Mäkinen et al. EURASIP Journal on Audio, Speech, and Music Processing 2012, 2012:23
http://asmp.eurasipjournals.com/content/2012/1/23

EURASIP Journal on
Audio, Speech, and Music Processing
a SpringerOpen Journal

RESEARCH Open Access

An evolutionary feature synthesis approach for content-based audio retrieval

Toni Mäkinen*, Serkan Kiranyaz, Jenni Raitoharju and Moncef Gabbouj

Abstract

A vast amount of audio features have been proposed in the literature to characterize the *content* of audio signals. In order to overcome specific problems related to the existing features (such as lack of discriminative power), as well as to reduce the need for manual feature selection, in this article, we propose an evolutionary feature *synthesis* technique with a *built-in* feature selection scheme. The proposed synthesis process searches for optimal linear/nonlinear *operators* and feature *weights* from a pre-defined multi-dimensional search space to generate a highly discriminative set of new (artificial) features. The evolutionary search process is based on a stochastic optimization approach in which a multi-dimensional *particle swarm optimization* algorithm, along with *fractional global best formation* and heterogeneous particle behavior techniques, is applied. Unlike many existing feature generation approaches, the dimensionality of the synthesized feature vector is also searched and optimized within a set range in order to better meet the varying requirements set by many practical applications and classifiers. The new features generated by the proposed synthesis approach are compared with typical low-level audio features in several classification and retrieval tasks. The results demonstrate a clear improvement of up to 15–20% in average retrieval performance. Moreover, the proposed synthesis technique surpasses the synthesis performance of evolutionary *artificial neural networks*, exhibiting a considerable capability to accurately distinguish among different audio classes.

Keywords: Content-based retrieval, Evolutionary computation, Particle swarm optimization, Feature selection, Feature generation

Introduction

Due to the drastically increased amount of multimedia data available in the Internet and in various public and personal databases, the development of efficient indexing and retrieval methods for large multimedia databases has become a widely studied research topic. Scientific fields, such as digital signal processing (DSP) and computer science (particularly *machine learning*), provide efficient and mathematically well-defined methods for data mining and knowledge discovery from specific observations or databases [1]. One of the major research foci in the field is concentrated around *content-based classification* using supervised learning methods. In these, a dataset together with its perceived class labels, known as "*ground truth*," is used to train a classifier, so that it can learn to discriminate among the individual classes in the training dataset. This enables the classifier to classify new, previously unseen data items with a certain degree of accuracy. Once successful such methods can be applied to several application areas, such as advanced database browsing, query-by-example retrieval, highlight-spotting from movies and/or sport events, speaker recognition, and so on.

In general, whenever machine learning techniques are to be applied to data classification or clustering tasks, certain *features* need to be extracted from the data. The features can be numerical or nominal scalars or vectors describing specific characteristics of the data such as, in the case of audio signals, *tonality* or *fundamental frequency* (FF). Because the data classification and mining methods are strongly dependent on the extracted features, their quality and discriminative capability have an obvious influence on overall classification performance. Unfortunately, despite the enormous number of different audio feature extraction methods available in the

* Correspondence: toni.makinen@tut.fi
Department of Signal Processing, Tampere University of Technology, P.O. Box 553, Tampere, Finland

literature, the features have limitations and drawbacks in describing the data content, so that the current audio classifiers cannot really compete with the human auditory perception system. As will be shortly reviewed, such a lack of semantic representation, the "*semantic gap*," has led to developing several promising techniques to obtain more power of discrimination from the extracted low-level features. The related work in this field is presented in the following section, which focuses on the two most important feature enhancement methods in the literature, *feature selection* and *feature synthesis* (also known as feature *generation/construction/transformation*).

Related work

Generally in machine learning, it is desirable to work with low-dimensional feature vectors (FVs) to reduce computational complexity, and also to avoid the so-called *curse of dimensionality* phenomenon [2], which basically states that in high-dimensional representations the available data become too sparse for any decent statistical or structural analysis. In a feature selection scheme, the FV dimensionality is lowered by selectively choosing an *expressive* and *compact* set of features among a possibly much larger original set. Evolutionary algorithms, such as *genetic algorithms* (GAs) [3] and *genetic programming* (GP) [4], are encountered in several feature selection approaches in the literature (see, e.g., [5,6]). Recently, another population-based stochastic optimization algorithm, *particle swarm optimization* (PSO) [7], was used by Ramadan and Abdel-Kader [8]. They applied PSO to features extracted by discrete cosine transform and discrete wavelet transform. The face-recognition results were comparable to GA-based feature selection but with the benefit of fewer features. Another PSO-based feature selection approach was presented by Chuang et al. [9] in which an *improved binary* particle swarm optimization was applied to a set of gene expression data classification problems. The highest classification accuracy was obtained in 9 out of the 11 tested gene expression problems. It was also reported that the average classification accuracy obtained by a *K*-nearest neighbor classifier was increased by 2.85% when compared to the previously published methods. Other types of classifiers, such as *support vector machines* (SVM) [10,11] and *back-propagation networks* [12], have also been tested with PSO-based feature selection in varying types of classification problems. Finally, in [13], a survey of several other feature selection methods was presented, leading the authors to conclude that "*applying first a method of automatic feature construction yields improved performance and a more compact set of features.*" Hence, research for generating completely new (or modified) features has gained more attention during the past few years.

To date, several feature generation approaches have been proposed [14], which have shown improvements over many types of classification problems. In one of the pioneer works, Markovitch and Rosenstein [15] proposed a framework for feature generation based on a *grammar* consisting of feature construction functions (such as arithmetic and logic operators). In their research, new features were *iteratively* constructed using decision trees, while the evaluation of the framework was done using the Irvine repository of (symbolic) classification problems. Improved classification results were obtained with several tested classifiers when applying them with the original and constructed feature sets (FS). However, such grammar-based methods lack the ability to generalize across more concrete and realistic cases, where, instead of symbols, the input data consists of raw signals. The challenge with signals is that there are no "universally good" features available; rather one has to *manually* choose and extract a specific set of features among the huge amount of existing possibilities. Thus, it can never be guaranteed that the selected features truly represent the optimal set of features for the problem at stake. To address the issue, a fascinating and rather recent idea of *automatic feature generation* has proven to be a promising approach, as it allows going beyond the limitations of human imagination in producing new transformations and (artificial) features.

Automated feature generation approaches are generally based on a "trial and attempt" type of methodology, meaning that *stochastic* searching and optimization algorithms are commonly applied. In [16], a combination of both feature selection and generation was proposed based on a modified GA. The algorithm was applied for the feature transformation process, and an inductive learner was used to evaluate the constructed features on an interpretation of chromatography time series. It was confirmed that one can significantly improve the learning performance when using the constructed features instead of the original time series data. The term "feature synthesis" was first used by Krawiec and Bhanu [17] in the context of object recognition. They applied linear GP to encode potential recognition *procedure synthesis* solutions, expressed in terms of elementary operations. The training consisted of co-evolving feature extraction procedures, each being a *sequence* of elementary image processing and feature extraction operations. The recognition accuracies obtained were comparable to those achieved by standard methods. Bhanu et al. [18,19] continued the work in the context of face expression recognition, where a Gabor-wavelet representation was used for primitive features and linear/nonlinear *operators* were selected among 37 different options to synthesize new features. Each individual in the applied GP algorithm was represented by a binary tree, each of which

corresponded to a single *composite* operator (consisting of several primitive operators). The operator selection was tuned using a Bayesian classifier, and the operator yielding the best classification accuracy was used in synthesizing new FVs for each image in the database. Improved classification accuracy with fewer features was obtained compared to results obtained using the original set of primitive features in the expression recognition task. The work was expanded in [20], where *co-evolutionary* processing was added to the approach to enable using *several sub-populations* in the GP algorithm. In this case, the final FVs were formed by combining the composite features synthesized by each individual subpopulation. The obtained classification results for synthetic aperture radar images showed occasional improvements compared to the primitive features; as before, fewer features were required to obtain comparable recognition rates. However, the authors also conclude that "*... it is still very important to design effective primitive features. We cannot entirely rely on CGP (co-evolutionary GP) to generate good features.*"

Probably the first *audio* feature generation system was one proposed by Pachet and Zils [21]. Their approach uses GP as the core feature generation algorithm in an *extractor discovery system* (EDS) framework, to explore large operator function space and to automatically discover new high-level audio features. The search is guided by specific heuristics, which enable applying knowledge representation schemes about signal processing functions as part of the feature generation process. More recently, Pachet and Roy [22] applied the same EDS framework, where *analytical features* (AF) were also introduced. These represent a large subset of all possible audio DSP functions, and are expressed as a functional term consisting of basic operators. The main idea in [22] is to apply genetic transformations in order to improve the current population of the (first random) AFs, while the fitness of each AF is evaluated using an SVM classifier. The idea of EDS bears some similarities to the framework proposed in [15] and some other feature generation approaches, but differs in providing operator knowledge (such as function *patterns* and *heuristics*) within the process. As a result, improved classification results compared to common audio features were obtained with the AFs in several challenging classification tasks. The authors also participated to the research made in [23] with AFs proposing a method to improve search performance involved in feature generation tasks. The applied algorithm is a variant of *simulated annealing*, guided by the so-called *spin patterns*, which are statistical properties of the feature space. Three audio classification problems were evaluated using the generated features, and significant improvements in execution time were reported when the results were compared to those obtained with features searched using GA, as described in [22].

Generally in audio signal processing, *ad hoc* domain-specific features have also gained considerable attention during the past few years, mainly applied to specific audio classification problems. For example, Mörchen et al. [24] constructed a large set of features by applying *cross products* between several existing short- and long-term (obtained by several aggregation methods) feature functions, resulting in approximately 40,000 audio features in total. It was shown that some of the constructed features could indeed improve music classification performance relative to conventional features. Another such example was presented by Mierswa and Morik [25], where *method trees* consisting of *ad hoc* features for a given audio signal were introduced. The trees were automatically generated with GP by combining elementary feature extraction *methods*. To do this, an additional complexity constraint was applied to keep the computational processing feasible. Improvements were reported in music genre classification accuracy over approaches with traditional audio features. Furthermore, in [26] the same approach was applied to speech emotion recognition with comparable results.

