the application of this method after 50 generations of FA-ES brings measurable improvement. Also a slight advancement was observed for 25% ASM Hybrid applied on IV category.

No specific recommendation for the feature set for different classification tasks can be made, as it has been already shown in [11]. But for the partition size and the overall number of used features we can observe some trends from the experimental results and Table II. As the complexity of the problems falls, the larger partition sizes are better and the number of the required features is smaller. Certainly the simpler music categorization like classic vs. rock does not require many features, and it allows the aggregation of the features over larger partitions. For further investigations of this hypothesis we plan to start more experiments with different categories in the near future.

B. Analysis of the Music Category Examples

For the better interpretation of the structure of the category instances we performed the analysis of the best individuals ever found during 80 runs for each category. Left subplots of Figures 4-6 depict the experiments where we added features with LS^+ operator and reduced feature number with LS^- operator starting from the best found individual. It is marked with a vertical line.

One can clearly see that a very small number of features is not sufficient for the classification. Too many features are also not very efficient. Since the decision trees already reduce the number of the features and prune the trees, different numbers of features may produce the same fitness values as a consequence. This is supported by the existence of plateaus in the figures. However the feature pruning mechanism of C4.5 seems to have its limitations, since the adding of features one-by-one due to their correlation rank leads to worse fitness values at increasing feature numbers. Here the overtraining of the classifier happens. Therefore the feature pre-selection makes sense even if a complex classifier with the integrated selection strategy is applied. ES can be applied for this task.

Another issue is that the application of LS operators from the optimum value implies not only a monotonic change of feature number, but also almost always the monotonic increase of the error metric. It confirms the usefulness of the designed operators.

We analyzed also the changes in partition size with a frozen feature configuration starting from the best found solutions (see right subplots of Figures 4-6).

The functions which describe the impact of the partition size for a given music category and feature set seem to have very noisy progress but also almost linear general trends. It is obvious, that the very small partition sizes cannot capture meaningful music characteristics and produce also too many inputs which may lead to overtraining of the classifier. On the other side very large partitions correspond to audio segments with a lot of different music events and are also not optimal. Interestingly a compromise is very different for the examined categories. ED and IV solutions are at best if small partitions are used (see Table II for the exact details), and WT individuals produce smaller classification errors with rather large partitions. A suggestion here is that the simpler categories which distinguish between very different music categories (e.g. classic vs. rock) are more robust at mixing features over larger partitions and producing the smaller number of the input instances for the classifier. Since no general recommendation about the optimal partition size can be made, here it is again a parameter which can be tuned by ES.

C. Comparison of Algorithms Starting with the Same Individuals

The experiments described in section IV-A were performed starting with the randomly generated individuals. Another possibility for the algorithm comparison is to start from the same individuals. This situation is more artificial but it provides some interesting results if the start individuals of the different complexities are taken into account.

We generated 100 solutions at random for each problem and saved the individuals with the minimum and maximum values. The goal was to compare the algorithms which start from solutions with different quality. An individual with the minimum fitness value can be harder to optimize if it corresponds to a local optimum. Start from an individual with the maximum fitness value is easier if local optima should be avoided, but it is obviously not a very good solution at the beginning. FA-ES, SA-ES and 100% ASM Hybrid provided the best results during the previous experiments, so we made five runs for these three methods. The results are outlined in Table III.

TABLE III
OPTIMIZATION RESULTS AFTER 100 EVALUATIONS, NORMALIZED MEAN SQUARED ERROR $E^2 \cdot 10^2$ (MEDIAN AND MEAN DEVIATION FROM MEDIAN)

	start	ED	IV	WT
FA	min	16.66 ± 00.49	05.81 ± 00.08	04.02 ± 00.09
SA	min	16.82 ± 00.17	05.99 ± 00.01	04.20 ± 00.01
ASM	min	16.40 ± 00.37	05.96 ± 00.13	04.07 ± 00.01
FA	max	19.02 ± 00.39	05.52 ± 00.18	04.64 ± 00.40
SA	max	17.66 ± 00.61	05.65 ± 00.12	04.61 ± 00.34
ASM	max	16.67 ± 00.44	05.44 ± 00.30	05.28 ± 00.67

The results are comparable to the previous experiments. ASM Hybrid is again the best algorithm for ED category (both minimum and maximum starts). The worst performance of ASM Hybrid was for the 'easiest' individual to optimize, namely WT maximum. FA-ES is in most cases better than SA-ES. Therefore we can carefully support here the conjecture that FA-ES performs better for simpler categories, ASM Hybrid for more complex categories, and SA-ES is slightly worse than FA-ES. At any rate, more statistics and also classification problems are required for the more detailed analysis of these algorithms.