Considering potential drawbacks and uncertainties in the previously proposed feature generation approaches, an important issue relates to the computational time and *complexity* required for the synthesis process. More specifically, generating new FVs with *high dimensionality* may be laborious and time-consuming; for example, in [20], a separate subpopulation needed to be generated for each new generated feature. However, despite the increasing amount of computation required, also high-dimensional representations should be considered when striving to generate efficient new FVs. The *individual feature search* method, introduced in [23], provides a significant contribution to the field in decreasing computation time with respect to GP (by an order of magnitude). Due to the somewhat constrained set of experiments, however, eventually the method shows particularly significant differences to traditional GP mainly at the initial stage of the search, whereas the fitness difference becomes less significant as the search goes on. The allowed search space size in general plays an important role, as it may become too large to be explored efficiently (and throughout). For example, in the case of the AF proposed in [22], it was said that "*the space of 'reasonable size' AFs is huge*" (as it should be to allow capturing a sufficient collection of DSP functions), and, later, that "*eventually the EDS framework reaches the fringe of the space, although it certainly does not explore all of it.*" Now, depending on the case, such partial exploration might cause some lack of performance to the generated features, which is addressed in this article by applying two

dedicated techniques for converging towards the global optimum of the parameter search space. The EDS framework applied and discussed in [22] uses several *heuristics* as a vital component to guide the search. These may complicate the system implementation and, in some cases, cause additional uncertainty or fuzziness (due to stochastic behavior) to the process. Considering the combined feature selection and feature generation methods proposed so far, a *separate* feature selection scheme is generally required as a part of the main system topology, whereas the approach proposed in this article provides feature selection as a built-in property within the underlying PSO algorithm. This makes the overall design and implementation of the method easier, and possibly decreases the number of adjustable parameters. Finally, in some cases (e.g., in [20] or in [24] with most of the cases), it was demonstrated that the generated features cannot always improve the final classification results, which is an issue worth taking into a deeper discussion in order to discover valid reasoning for such synthesis behavior. It could be that the search for the optimal synthesis parameters in a high-dimensional solution space gets trapped into a local optimum, ultimately yielding deficient synthesis results. Another reason could be that the evaluation of the feature quality does not correlate with the actual performance obtained using the features. The problem might also relate to the *manually* selected dimension for the synthesized FV, which may serve as an apparent source of sub-optimality. Nonetheless, in the previously proposed feature generation approaches, the dimensionality issue is only rarely considered and discussed. In [22,27], separate classification experiments with different FS dimensions are performed and compared. Automatic, simultaneous, and *on-going* search and optimization regarding to output FV dimensionality, however, is a novel property provided by the synthesis technique proposed in this article.

The proposed feature synthesis technique

In this article, we aim to overcome the mentioned problems by proposing an evolutionary feature synthesis (EFS) technique based on PSO. The technique is applied for audio feature selection and synthesis. The main motivation for the work is to provide improved content-based audio classification and retrieval performance. To motivate the selection of PSO instead of the generally applied GA (or its derivatives), a comparison of the two algorithms, based on [9], is provided in Table 1. In short, the main advantage of PSO over GAs is that the algorithm provides more profound *intelligent background* [28], and it can be performed more easily than GAs [28]. Also, the computation time of PSO is usually less than for GAs, because all the particles in PSO tend to converge to the best solution rather quickly [29]. The synthesis approach presented in this article provides the ability to apply any fitness measure found appropriate for the final feature task at hand (such as classification). Furthermore, we apply a *multi-dimensional extension* of the basic PSO algorithm (MD PSO, [30]) to allow *dynamic* output FV dimensions. This avoids the need of fixing the dimension of the solution space (corresponding to the dimensionality of the synthesized vector) in advance, which is a property not considered in the audio feature generation methods published before.

In order to better avoid the problem of *premature convergence* related to the traditional PSO, a recent technique, the *fractional global best formation* (FGBF) suggested in [31], is also adopted within the proposed synthesis approach. A preliminary work was presented in [32], in which the performance improvement provided by the approach was tentatively verified in the context of images. Furthermore, in this article, a *heterogeneous particle behavior* approach, recently proposed by Engelbrecht [33], is considered. The approach provides further assistance for the particle swarm to converge to the global optimum of the search space by altering the particle velocity update rules. Hence, finally, as a combination of all the PSO extensions, in this study we propose applying an MD PSO algorithm with FGBF and heterogeneous particle behaviors for audio feature synthesis. To the best of the authors' knowledge, we are not aware that such an approach (or PSO in general) should have been proposed earlier in the audio feature generation field.

Table 1 Comparison of GA and PSO as search algorithms

Property	GA	PSO
Genetic operators	Included	Excluded
Key functions	Crossover	Social particle interaction
	Mutation	Particle velocity updates
Information source	All the chromosomes (the whole population moves in one group)	The *global best* particle (evolution only looks for the best solution)
Update occurrence	Probabilistic (cross-over and mutation *rates*)	All particles are updated after each iteration
Local optimum	Can become easily trapped	Can avoid well the local optima

The rest of the article is organized as follows. The low-level audio features considered in this research are described in "Low-level audio features" section, whereas in "Evolutionary optimization techniques" section, the underlying evolutionary optimization technique, the MD PSO, is introduced in detail. "EFS and selection" section presents an overview and technical details of the proposed feature synthesis approach, and the experimental results with comparative evaluations are shown and discussed in "Experimental results" section. Finally, "Conclusion" section concludes the article and discusses topics for future research.

Low-level audio features

In order to provide some important background information for the ultimate goal of this study, i.e., improving the audio retrieval performance, we start by introducing the applied original (low-level) audio features and the extraction procedures preceding the actual feature synthesis process. As mentioned in "Introduction" section, audio features play an essential role in content-based classification and retrieval tasks, so that selecting and extracting the "correct" features for the problem at hand is of utmost importance. Thus, the major focus is on synthesizing new features from the low-level audio features most typically used in the literature. This provides us a solid "baseline" FSs and allows us to demonstrate the effectiveness of the proposed synthesis technique.

Two separate audio feature extraction approaches are used to evaluate the performance of the feature synthesis with different types of features. In order to detect specific temporal signal characteristics, the considered audio clips are processed in short time *frames* of 40-ms duration. The first approach extracts *segment-based* features, while the second one is based on the so-called "*bag-of-frames*" approach [34], where the features are extracted directly from the short-time frames. The details of the extraction methods are provided in the following sections.

Segment features

The segment features are extracted based on the method introduced in [35]. The *energy levels* of the audio frames are computed, and then compared to the average energy level of the whole audio signal to detect and discard silent audio frames. In the second phase, seven audio FSs (specified on the left column of Table 2) are extracted from the non-silent frames, and consecutive *non-silent* frames are *merged* to form distinct audio segments. Hence, theoretically, an audio signal with no silent sections would be considered as a single segment. However, due to some background noise or environmental acoustics, occasional non-silent frames may occur in the middle of a silent section. To filter out such noisy frames in the process, an empirically determined threshold of *five* consecutive frames was set as a minimum duration for a signal segment. Finally, the actual audio segment features are formed by computing the *mean* (μ) and *standard deviation* (σ) statistics of *each* FS (including also the STAT features, i.e., means of means, etc.) over the formed segments. Thus, as an example, an audio signal consisting of four separate segments is represented by four corresponding segment FVs of each FS. For a more detailed description of the segment feature extraction, the reader is referred to [35].

Table 2 The extracted low-level audio features

Segment features	Key-frame features
STAT[a] (39-D)	MFCC + Δ-MFCC + ΔΔ-MFCC (39-D)
13 Mel-frequency cepstral coefficients (MFCC) (26-D)	12th-order LPC + 14th-order LPCC (26-D)
13 Δ-MFCC (26-D)	K_AUDIO[c] (31-D)
13 ΔΔ-MFCC (26-D)	
10th-order linear prediction coefficients (LPC) (20-D)	
14th-order linear prediction cepstral coefficients (LPCC) (28-D)	
S_AUDIO[b] (38-D)	

[a]STAT includes the mean (μ) and standard deviation (σ) values of signal *statistical features*, both in time and frequency domain: mean, variance, standard deviation, average deviation, skewness, kurtosis, and *also* the following *segment* features (μ,σ): band-energy ratio (BER), spectral centroid, transition rate, FF, irregularity (2 versions), flatness (both in linear and decibel scale), and tonality.
[b]S_AUDIO includes the following *segment* features (μ,σ): tristimulus, smoothness, spectral spread, spectral roll-off, RMS amplitude, inharmonicity, spectral crest, loudness, noisiness, power, odd-to-even ratio, and sub-band powers of six frequency bands.
[c]K_AUDIO includes the following *key-frame* features: irregularity (two versions), tristimulus, smoothness, spectral spread, zero-crossing rate, spectral roll-off, loudness, flatness (linear and decibel scale), tonality, noisiness, RMS amplitude, inharmonicity, spectral crest, odd-to-even ratio, spectral slope, FF, skewness, kurtosis, spectral skewness, spectral kurtosis, and 7-band sub-band powers.

Key-frame features

In the second feature extraction approach, the features are extracted directly from the short time frames. Due to this, the frames are first *Hamming*-windowed to avoid sharp discontinuities at the frame edges. Because there are many frames already in a single audio clip, a specific *key-frame* extraction method, proposed in [36], is applied to reduce the most *redundant* frames. Such redundancy occurs because many audio classes, such as music or speech, contain similar and almost identical sounds (such as common vowels or same notes of an instrument). In short, the main idea in the frame reduction approach is to partition the extracted frame features into distinct *clusters* (based on their similarity/distance between each other) and to select only one or few *key-frames* from each cluster to represent its corresponding sound. For this, a *minimum spanning tree* clustering algorithm is applied, which is detailed in [36]. As an outcome of the procedure, the overall amount of frames is

significantly decreased, whereas most of the feature description power is still maintained. The three frame-level FSs used in this study are listed on the right column of Table 2.

The extracted features of Table 2 represent an inclusive set of common audio features found from the literature. The feature implementations are based on a publicly available "*LibXtract*" library [37], although some of the segment features, such as dominant BER and segment FF, are extracted as described in [35]. The dimensions of the extracted FSs/vectors are also given in parenthesis for all sets in Table 2. For example, taking the mean and standard deviation of the 13th-order segment MFCC features results to a 26-dimensional FV. The feature values of each segment/key-frame—as well as their corresponding ground truth class labels and clip indices (describing from which audio file a particular FV is extracted from)—are stored in a single plain text file, so that the actual audio files are not needed anymore after the feature extraction phase. For specific definitions and formulas for the extracted features, the reader is referred to the audio signal processing literature (see, e.g., [38-40]).

Evolutionary optimization techniques

In this section, the PSO algorithm, its multi-dimensional extension, and the FGBF technique are introduced. The adaptation of *heterogeneous* particle behaviors within the MD PSO process is discussed in detail at the end of the section.