V. Conclusion and Outlook

In our work we concentrated on the two important parameters of the music classification chain, namely the features to select and the size of a music partition for the feature consolidation. Several ES algorithms were designed using the different techniques (self-adaptation, hybridization with local search, operators with domain-specific knowledge).

Several trends can be observed for the comparison of methods and optimized parameter settings. For the algorithms, no clear winner was found in the course of our experiments, however the ASM Hybrid performed well for IV and WT categories and was the best for the most difficult category ED. The parameter optimization for the simpliest category was done at best with a standard ES with mutation adjustment by a function. For the hybrid algorithms, the application of the VNS scheme seems to be reasonable only after successful mutations. Running several steps of the LS operators for each new offspring requires too much evaluation time. Also, it did not offer performance progress except of two cases running a hybrid method after some pre-optimization by e.g. FA-ES.

No concrete recommendation for the optimized parameters for the different categories can be made, as already have been shown in our previous and other works. Therefore parameter tuning by evolutionary algorithms is a resonable step if smaller classification errors are desired. However some general trends can be observed. Too few or too many features should not be used, as well as too small or too large partitions. Further suggestion is that simpler classification problems may be categorized better with larger partitions and they require fewer features. More experiments with different music categories are necessary.

Here are some suggestions for further work: As mentioned before, the better understanding of such a complex optimization problem like music categorization requires large empirical studies. We want to extend our category set using more personal categories and also genre classifications. The complete feature processing chain can be extended and optimized by ES. We want to add some new low-level features from the latest research activities. A possibility to switch on or off single dimensions of multi-dimensional features (e.g. using only a couple from 13 MFCC coefficients) has been considered, but on the other side produces a more complex representation and makes the optimization a harder task. We plan to add further feature selection methods, using some statistical analysis like PCA or LDA. Finally also the algorithms can be tuned. New and also domain-specific operators, meta-optimization and other methods like swarm-based optimization are promising areas for future work.

References

[1] Bäck, T. Hoffmeister, F.: Basic Aspects of Evolution Strategies, Statistics and Computing vol. 4 (2), pp. 51-63 (1994)

[2] Bishop, C.M.: Pattern Recognition and Machine Learning, Springer (2006)

[3] Duda, R., Hart, P., and Stork, D.: Pattern Classification (2nd Edition), Wiley-Interscience (2000)

[4] Flexer, A., Gouyon, F., Dixon, S., and Widmer, G.: Probabilistic Combination of Features for Music Classification, in Proc. of the 7th International Conference on Music Information Retrieval (ISMIR), pp. 111-114 (2006)

[5] Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, L.: Feature Extraction, Foundations and Applications, Springer (2006)

[6] Hall, M.: Correlation-based feature selection machine learning. PhD thesis, University of Waikato, New Zealand (1998)

[7] McLachlan, G.: Discriminant Analysis and Statistical Pattern Recognition, Wiley Interscience (1992)

[8] Mierswa, I., Morik, K.: Automatic Feature Extraction for Classifying Audio Data. Machine Learning Journal, 58:127–149 (2005)

[9] Ong, B.: Structural Analysis and Segmentation of Music Signals. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain (2006)

[10] Peeters, G.: A Large Set of Audio Features for Sound Description (Similarity and Classification) in the CUIDADO Project. IRCAM, France (2004)

[11] Pohle, T., Pampalk, E., Widmer, G.: Evaluation of Frequently Used Audio Features for Classification of Music into Perceptual Categories, in Proc. of the 4th International Workshop on Content-Based Multimedia Indexing (2005)

[12] Quinlan, J.R.: C4.5: Programs for Machine Learning, Morgan Kaufmann (1993)

[13] Rasmussen, C.: The Infinite Gaussian Mixture Model, in Advances in Neural Information Processing Systems, MIT Press, vol. 12, pp. 554-560 (2000)

[14] Rudolph, G.: An Evolutionary Algorithm for Integer Programming, in Proc. of the 3rd International Conference on Parallel Problem Solving from Nature (PPSN), Jerusalem, pp. 139-148 (1994)

[15] Scaringella, N., Zoia, G., and Mlynek, D.: Automatic Genre Classification of Music Content, IEEE Signal Processing Magazine, vol. 23, pp. 133-141 (2006)

[16] Seppänen, J., Eronen, A., and Hiipakka, J.: Joint Beat and Tatum Tracking from Music Signals, in Proc. of the 7th International Conference on Music Information Retrieval (ISMIR), Victoria, pp. 23 - 28 (2006)

[17] Theimer, W., Vatolkin, I., and Eronen, A.: Definitions of Audio Features for Music Content Description, Algorithm Engineering Report TR08-2-001, Technische Universitt Dortmund (2008)

[18] Tzanetakis, G. and Cook, P.: Musical Classification of Audio Signals, IEEE Transactions on Speech and Audio Processing, vol. 10, pp. 293-302 (2002)

[19] Vatolkin, I. and Theimer, W.: Introduction to Methods for Music Classification Based on Audio Data, Technical Report NRC-TR-2007-012, Nokia Research Center (2007)

[20] Vatolkin, I. and Theimer, W.: Optimization of Feature Processing Chain in Music Classification by Evolution Strategies, in Proc. of the 10th International Conference on Parallel Problem Solving from Nature (PPSN), Dortmund, pp. 1150-1159 (2008)

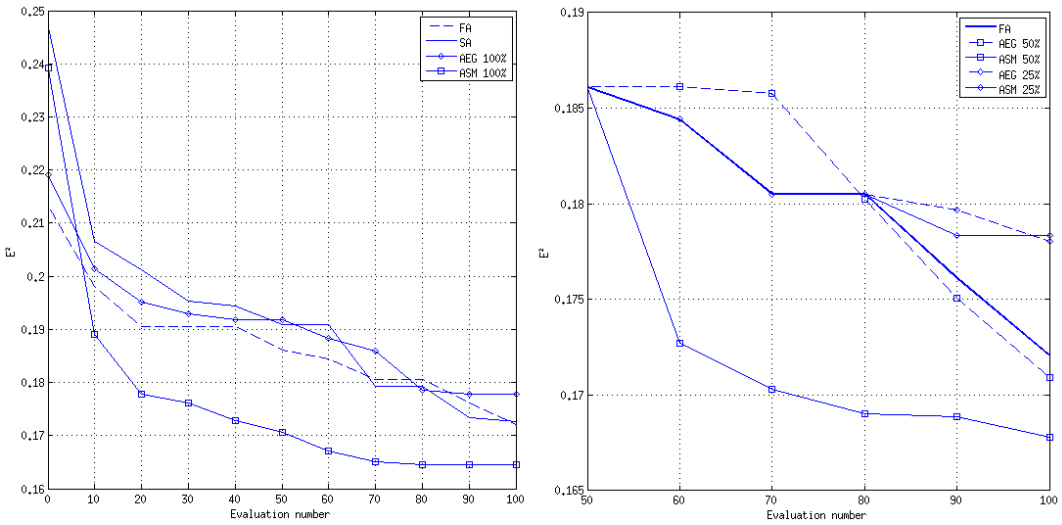

Fig. 1. Category ED: Median E^2 values for evaluation numbers of different algorithms