MD PSO

PSO was first introduced by Kennedy and Eberhart [7]. The PSO algorithm is a *population-based* optimization technique, in which a swarm of particles propagates in a pre-defined search space. Each individual particle, p, of the swarm represents a *potential solution* to an underlying optimization problem, in which the particles are evaluated using a proper *fitness function*, $\mathcal{F}[p]$. The PSO algorithm is specifically designed for solving nonlinear optimization problems. Due to the *diversity* associated with randomly distributed particles, the algorithm is capable of searching the best solution among several local minima. After the initialization phase of the algorithm, where the particle randomization is performed, the particles are evaluated and moved iteratively in the search space. In order to eventually converge to the global optimum of the search space, each particle p holds in its memory both *social* and *cognitive* terms, where the former corresponds to the best position found so far by the entire swarm (the global best) and the latter stands for the best position found by the particle p itself (personal best). As will be shortly seen, both the social and cognitive terms contribute in a *stochastic manner* to the particle position at the next iteration round.

An MD PSO algorithm was proposed in [30]. The algorithm allows the particles to make inter-dimensional jumps and visit any dimension, d, within a given range, $d \in [D_{\min}, D_{\max}]$. Thus, in order to provide improved fitness scores, the MD PSO searches for the global best solution among *several* search spaces with different dimensions. The particle navigation among the dimensions is controlled by a separate *dimensional* PSO process, which is interleaved with the regular positional update process. For this, each particle keeps also track of its personal (and the global) best dimension (from which the best fitness value so far has been achieved).

In a MD PSO process, the components of each particle p at iteration round t in a swarm of P particles are presented as,

$d_p(t)$: *dimension* of particle p,
$x_{p,j}^{d_p(t)}(t)$: j^{th} element of the *position* of particle p in dimension $d_p(t)$,
$v_{p,j}^{d_p(t)}(t)$: j^{th} element of the *velocity* of particle p in dimension $d_p(t)$,
$y_{p,j}^{d_p(t)}(t)$: j^{th} element of the *personal best position* of particle p in dimension $d_p(t)$,
$vd_p(t)$: *dimensional velocity* of particle p,
$yd_p(t)$: *personal best dimension* of particle p,
$\hat{y}_j^d(t)$: j^{th} element of the *global best position* of the whole swarm in dimension d, j, $\in [1,d]$
$\hat{y}d(t)$: *global best dimension* of the whole swarm,

where (if not stated otherwise) $j \in \{1,\ldots,d_p\ (t)\}$. The particle fitness values are only evaluated within its current dimension, meaning that the *positional* PSO components in all other dimensions remain the same for the next iteration round $t+1$, that is, $x_p^d(t+1) = x_p^d(t)$, $v_p^d(t+1) = v_p^d(t), y_p^d(t+1) = y_p^d(t), \forall d \in [D_{\min}, D_{\max}] \wedge d \neq d_p(t)$. After computing the fitness score of each particle position with the applied fitness function $\mathcal{F}$, the following update equations are used for the personal best position and

$$y_p^{d_p(t+1)}(t+1) = \begin{cases} y_p^{d_p(t+1)}(t), & \text{if } \mathcal{F}\left[x_p^{d_p(t+1)}(t+1)\right] > \mathcal{F}\left[y_p^{d_p(t+1)}(t)\right] \\ x_p^{d_p(t+1)}(t+1), & \text{else,} \end{cases} \quad (1)$$

dimension of particle p for iteration $t+1$:
and

$$yd_p(t+1) = \begin{cases} yd_p(t), & \text{if } \mathcal{F}\left[x_p^{d_p(t+1)}(t+1)\right] > \mathcal{F}\left[y_p^{yd_p(t)}(t)\right] \\ d_p(t+1), & \text{else.} \end{cases} \tag{2}$$

Furthermore, for each dimension $d \in [D_{\min}, D_{\max}]$, the global best particle position is updated as

$$\hat{y}^d(t+1) = \begin{cases} \hat{y}^d(t), \text{if } \min\limits_p \left(\mathcal{F}\left[y_p^d(t+1)\right]\right) \geq \mathcal{F}\left[\hat{y}^d(t)\right] \\ \underset{y_p^d(t+1)}{\text{argmin}} \left(\mathcal{F}\left[y_p^d(t+1)\right]\right), \text{else,} \end{cases} \tag{3}$$

and, finally, the global best dimension is updated as

$$\hat{y}d(t+1) = \begin{cases} \hat{y}\,d(t), \text{if } \min\limits_p \left(\mathcal{F}\left[y_p^{yd_p(t+1)}(t+1)\right]\right) \geq \mathcal{F}\left[\hat{y}^{\hat{y}d(t)}(t)\right] \\ \underset{yd_p(t+1)}{\text{argmin}} \left(\mathcal{F}\left[y_p^{yd_p(t+1)}(t+1)\right]\right), \text{else,} \end{cases} \tag{4}$$

where, in both (3) and (4), $p \in [1, P]$.

The particle positions within the current dimension $d_p(t)$ are updated after each iteration as shown in (5), where $w(t)$ is a so-called *inertia weight*, c_1 and c_2 are pre-determined constants, $\bar{r}_1$ and $\bar{r}2$ are vectors of uniformly distributed random variables, that is $r_{1,j}, r_{2,j} \sim U(0,1), \forall j \in \left[1, d_p(t)\right]$, and $\mathbb{C}(\bar{\boldsymbol{z}}, [Z_{\min}, Z_{\max}])$ works as a *clamping* operator that limits the elements of vector $\bar{\boldsymbol{z}}$ between the specified values $Z_{\min}$ and $Z_{\max}$. Typical PSO parameters [41] were used in this study, that is, the inertia weight was linearly decreased from 0.9 to 0.4, and c_1 and c_2 were both set to 2. The limiting values for the particle position, velocity, and dimensional velocity, *X*, *V*, and *VD*, respectively, were empirically set into proper values, as will be discussed later in "Experimental results" section. Note that the new particle position, $x_p^{d_p(t)}(t+1)$, remains in the current dimension $d_p(t)$ after the positional update, whereas the dimension may change afterwards in the dimension update process defined in (6), which is performed at the end of each iteration round. The $\lfloor . \rfloor$ operator in (6) stands for a floor function, and r_1 and r_2 are now *scalar* uniformly distributed random variables; otherwise the update is performed similarly to the positional updates. For an interested reader, the pseudo-code and further details of the MD PSO are provided in [30].

$$\begin{aligned} v_{p,j}^{d_p(t)}(t+1) &= w(t) v_{p,j}^{d_p(t)}(t) + c_1 r_{1,j}(t)\left(y_{p,j}^{d_p(t)}(t) - x_{p,j}^{d_p(t)}(t)\right) + c_2 r_{2,j}(t)\left(\hat{y}_j^{d_p(t)}(t) - x_{p,j}^{d_p(t)}(t)\right) \\ \underset{x}{p,j d_p(t)}(t+1) &= x_{p,j}^{d_p(t)}(t) + \mathbb{C}\left(v_{p,j}^{d_p(t)}(t+1), [V_{\min}, V_{\max}]\right) \\ x_{p,j}^{d_p(t)}(t+1) &\leftarrow \mathbb{C}\left(x_{p,j}^{d_p(t)}(t+1), [X_{\min}, X_{\max}]\right) \end{aligned} \tag{5}$$

$$\begin{aligned} vd_p(t+1) &= \lfloor vd_p(t) + c_1 r_1(t)\left(yd_p(t) - d_p(t)\right) + c_2 r_2(t)\left(\hat{y}d(t) - d_p(t)\right)\rfloor \\ d_p(t+1) &= d_p(t) + \mathbb{C}\left(vd_p(t+1), [VD_{\min}, VD_{\max}]\right) \\ d_p(t+1) &\leftarrow \mathbb{C}\left(d_p(t+1), [D_{\min}, D_{\max}]\right) \end{aligned} \tag{6}$$

FGBF algorithm

In some cases, the (MD) PSO algorithm suffers from the so-called *premature convergence* problem, meaning that the global best particle traps into a local minimum in the search space. This is especially true in high-dimensional and multi-modal search spaces, which are often encountered in real-world applications. The problem is mainly caused by the loss of diversity, meaning that all the particles are clamped too close to each other in the search space. A recent method called FGBF [31] has been proposed to tackle this problem by exploiting the potential of individual particle *elements* of each particle position. The idea in the technique is to evaluate a separate fitness score for each particle element, in order to form an *artificial global best* (*aGB*) particle by combining the best elements found from the entire swarm. The formed *aGB* particle is then applied whenever it surpasses in fitness the regular global best particle, $\hat{y}^d(t)$, of the swarm. For MD PSO, this means that a separate *aGB* particle needs to be assigned for each search space dimension, $d \in [D_{\min}, D_{\max}]$. However, in this case the *aGB* particle of a particular dimension can be also formed by combining particle elements from dimensions *other* than the current one. This is demonstrated in Figure 1, where elements from three different particles from separate dimensions,

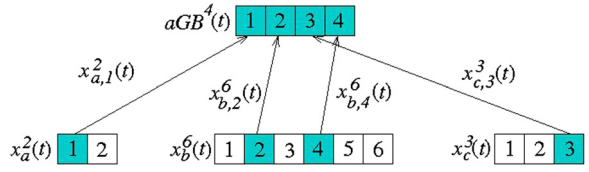

Figure 1 An illustration of the formation of an *aGB* particle dimension of 4. Elements of three different particles, *a*, *b*, and *c*, having the dimensions of 2, 6, and 3, respectively, are brought together and combined in the formation process. Note that several elements can be taken from a single particle.

$x_{p,j}^{d_p(t)}(t)$, are combined to form the *aGB* particle. Such an approach increases the probability of finding an *aGB* particle with a higher fitness score than the existing $\hat{y}^d(t)$ solution for the dimension d at stake. For the sake of clarity, pseudo-code for the FGBF approach within the MD PSO algorithm is shown in Algorithm 1.

Algorithm 1 Pseudo-code of the FGBF algorithm in MD PSO

Let $[j] = \arg\min_{p\in[1,P]}\mathcal{F}\left[x_{p,j}^{d_p(t)}(t)\right]$, then

1. Select the best particle indices for each element, $b[j]$, where $j \in [1,D_{\max}]$, among all the particles, $p\in[1,P]$.
2. For ($d\in[D_{\min}, D_{\max}]$) do:
   a. Assign the best elements into the *aGB* solution: $aGB_j^d(t) = x_{b[j],j}^{d_{b[j]}(t)}, j \in [1,d]$.
   b. If $\left(\mathcal{F}\left[aGB^d(t)\right] < \mathcal{F}\left[\hat{y}^d(t)\right]\right)$, then $\hat{y}^d(t) = aGB^d(t)$.
3. Re-evaluate: $\hat{y}d(t) = \arg\min_d\mathcal{F}\left[\hat{y}^d(t)\right]$.

Heterogeneous particle behaviors

As proposed recently by Engelbrecht [33], variation of the particle *behaviors*, i.e., the velocity update rules, is another efficient way to enhance the convergence ability of the swarm in highly multi-modal problems. In addition to the particle velocity update equations (5) and (6), *four* other update models are introduced in this section. The models are extended to be used with the MD PSO approach so that the corresponding dimensional update rules are changed accordingly. Whenever a particle seems to get stuck into a local optimum, a new behavior model is assigned to it in a random manner. As well, the initial behaviors are chosen randomly for each particle.

Cognitive-only MD PSO model

In the cognitive-only MD PSO model [42], as the name suggests, the *social* terms $\hat{y}^{jd_p(t)}(t)$ and $\hat{y}d(t)$ of the particles (i.e., the latter terms of (5) and (6) with the c_2 coefficients) are *removed* from the velocity update equations. This leads to broader particle exploration as interaction among particles ceases. This, instead, causes every particle to become an *independent* hill-climber in the search space. Thus, whenever the particle position is updated using this model, the (artificial) global best particle position is not considered in the update process.

Social-only MD PSO model

Like the previous model, in the social-only velocity rules [42], the *cognitive* terms $y_{p,j}^{d_p(t)}(t)$ and $yd_p(t)$ of the particles (i.e., the former terms of (5) and (6) with the c_1 coefficients) are removed from the velocity update equations. Such a behavior provides faster particle exploitation, as now the entire swarm becomes a single *stochastic* hill-climber.

Barebones MD PSO

The Barebones PSO was suggested in [43], and in this model the velocity update rule is replaced by

$$v_{p,j}^{d_p(t)}(t+1) \sim \mathcal{N}\left(\frac{y_{p,j}^{d_p(t)}(t) + \hat{y}_j^{d_p(t)}(t)}{2}, \sigma\right), \tag{7}$$

where $\sigma = \left|y_{p,j}^{d_p(t)}(t) - \hat{y}_j^{d_p(t)}(t)\right|$. The position update changes to $x_{p,j}^{d_p(t)}(t+1) = v_{p,j}^{d_p(t)}(t+1)$, so that the velocity ends up being the new position of the particle, sampled from the described Gaussian distribution $\mathcal{N}$. Similarly, for the particle dimensional velocity, the following equation is applied

$$vd_p(t+1) \sim \mathcal{N}\left(\frac{yd_p(t) + \hat{y}d(t)}{2}, \sigma\right), \tag{8}$$

where $\sigma = \left|yd_p(t) - \hat{y}d(t)\right|$. Again, the dimension velocity is considered as the actual new dimension, i.e., $d_p(t+1) = vd_p(t+1)$. Note that the clamping operation is still applied to the obtained positions, so that the set limiting values are not exceeded.

In the positional point of view, the Barebones MD PSO facilitates an *initial exploration*, because at first the personal best positions are far away from the global best solution, causing large deviations to the Gaussian distribution. However, as more iterations are performed, the deviation approaches to zero, causing the behavior to focus on exploitation of the average of the personal best and global best positions.

Modified barebones MD PSO

A modified version of the Barebones, also suggested in [43], is defined as

$$v_{p,j}^{d_p(t)}(t+1) = \begin{cases} y_{p,j}^{d_p(t)}(t), & \text{if } U(0,1) < 0.5 \\ \mathcal{N}\left(\frac{y_{p,j}^{d_p(t)}(t) + \hat{y}_j^{d_p(t)}(t)}{2}, \sigma\right), & \text{else} \end{cases} \tag{9}$$

and for dimensional update, similarly, as,

$$vd_p(t+1) = \begin{cases} yd_p(t), & \text{if } U(0,1) < 0.5 \\ \mathcal{N}\left(\frac{yd_p(t) + \hat{y}d(t)}{2}, \sigma\right), & \text{else.} \end{cases} \tag{10}$$

Such a modification increases the exploration during the initial stages of the search process (compared to

regular Barebones), as now 50% of the time the focus is on the personal best solutions. As the process converges, the behavior will turn to exploitation, when all of the personal best solutions converge towards the global best solution.

As mentioned, each particle obtains its starting behavior randomly among the introduced models (including the regular one), after which the behavior is changed whenever a particle cannot improve its fitness score for the last *ten* consecutive iteration rounds. The whole idea here is that a new behavior model may help the particle to step out from a possible local optimum, and hence eventually provide improvements to the particle fitness score.

EFS and selection

As earlier discussed, the motivation behind proposing the EFS technique is to obtain enhanced audio features, so that audio classification and retrieval performance can be improved. In this section, we will describe the proposed feature selection and synthesis technique in detail. It will be shown that, with a proper encoding scheme (encapsulating several linear/nonlinear operators and the selected original features with their weights), the MD PSO particles can perform an evolutionary search towards finding the optimal synthesis parameters and output vector dimension. For this, a proper fitness measure is to be designed, which maximizes the overall classification (or retrieval) performance. The fitness functions applied in this study for evaluating the particle swarm performance during the synthesis procedure are introduced at the end of the section.

Definition of the main objectives

In an ideal case, a feature synthesis system, also called here as a *feature synthesizer*, receives as its input a specific set of (low-level) audio features, selects the most representative and appropriate subset among them, combines and modifies the features by applying a proper set of transformation operators and feature weights, and finally produces a set of new and descriptive features in an optimal dimension with respect to the fitness function assigned for the problem. Such an ideal feature synthesis operation (for the purpose of clustering) is demonstrated in Figure 2, where two-dimensional features of a 3-class dataset are successfully synthesized into clearly distinct clusters, enabling a much easier classification and/or retrieval task compared to the original feature distribution. Note that, unlike in the figure, the proposed feature synthesis approach allows the output dimension to differ from the original dimension.

Changing the output dimensionality makes the approach somewhat similar to SVM, which attempt to transform the original features into a higher dimension

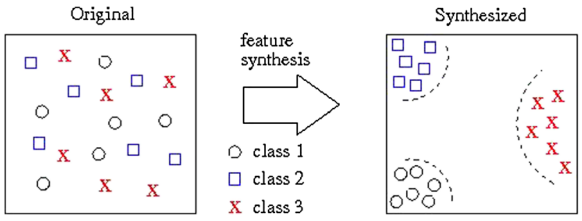

Figure 2 An example of an ideal feature synthesis performed to 2D FVs representing three classes. The synthesized features are much more distinct from each other than the original ones.

to enable linear separation. However, a drawback with SVMs is their high dependency on the applied kernel function and the corresponding internal parameters, which may not fit to the problem at hand properly. This phenomenon is demonstrated in Figure 3, where two successful sample feature synthesizers are presented for a two-class classification problem. The upper case corresponds to an SVM with a *polynomial* kernel in a quadratic form. It is indeed capable of performing a proper transformation from 2D to 3D, enabling thus a linear separation between the two classes. However, in the lower case, a sinusoid with a proper angular frequency, ω, needs to be applied instead for satisfactory class discrimination. Hence, it can be seen that searching for the correct transformation (instead of applying a fixed kernel) is of paramount importance, and this is actually considered as one of the main motivations for designing the feature synthesis scheme proposed in this article.

In light of the above discussion, the main objectives of the proposed EFS technique are to

- perform a proper *feature selection* among the original features,

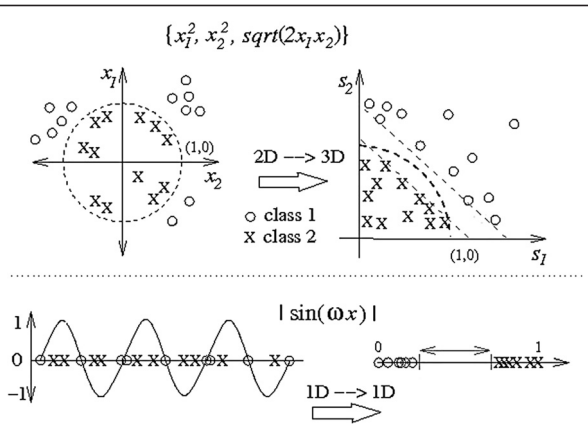

Figure 3 Two examples of feature synthesis (or transform) performed on 2D (upper case) and 1D (lower case) feature spaces using SVM. Depending on the original data distributions, different transformation functions are needed for successful feature separation.

- search for the optimal *operators* and *feature weights* for the synthesis process, and
- search for the optimal output FV dimension among a defined dimension range.

Overview of the proposed feature synthesis system

To meet the aforementioned objectives, an evolutionary search procedure is performed. For each new synthesized feature (meaning a specific *single* feature in the generated output FV), the system, with a specified *synthesis depth* value K,

1. selects $K+1$ original features $f_0,\ldots f_K$, ,
2 scales the selected features with proper weights $w_0,\ldots,w_K$,
3. selects K operators, $\theta_1,\ldots,\theta_K$, to be applied over the selected and scaled features, and
4. bounds the output with a nonlinear operator (here *tangent hyperbolic* is applied).

Now, suppose that $\theta_n(f_a,f_b)$, where $n\in[1,K]$, stands for performing a specific operator θ_n over the features f_a and f_b. Then, a formula for synthesizing a new feature s_j can be defined as

$$s_j=\tanh[\theta_K(\theta_{K-1}(\ldots\theta_2(\theta_1(w_0f_0,w_1f_1),w_2f_2),\ldots),w_Kf_K)], \tag{11}$$

that is, first the operator θ_1 is applied to the *weighted* features f_0 and f_1, after which the operator θ_2 is applied to the *result* of the first operation and the weighted feature f_2, and so on. Finally, the operator θ_K is applied to the result of *all* the previous operations and the weighted feature f_K. With the details provided in "Encoding of the particles" section, the dimension of the synthesized FV, $\bar{s}$, along with the rest of the parameters in (11) are simultaneously optimized within the applied MD PSO search process.

In this article, the term "evolutionary" refers both to the underlying computing technique, the MD PSO, as well as to the nature of the feature synthesis process itself, which can be performed in one or several *runs*. The idea here is that each new run can further synthesize the features generated at the previous run and, hopefully, further increase their discrimination power. A block diagram of the overall synthesis process is illustrated in Figure 4, where R synthesis runs are performed. The total number of runs, R, can either be determined in advance or *adaptively*, in which case the fitness evaluation results are monitored after each run, and the process is stopped after no significant improvement is obtained anymore. Whether there should be more than a single set of features to be synthesized, an *individual* feature synthesizer will be evolved for *each* FS. This is done in order to decrease the computational time needed for the overall processing, as this enables synthesizing the FSs *in parallel* by separate processes.