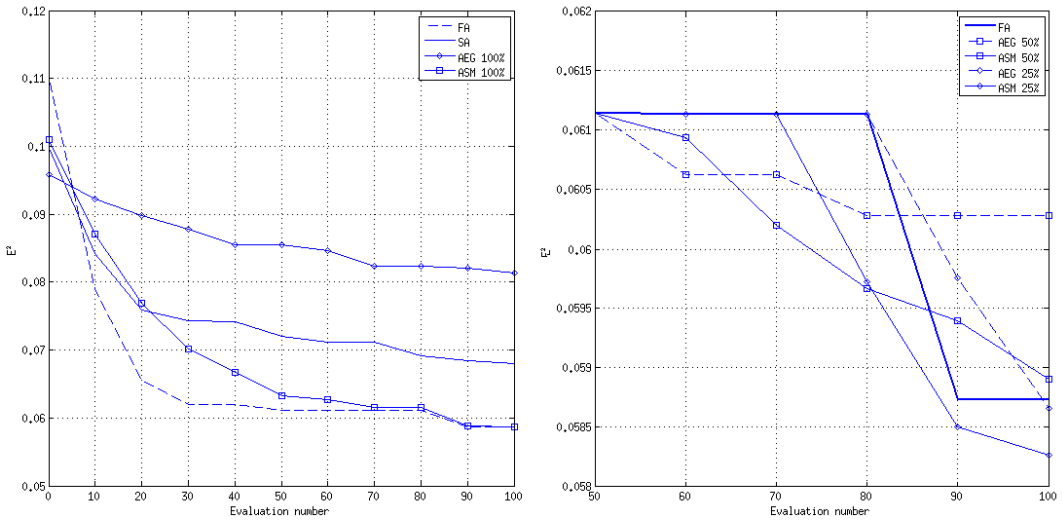

Fig. 2. Category IV: Median E^2 values for evaluation numbers of different algorithms

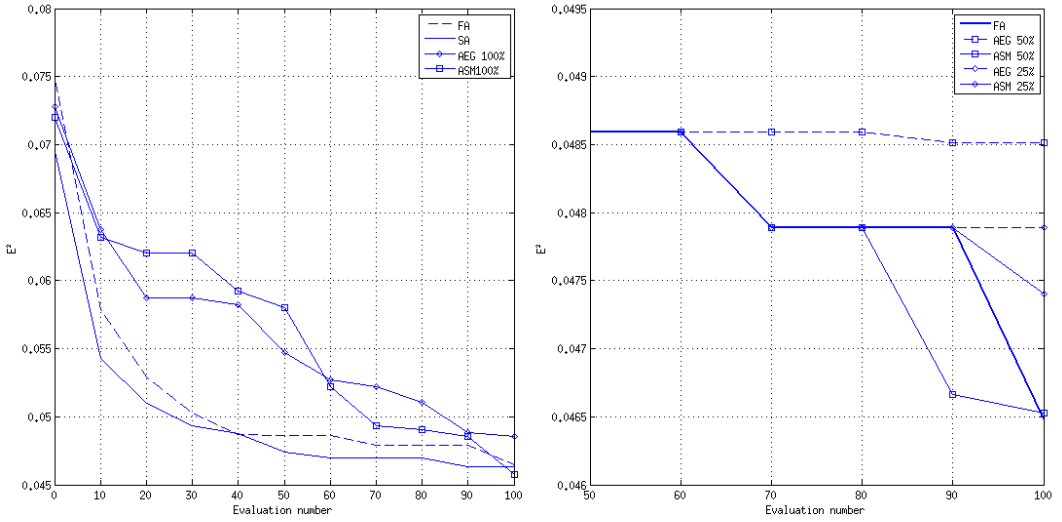

Fig. 3. Category WT: Median E^2 values for evaluation numbers of different algorithms

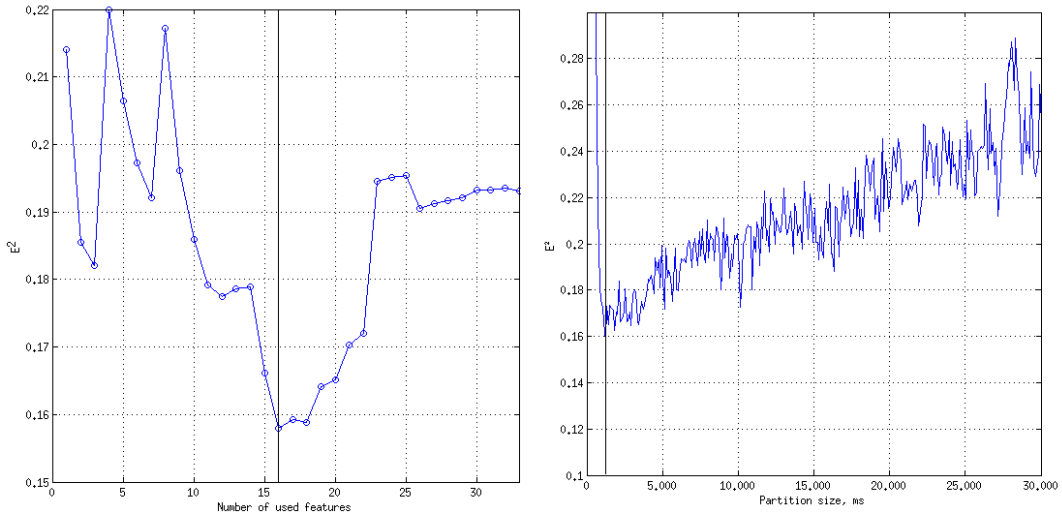

Fig. 4. Category ED: E^2 values for different numbers of selected features (left) and partition sizes (right) starting from the best individual found