In a sense, the proposed EFS technique can be seen as a *generalized form* of *artificial neural networks* (ANN). Considering the four system steps listed above, a *single-layer perceptron* (SLP) classifier performs only steps 2 and 4, as neither feature nor operator selection is involved in the process. Instead, SLP does add a bias value to its weighted features, which can be mimicked also in our approach by inserting an additional constant value of *1* at the end of each original input FV (which the synthesizer can then select and scale among the other selected features). However, as no notable performance gain was witnessed by performing such an action, in the end the bias encoding is not considered in the proposed EFS technique. For further comparison, note that in the SLP topology the output layer dimension is fixed, whereas in the EFS the output dimension is (as mentioned) optimized within the set range. Also notice that performing several consecutive EFS runs corresponds to a *multi-layer perceptron* (MLP), or, in fact, any feed-forward ANN type. Similarly to SLP, the MLP does not include feature selection, and it also performs with fixed operator and output dimension. An important difference between the MLP and EFS approaches is that in the EFS technique the fitness of the synthesized FV is evaluated after *each run*, whereas in MLPs only the *final* fitness score in the output layer is considered, as the intermediate network layers are "hidden."

Encoding of the particles

Recalling the PSO definitions introduced in "MD PSO" section, the position of a $d_p(t)$-dimensional particle p at

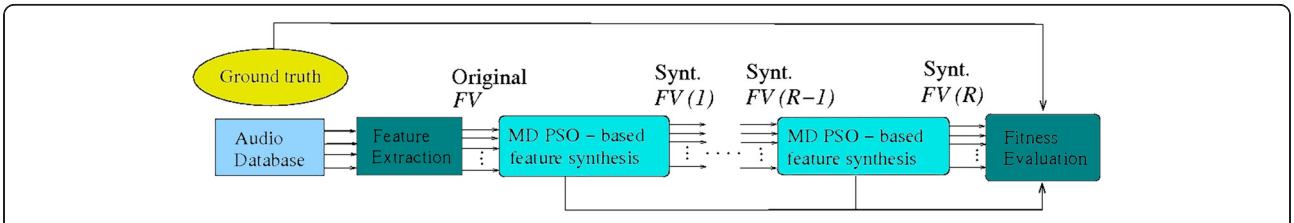

Figure 4 A block diagram of the proposed EFS approach with *R* runs. Different arrows correspond to parallel synthesis processes with different FSs. The fitness evaluation is performed with a specified fitness function.

time t, $x_p^{d_p(t)}(t)$, represents a *potential solution* on how to perform the synthesis operation over the original features, that is, a *potential synthesizer*. The search space dimension, $d_p(t)$, corresponds to the number of features synthesized into the output FV, that is, the output FV dimension. Each particle position encapsulates a complete set of synthesis *parameters*: the indices of the selected features, the feature weights, and the selected feature operators. Accordingly, each *positional element* of a particle p, $x_{p,j}^{d_p(t)}(t)$, where $j \in [1, d_p(t)]$, corresponds to a way of synthesizing the jth feature of the output FV. Thus, referring to the previously introduced four system steps, each such element must contain the following: $K+1$ indices for selecting the original features, $K+1$ feature weights, and K operators in an encoded form to synthesize the corresponding output feature. For this, the positional elements of each particle in the particle swarm are encoded in a *vector form* of length $2K+1$, including $K+1$ "A-type" and K "B-type" components. These define the corresponding synthesis parameters as follows:

$$\begin{aligned} f_n &= \lfloor A_n \rfloor + 1, n \in \{0, K\}, \\ w_n &= A_n - \lfloor A_n \rfloor, n \in \{0, K\}, \\ \theta_n &= \lceil B_n \rceil, n \in \{1, K\}, \end{aligned} \tag{12}$$

where the $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ operators correspond to the *floor* and *ceiling* mathematical integer functions, respectively. The value ranges for the components can be defined based on the *input* FV dimension, F, and the total number of operators available, Θ, as $A_n \in [0, F[$ and $B_n \in]0, \Theta]$. The weight values are limited to $0 \le w_n < 1$.

To give an example of the particle encoding scheme, Figure 5 presents a 6D particle p with the corresponding synthesis process. Note that only the synthesis process of the first element of the output FV at run r, FV(r), is shown in detail, although a similar process is performed for all the output vector elements. For simplicity, the synthesis depth value, K, is set to 3, meaning that only $K+1=4$ features, $f_0,\ldots,f_3$, are selected from input FV($r-1$). Thus, as demonstrated in the figure, each of the particle elements includes $2K+1=7$ encoded synthesis components, $A_0,\ldots,A_3$ and $B_1,\ldots,B_3$. The dimension of the input FV (which may either refer to the original FV consisting of a specific set of low-level features, or to an output FV from a previous EFS run, $r-1$) is $F=8$. As the total number of operators is set to $\Theta=5$, the value ranges for the two component types can be defined as $A_n \in [0, 8[$ and $B_n \in]0, 5]$. In Figure 5, the selected features obtained by the underlying MD PSO process are the 7th, 3rd, 1st, and again the 3rd element of the input FV, while the corresponding operators are selected as '+', 'min', and '$*$' Thus, performing the synthesis process as given in (11), the first element of the output FV is obtained by

$$s_1 = \tanh\left[\min\left(\left(w_0 f_{[7]} + w_1 f_{[3]}\right), w_2 f_{[1]}\right) * w_3 f_{[3]}\right], \tag{13}$$

where $f_{[n]}$ stands for the nth element of the input FV.

Considering again the similarities of the EFS technique and an SLP classifier, it can be noticed that by setting

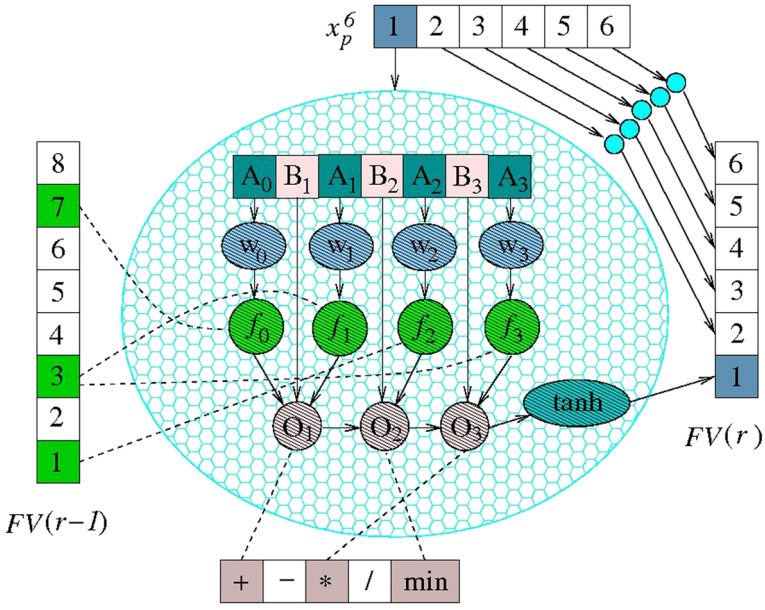

Figure 5 An example of a particle encoding in a 6D feature space with a synthesis depth set to $K=3$, an input FV dimension of $F=8$, and the number of operators defined as $\Theta=5$.

$K+1=F$, discarding the feature selection as $f_n = f_{[n]}$, and setting each operator θ_n to '+', the two approaches become identical. Similarly, by performing several runs with such synthesis parameters corresponds to an MLP approach with a one-to-one analogy between the number of hidden layers and the number of runs performed. In this sense, it can be stated that a regular feed-forward ANN is a *special case* of the proposed EFS approach.

The fitness measures

Proper designing of the applied fitness measure plays an essential role in the feature synthesis process. The design is dependent on the intended use of the features, as the measured fitness value should highly relate to the objective of the synthesis process. Traditionally in content-based classification and retrieval scheme, specific *similarity measures*, such as Euclidean distance, are applied to measure the distances between the FVs of a classified (or queried) item and each item belonging to the database. In a retrieval case, the performance can be evaluated using the *average precision* (*AP*) metric, or the so-called *average normalized modified retrieval rank* (*ANMMR*), which is defined in the MPEG-7 standard [44].

As the main goal of the research is to improve audio retrieval performance by the means of feature synthesis, an intuitive approach for constructing a proper fitness function $\mathcal{F}[\cdot]$ for the task would involve computing either the average retrieval precision (*AP*) or the *ANMMR* values. However, neither option would be computationally feasible for large databases, as they both require conducting a separate *batch query* (i.e., selecting each item in the database as a query item one by one, performing a separate query for each of them, and finally taking the mean of the obtained retrieval results) for every fitness evaluation during the synthesis process. Therefore, in this article, we concentrate on obtaining a *maximal discrimination* between the features of different database classes, which should in turn result in improved retrieval performance. To achieve this we propose two alternative ways to form the fitness function, described in the following sections.

Discrimination measure

First, a measure for evaluating the discrimination capability provided by the synthesized features is proposed. The measure is based on two widely used criteria in clustering:

- *Compactness*: the database items of one cluster should be similar and close to each other in the feature space, and
- *Separation*: different clusters and their centroids should be distinct and well separated from each other.

Suppose the different labels of an L-class database are denoted as $l_0, \ldots, l_{L-1}$, and the corresponding class centroids as $\mu_0, \ldots, \mu_{L-1}$. Then, the following *discrimination measure* (*DM*) can be defined over a set of synthesized FVs, $S = \{s\}$,

$$\begin{aligned} &\mathrm{DM}[S] = \mathrm{FP}(S) + \delta_{\mathrm{mean}}(l_n)/\delta_{\mathrm{min}}(l_n, l_m), \\ &\text{where} \\ &\delta_{\mathrm{mean}}(l_n) = \frac{1}{L}\sum_{n=0}^{L-1}\frac{\sum_{\forall s \in l_n}\|\mu_n - s\|}{|l_n|}, \\ &\delta_{\mathrm{min}}(l_n, l_m) = \min_{n \neq m}(\|\mu_n - \mu_m\|). \end{aligned} \tag{14}$$

The terms of (14) are defined as follows: FP(S) stands for the number of *false positive* FVs occurring among the synthesized FVs S, meaning that those FVs are actually located in closer proximity to some other class centroid than their own, $\delta_{\mathrm{mean}}(l_n)$ is the average intra-class distance, and $\delta_{\mathrm{min}}(l_n, l_m)$ corresponds to the minimum centroid distance among all the classes. Thus, the discrimination measure, DM[S], strives for minimizing the intra-cluster distance, while maximizing the shortest inter-cluster distance. Ideally, each synthesized feature is in the closest proximity of its own class centroid, thus leading to a high discrimination among classes as illustrated in Figure 2. However, minimizing the DM does not always lead to improved retrieval results. This is due to the fact that query items located at the outskirt of their own classes may be actually situated in closer proximity to some other FV located on the outskirt of its corresponding (wrong) class. Thus, in order to improve not only the feature discrimination in the feature space, but also the audio retrieval performance, next we propose applying a similar methodology that is utilized in feed-forward ANNs.

Target vector assignment

In the second approach, the idea is to assign a binary synthesizer *target vector* for each class, and let the underlying optimization algorithm to search for the proper synthesis parameters producing the desired output. The actual fitness value is then obtained by comparing the obtained and desired output vectors in a *mean square error* (MSE) sense. However, as the output dimension of the EFS is not fixed in advance, a separate target vector is generated for each dimension $d \in [D_{\mathrm{min}}, D_{\mathrm{max}}]$, resulting to a complete target *matrix*. For producing the matrix, a so-called *error correcting output code* [45] analogy is applied, which suggests two criteria for generating proper binary target matrices:

- *Row separation*: The target vectors should be well separated from each other in the terms of Hamming distance.

- *Column separation*: Also the columns of the target matrix should be well separated from each other in the terms of Hamming distance.

Large row separation allows the synthesized vectors to differ somewhat from the actual target vectors without losing the discrimination between different classes. The reasoning for large column separation follows from the fact that each column in the target matrix can be seen as an individual *binary* classification task (between the classes having a value 1 and the classes having a value −1 in a specific column). Because of the varying similarity between two arbitrary audio classes, some of such binary classifications are likely to become much easier than others. Hence, as the same target vectors are nonetheless applied to any given input classes, it is also beneficial to keep the binary classification tasks as different as possible by maximizing the column separation.

To generate a binary matrix with maximal row and column separation, the following matrix generation procedure is used:

1. Compute the minimum number of bits, $b_{\min}$, needed to represent the total number of classes, L, in the database.
2. Form an empty matrix with L rows.
3. For each row, assign a binary representation of the row number $n \in [0, L-1]$ as the first/next $b_{\min}$ target vector values.
4. Move the first row of the matrix to the bottom and shift the other rows up by one.
5. Repeat the steps 3 and 4 until $D_{\max}$ target vector values have been assigned.
6. Replace the first L values of each target vector with a 1-of-L coded [46] section.

The procedure generates new columns until the matrix is rotated back to its original order, resulting into a high column separation. Simultaneously, the row separation is greatly increased compared to the regular 1-of-L coding section. However, in practice it was observed that for distinct classes it is often easiest to find a synthesizer that discriminates a certain single class from the others, and, therefore, sparing the 1-of-L coding section at the beginning of the matrix generally improves the synthesis results. For the vector dimensions $d < D_{\max}$, only the first d elements of the target vectors are considered. This yields identical target vector elements between the vectors of different length, allowing the FGBF algorithm to combine particle position elements from different dimensions (refer to FGBF algorithm).

For clarification of the procedure, a target matrix for a 4-class database is demonstrated in Figure 6, where the maximum dimension is set to $D_{\max} = 10$, and the minimum number of bits needed to represent the $L = 4$ classes is $b_{\min} = 2$. The empty entries of the matrix correspond to values −1, and $T[l_n]$ stands for a target vector for class l_n. As illustrated in this example, the generated matrix follows the provided algorithm with its 1-of-L coding section for the first L elements, followed by the elements created by the shifted rows of 2-bit row number representations.

By applying the definitions given above, the fitness value for the jth elements of all the synthesized FVs, S_j, can be computed as

$$\mathcal{F}_j[S_j] = \sum_{n=0}^{L-1} \sum_{\forall s \in l_n} \left(T[l_n]_j - s_j \right)^2, \tag{15}$$

where $T[l_n]_j$ denotes the jth element of the target vector of class l_n, and s_j is the jth element of a single synthesized output vector belonging to class l_n. The overall fitness score $\mathcal{F}[S]$ can then be formed by adding the *fractional* fitness scores $\mathcal{F}_j[S_j]$ together and applying normalization with respect to the number of dimensions. However, due to an observation that the first L vector elements, having the 1-of-L encoding, are usually the easiest ones to synthesize (and thus mainly favored in the dimension search by the MD PSO algorithm), the first L vector elements are handled separately in the summation process. Thus, for the dimensions $d > L$, the normalizing divisor is *strengthened* by an additional power parameter $\alpha > 1$, which is to moderately increase the probability of converging into higher output vector dimensions whenever found beneficial. As a result, the overall fitness function $\mathcal{F}[S]$ can be formulated as

$$\mathcal{F}[S] = \frac{1}{L} \sum_{j=1}^{L} \sum_{n=0}^{L-1} \sum_{\forall s \in l_n} \left(T[l_n]_j - s_j \right)^2 + \frac{1}{(d-L)^{\alpha}} \sum_{j=L+1}^{d} \sum_{n=0}^{L-1} \sum_{\forall s \in l_n} \left(T[l_n]_j - s_j \right)^2, \tag{16}$$

in which it is assumed that $D_{\min} > L$.

$T[l_0]$	1				1		1	1		
$T[l_1]$		1			1	1				1
$T[l_2]$			1					1	1	
$T[l_3]$				1		1	1		1	1

Figure 6 An illustration of a target matrix assigned for $L = 4$ classes.

Experimental results

The EFS technique proposed in this article was tested with several audio classification and retrieval experiments using two separate audio databases. The first database consists of six distinct classes, while the second one was created by adding another six classes to the first database to form a more challenging 12-class database. The contents of the two databases are shown in Table 3, where the class numbers ranging from 1 to 6 belong to the *basic* database of 6 classes, and the *extended* database with 12 classes includes also the class numbers ranging from 7 to 12. The audio class samples are collected from a few different data sources; the speech classes (class nos. 1 and 7) are derived from the TIMIT[a] database, the music classes (class nos. 5, 6, and 12) are from the RWC Music Database[b] and another music collection at Tampere University of Technology (TUT), the "general" audio sounds (class nos. 4, 9, 10, and 11) were purchased from the *StockMusic.com* webpage,[c] and, finally, the singing and whistling samples (class nos. 2, 3, and 8) are self-recorded and produced at TUT. The reasoning for two separate databases is to evaluate the *scalability* of the proposed feature synthesis system to more complex and difficult classification and retrieval tasks (i.e., to those cases where good features are truly needed).

In the experiments shown in this section, unless stated otherwise, the following parameters and settings were used for the EFS: the depth of the synthesis was set to $K = 7$, meaning that 7 operators and $K + 1 = 8$ features were chosen for the synthesis process of each output vector element, and the total number of operators, listed in Table 4 for features f_a and f_b, was set to $\Theta = 18$. Similarly, the parameters for the MD PSO algorithm were set as follows: the swarm size was set to $P = 600$ particles, 1,500 iterations were used, the dimension ranges for the basic and extended databases were set as $[D_{\min}, D_{\max}] = [7, 45]$ and $[D_{\min}, D_{\max}] = [13, 45]$, respectively, and the dimensional velocity range was experimentally determined as $[VD_{\min}, VD_{\max}] = [-6, 6]$. Finally, the parameter $\alpha > 1$ used in (16) was determined separately for the key-frame and segment features as $\alpha = 1.15$ and $\alpha = 1.3$, respectively, as these values were found out to yield the best trade-off between the computational cost caused by higher dimensionality and the possible lack of performance caused by reduction of the synthesized FV dimension.

In this study, separate "training sets" were considered for both databases. The training sets were formed by random selection, such that 45% of the audio clips of each class from both databases were included to them. Analogous to supervised machine learning, extracted features of these training sets were then used in searching the most suitable synthesis parameters for the corresponding databases, after which the found parameters were applied in synthesizing the features of the whole data. Thus, after the parameter search process, the resulting EFS system may synthesize new features for the rest of the (unseen) data without further optimization processes. The final obtained synthesized features were tested with multiple evaluations, demonstrating their enhanced discrimination capabilities over the original FSs.

Performance evaluations on feature discrimination

We will first concentrate on evaluating the feature DM of the original and synthesized FSs. The DM was given in (14), and the obtained results for the original and synthesized segment- and key-frame-level audio features of low-level audio features are shown in Table 5. Recall that the higher the DM value, the more the features are mixed up with each other in the feature space (indicating less discrimination). As

Table 3 The contents of the two audio databases

		Class number	Audio class name	Number of samples
12-class audio database (extended)	6-class audio database (basic)	1	Female speech	111
		2	Male singing	63
		3	Whistling	94
		4	Breaking glass	83
		5	Classical music	116
		6	Electrical/techno music	107
	The 6 added classes	7	Male speech	103
		8	Female singing	119
		9	Bird singing	91
		10	Dog barking	49
		11	Fire sounds	96
		12	Rock/metal music	72
			Total	1104

Table 4 The list of operators used in the feature synthesis process for features f_a and f_b

θ_n	Formula	θ_n	Formula
0	$-f_a$	9	$f_a * f_b$
1	$-f_b$	10	$10\ (f_a * f_b)$
2	$\max\ (f_a, f_b)$	11	f_a/f_b
3	$\min\ (f_a, f_b)$	12	$\sin\ (100\pi(f_a + f_b))$
4	$f_a * f_a$	13	$\cos(100\pi(f_a + f_b))$
5	$f_b * f_b$	14	$\tan(100\pi(f_a * f_b))$
6	$f_a + f_b$	15	$\tan(100\pi(f_a + f_b))$
7	$10\ (f_a + f_b)$	16	$0.5 * \exp(-(f_a - f_b) * (f_a - f_b))$
8	$f_a - f_b$	17	$0.5 * \exp(+(f_a + f_b) * (f_a + f_b))$

expected, it can be seen that discriminating the features of the extended 12-class database is significantly more challenging than those with the basic database. However, after performing the feature synthesis procedure with the found synthesis parameters, substantial DM improvements were obtained for all the FSs. Note that, due to the stochastic nature of the underlying optimization approach, *ten* separate EFS evaluations were performed for all the FSs, and the obtained mean (μ) and standard deviation (σ) values are also reported. The rest of the synthesis results presented in this section conform to the same protocol.