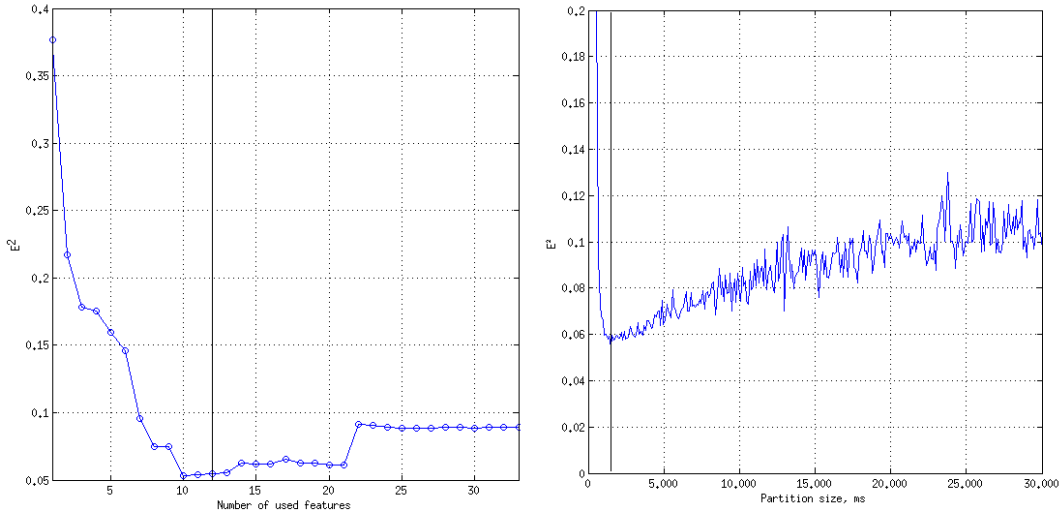

Fig. 5. Category IV: E^2 values for different numbers of selected features (left) and partition sizes (right) starting from the best individual found

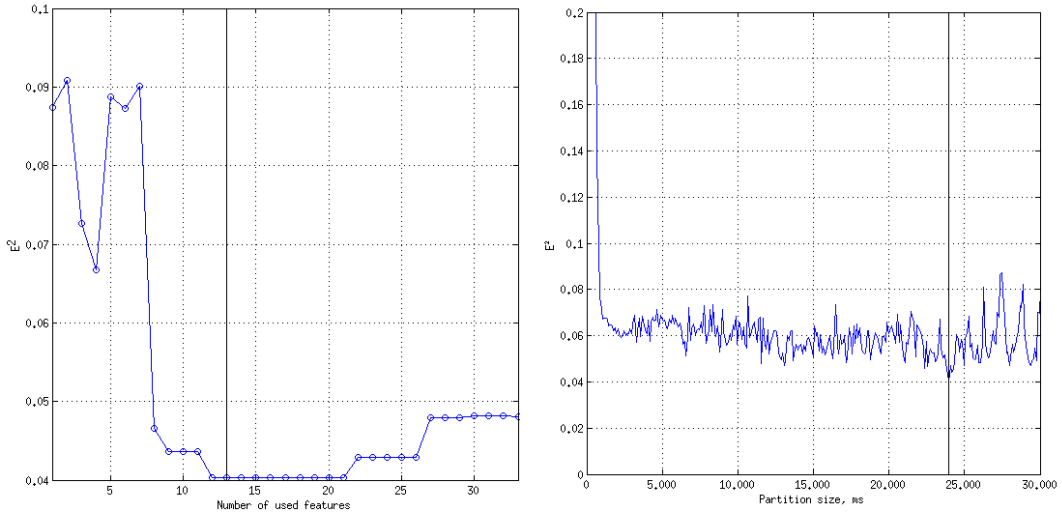

Fig. 6. Category WT: E^2 values for different numbers of selected features (left) and partition sizes (right) starting from the best individual found