As the DM as such is not an intuitive measure of fitness for the FSs, we will concentrate on demonstrating the audio retrieval performances obtained with different FSs. Recall, however, that in general the usage of new synthesized features is not by any means restricted to merely retrieval purposes. Instead, as long as a proper fitness function can be designed, the EFS technique can be applied to basically any task requiring feature improvement. To show an example, a rather obvious application is demonstrated, in which a *K-means* classifier is applied. The algorithm was first run with the formed training sets to compute the class centroids, after which the unseen (55%) *test* data samples were classified according to their closest class centroid. The *classification error* (CE) results obtained with the synthesized features for the both databases were compared to the errors associated with the original FSs. The results, shown in Table 6, indicate clearly improved classification performance obtained by the new synthesized features.

Performance evaluations on audio retrievals

For evaluating the audio retrieval performance, the metrics, ANMRR and AP, introduced in "The fitness measures" section were applied. In our approach, we first compute (using Euclidean distance) the distances between the first FV of a query item and all the FVs of a particular database item from which we take the *minimum* distance (normalized by the vector length) and store it. Second, we take the next FV of the query item and continue similarly with all the query item FVs. Finally, before moving to the next database item, we take the *sum* of all the obtained minimum distances to obtain a single distance value between the queried item and the database item (which is used to rank that query item). In order to provide some baseline results, the ANMRR and AP values obtained by using a batch query and the originally extracted audio features are shown in Table 7. As expected, for both the segment- and the key-frame-based FSs, the retrieval performance deteriorates considerably as the database size increases from the basic 6-class database to the extended 12-class database. Like in the results shown in Tables 5 and 6, the segment MFCC features seem to provide the best performance among the original FSs.

Table 5 The DM statistics of ten separate EFS evaluations for the original and synthesized features

FS	Basic database (6 classes)		Extended database (12 classes)	
	Original DM	Synthesized DM (μ ± σ)	Original DM	Synthesized DM (μ ± σ)
Segment				
STAT	500.5	43.6 ± 6.7	1793	406.3 ± 47.6
MFCC	333.4	61.5 ± 8.5	1141	361.0 ± 22.4
Δ-MFCC	515.7	130.5 ± 7.9	1865	620.1 ± 27.1
ΔΔ-MFCC	622.6	140.9 ± 11.5	2072	532.7 ± 23.6
LPC	1114	270.3 ± 8.9	4520	1204 ± 29.1
LPCC	1638	310.3 ± 12.8	10830	1395 ± 26.5
S_AUDIO	342.6	56.0 ± 4.1	1365	382.8 ± 15.2
Key-frame				
MFCC + deltas	2627	820.6 ± 53.6	7113	3093 ± 138
LPC + LPCC	6951	2143 ± 39.4	23 900	6378 ± 157
K_AUDIO	3150	782.7 ± 50.5	10 140	2774 ± 154

Table 6 CE of a K-means classifier over the original and synthesized features

FS	Basic database (6 classes)		Extended database (12 classes)	
	Original CE (%)	Synthesized CE (%)	Original CE (%)	Synthesized CE (%)
Segment				
STAT	33.3	15.2 ± 1.6	54.8	37.4 ± 2.4
MFCC	18.2	13.5 ± 1.7	36.0	34.4 ± 2.9
Δ-MFCC	30.7	24.2 ± 1.4	48.3	39.9 ± 1.5
ΔΔ-MFCC	37.8	29.1 ± 2.1	56.7	39.4 ± 2.3
LPC	47.5	41.8 ± 1.3	70.8	68.4 ± 2.0
LPCC	57.6	45.6 ± 1.4	78.9	74.5 ± 1.6
S_AUDIO	22.1	18.0 ± 0.9	42.8	34.3 ± 2.0
Key-frame				
MFCC + deltas	37.2	29.2 ± 2.2	50.3	50.0 ± 2.0
LPC + LPCC	75.6	64.4 ± 1.2	86.8	84.5 ± 1.2
K_AUDIO	53.8	36.0 ± 4.6	69.1	46.2 ± 2.7

Next, a standard SLP classifier was trained with the MD PSO approach (by the methods described in [30]), in order to compare the proposed EFS technique with neural network-based approaches. Similar to EFS, an SLP can be treated as a feature synthesizer, in which case the original audio features are propagated through the network and the output layer dimensionality is set equal to the total number of classes, L. We call the resulting output vector a *class vector*, as it indicates the class designated for a particular input FV. The corresponding ANMRR and AP values obtained with these vectors are shown in Table 8. Compared to the values obtained with the original features, a clear improvement can be observed with nearly all the FSs, excluding some of the segment features of the extended database. As mentioned in "Low-level audio features" section, this is most probably due to the fact that, because of the nature of the segment-based features, the total number of FVs per an audio clip is considerably less for segment features than for key-frame features. Hence, because the FV dimensionality is highly decreased during the SLP synthesis process (to L), the extended database is extremely complicated for an SLP classifier to learn with such a limited amount of data. In contrast, the basic database is well learned by the SLP, as an AP of nearly 90% is achieved with the segment-based MFCC features.

We will now demonstrate the significance of the additional properties associated with the EFS technique

Table 7 The retrieval performances obtained with the original features for both audio databases

FS	Basic database (6 classes)		Extended database (12 classes)	
	ANMRR	AP (%)	ANMRR	AP (%)
Segment				
STAT	0.358	61.0	0.514	45.9
MFCC	0.228	73.8	0.367	60.2
Δ-MFCC	0.351	62.2	0.507	46.6
ΔΔ-MFCC	0.379	59.5	0.531	44.4
LPC	0.549	43.1	0.686	29.8
LPCC	0.569	41.1	0.717	26.9
S_AUDIO	0.263	71.0	0.450	52.3
Key-frame				
MFCC + deltas	0.460	51.4	0.568	41.0
LPC + LPCC	0.637	34.8	0.787	20.4
K_AUDIO	0.375	59.5	0.524	44.9

Table 8 The retrieval performances obtained with the class vectors of an MD PSO-trained SLP classifier for both audio databases

FS	Basic database (6 classes)		Extended database (12 classes)	
	ANMRR	AP (%)	ANMRR	AP (%)
Segment				
STAT	0.238	75.0	0.626	36.4
MFCC	0.101	89.2	0.518	47.3
Δ-MFCC	0.339	64.7	0.581	40.0
ΔΔ-MFCC	0.290	70.1	0.585	40.1
LPC	0.491	49.7	0.673	31.3
LPCC	0.564	42.3	0.714	27.4
S_AUDIO	0.229	75.7	0.578	41.2
Key-frame				
MFCC + deltas	0.279	70.2	0.414	56.5
LPC + LPCC	0.669	32.1	0.794	19.7
K_AUDIO	0.308	67.7	0.504	47.1

germane to SLPs. In the first demonstration, the feature selection property is enabled and applied to the input FVs, so that only $K+1=8$ original features are included into the actual synthesis process. The output FV dimensionality is fixed to L, and only the '+' and '−' operators are included in the synthesis process in order to mimic the behavior of typical neural networks (the subtraction operation is also needed to compensate with SLP weights, which are limited to [−1,1]). Such arrangements make the EFS similar to an SLP classifier (with the exception of additional feature selectivity). The retrieval performance obtained with these settings is shown in Table 9, from where it can be seen that, excluding the MFCC features, the results are fairly comparable to those obtained with the SLP classifier. This is a result worth noting as only *eight* features were selected among the original input FVs. Such a reduction in the number of original features (without significant performance loss) suggests that at least some of them may not be essential to attain optimal discrimination capability.

In the next phase, the fixed output dimensionality of the synthesis method was changed so that the optimal dimensionality found using the underlying MD PSO algorithm was used in the synthesis process. All the operators shown in Table 4 were included to the process to provide the synthesizer more possibilities for modifying the features. After the changes, a significant improvement in the retrieval performance was obtained, as shown in Table 10. Hence, optimizing the output FV dimension (and applying several operators in the synthesis process) significantly enhances the retrieval performance. However, it should be noted that retrieval performance obtained with the original segment-based MFCC features of the extended database could not be improved either by the SLP classifiers or by the EFS experiments made so far. We believe this indicates that the MFCC features need to be treated as *a whole* and that selecting only few of them may decrease the overall retrieval performance. However, in this instance the performance drop is limited and clear performance improvements relative to key-frame-based MFCC features are still achieved. Also key-frame LPC + LPCC features show only minor improvements in AP values, which implies that certain audio features are not as *synthesizable* as others or that different types of arithmetic or logic operators may be required to achieve significant improvements in discrimination performance.

Finally, several *consecutive* EFS runs were performed to see whether the obtained synthesis results can be further improved (see Figure 4 in "Overview of the proposed feature synthesis system" section). Retrieval performance associated with certain synthesized FSs improved over several runs (<25), whereas some other features could not be enhanced notably. More generally, the segment-based features were more suitable for consecutive EFS runs, as retrieval performance of the key-frame features remained rather stable during the runs. A graphical presentation of the experiments is shown in Figure 7, where the evolution of both the AP results and the synthesized FV dimensions over 25 synthesis runs are shown for several FSs and for both databases ($L=6$/$L=12$). For simplicity, those FSs unable to demonstrate

Table 9 The retrieval performance statistics obtained using the EFS technique with fixed output dimension and only two operators

FS	Basic database (6 classes)		Extended database (12 classes)	
	ANMRR ($\mu \pm \sigma$)	AP (%) ($\mu \pm \sigma$)	ANMRR ($\mu \pm \sigma$)	AP (%) ($\mu \pm \sigma$)
Segment				
STAT	0.278 ± 0.016	70.4 ± 1.7	0.534 ± 0.024	44.1 ± 2.4
MFCC	0.280 ± 0.022	69.6 ± 2.1	0.528 ± 0.023	44.9 ± 2.2
Δ-MFCC	0.375 ± 0.016	60.6 ± 1.6	0.574 ± 0.011	40.6 ± 1.0
ΔΔ-MFCC	0.393 ± 0.013	59.0 ± 1.3	0.598 ± 0.020	38.3 ± 2.0
LPC	0.549 ± 0.006	43.8 ± 0.6	0.729 ± 0.013	25.7 ± 1.2
LPCC	0.589 ± 0.015	39.6 ± 1.5	0.740 ± 0.006	24.8 ± 0.5
S_AUDIO	0.236 ± 0.015	74.7 ± 1.6	0.480 ± 0.013	49.9 ± 1.2
Key-frame				
MFCC + deltas	0.463 ± 0.024	51.4 ± 2.3	0.629 ± 0.030	35.2 ± 2.8
LPC + LPCC	0.709 ± 0.007	28.4 ± 0.7	0.813 ± 0.003	18.0 ± 0.3
K_AUDIO	0.346 ± 0.016	63.0 ± 1.5	0.532 ± 0.012	44.4 ± 1.2

Table 10 The retrieval performance statistics obtained using the EFS technique with dynamic output dimension and 18 operators

FS	Basic database (6 classes)		Extended database (12 classes)	
	ANMRR ($\mu \pm \sigma$)	AP (%) ($\mu \pm \sigma$)	ANMRR ($\mu \pm \sigma$)	AP (%) ($\mu \pm \sigma$)
Segment				
STAT	0.167 ± 0.013	81.7 ± 1.5	0.425 ± 0.012	55.0 ± 1.2
MFCC	0.221 ± 0.021	75.5 ± 2.2	0.398 ± 0.012	57.5 ± 1.2
Δ-MFCC	0.332 ± 0.015	64.7 ± 1.5	0.511 ± 0.096	43.9 ± 3.1
ΔΔ-MFCC	0.313 ± 0.016	66.7 ± 1.6	0.526 ± 0.033	45.0 ± 3.2
LPC	0.524 ± 0.016	46.1 ± 1.5	0.675 ± 0.008	31.1 ± 0.8
LPCC	0.556 ± 0.010	42.8 ± 1.0	0.720 ± 0.017	26.7 ± 1.6
S_AUDIO	0.171 ± 0.011	81.5 ± 1.2	0.442 ± 0.029	53.4 ± 2.9
Key-frame				
MFCC + deltas	0.371 ± 0.012	60.3 ± 1.2	0.470 ± 0.012	50.5 ± 1.1
LPC + LPCC	0.634 ± 0.013	35.3 ± 1.2	0.775 ± 0.011	21.6 ± 1.0
K_AUDIO	0.289 ± 0.029	68.7 ± 2.8	0.441 ± 0.009	53.2 ± 0.9

much improvement during the process are not shown in the graphs, whereas the best obtained AP values of all the FSs are shown in Table 11. The most significant observation concerns the behavior of the extended database segment features (upper right plots in Figure 7), where major improvements can be seen with all the shown FSs. Moreover, the dimensionality of the "S_AUDIO" FV could be reduced to only 13. Also note that now the AP performance of the synthesized segment-based MFCC features surpasses the original features after the second EFS run. However, in the case of key-frame features (the lower plots), the improvement is more moderate, stating that these features have not as much additional potential to be found by repeating the synthesis process, or at least such potential could not be found in the experiments using the applied MD PSO and EFS parameters.

Comparative evaluations and discussion

In order to provide some additional comparative results, an MLP classifier was trained using the MD PSO algorithm as described in [30]. In this approach, a specific *architecture space* (AS) with lower and upper limits for the number of neurons for each network layer is specified, from which the applied optimization algorithm searches for the optimal network structure for synthesizing new features. Two different ASs were experimented, specified by the limiting vectors

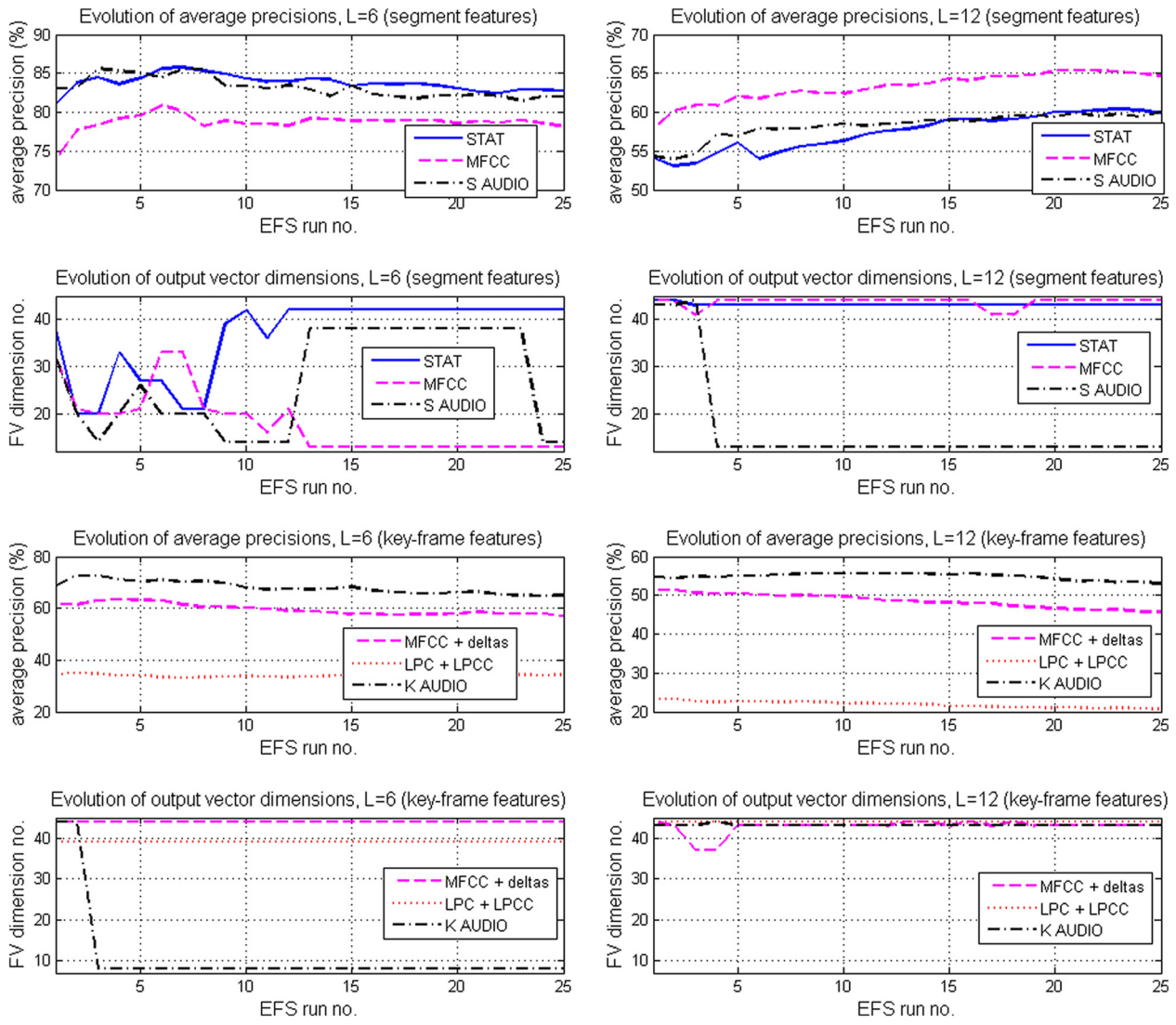

Figure 7 The consecutive EFS runs performed for basic (left) and extended (right) databases. It can be seen that the AP results generally improve during the couple of first runs. The segment features seem to improve over several runs, whereas the performance level obtained by key-frame features saturates more quickly. Note that satisfactory solutions can also be found with low FV dimensions.

Table 11 The best retrieval performances obtained by 25 consecutive EFS runs

FS	Basic database (6 classes)		Extended database (12 classes)	
	ANMRR (run no.)	AP (%) (run no.)	ANMRR (run no.)	AP (%) (run no.)
Segment				
STAT	0.134 (run 7)	85.8 (run 7)	0.380 (run 23)	60.4 (run 23)
MFCC	0.181 (run 6)	80.9 (run 6)	0.333 (run 22)	65.3 (run 22)
Δ-MFCC	0.235 (run 16)	76.0 (run 16)	0.514 (run 7)	47.0 (run 7)
ΔΔ-MFCC	0.258 (run 10)	73.2 (run 10)	0.467 (run 10)	51.4 (run 10)
LPC	0.446 (run 22)	54.3 (run 22)	0.673 (run 2)	31.4 (run 2)
LPCC	0.521 (run 24)	47.3 (run 24)	0.710 (run 1)	27.7 (run 1)
S_AUDIO	0.135 (run 3)	85.6 (run 3)	0.386 (run 25)	60.0 (run 25)
Key-frame				
MFCC + deltas	0.346 (run 4)	63.3 (run 4)	0.462 (run 2)	51.4 (run 2)
LPC + LPCC	0.638 (run 2)	35.0 (run 2)	0.757 (run 2)	23.3 (run 2)
K_AUDIO	0.251 (run 2)	72.5 (run 2)	0.426 (run 10)	55.6 (run 16)

Table 12 The retrieval performances obtained with the class vectors of an MLP classifier

FS	Basic database (6 classes)		Extended database (12 classes)	
	ANMRR	AP (%)	ANMRR	AP (%)
Segment				
STAT	0.334 (1)	66.0 (1)	0.620 (2)	37.1 (2)
MFCC	0.089 (1)	90.4 (1)	0.532 (1)	45.7 (1)
Δ-MFCC	0.340 (2)	65.3 (2)	0.680 (1)	30.9 (1)
ΔΔ-MFCC	0.320 (1)	66.8 (1)	0.634 (2)	35.3 (2)
LPC	0.489 (1)	50.2 (1)	0.704 (2)	28.7 (2)
LPCC	0.562 (1)	42.6 (1)	0.746 (1)	24.4 (1)
S_AUDIO	0.255 (1)	73.6 (1)	0.561 (2)	43.2 (2)
Key-frame				
MFCC + deltas	0.334 (1)	64.8 (1)	0.555 (1)	42.7 (1)
LPC + LPCC	0.669 (1)	32.1 (1)	0.805 (1)	18.8 (1)
K_AUDIO	0.391 (2)	59.4 (2)	0.604 (1)	38.0 (1)

$AS1_{\min} = \{F, 13, L\}, AS1_{\max}\{F, 45, L\}$ and $AS2_{\min} = \{F, 8, 4, L\}, AS2_{\max} = \{F, 16, 8, L\}$, where the fixed input and output layers F and L correspond to the number of input features and database classes, respectively. These settings correspond to performing two (AS1) and three (AS2) consecutive EFS runs. For this reason, the number of MD PSO iterations were set to $2*1500 = 3000$ and $3*1500 = 4500$, respectively, in order to have a fair comparison. Both ASs were separately searched with the MD PSO algorithm and the best network configuration was used to generate the corresponding MLP class vectors from the original features. The obtained retrieval performance measures are shown in Table 12, where also the AS number (1 or 2) yielding the better result is shown in parenthesis. The MLP classifier performs notably well with the segment-based MFCC features for the basic database, whereas its performance falls below the EFS results with the extended database. This result suggests that regular neural networks can succeed as synthesizers with rather small and simple databases. Also, as no feature selection is performed, the MLP can utilize the original MFCCs as a whole, which may ease the synthesis process. On the other hand, when it comes to the key-frame-based MFCC feature performance, the difference is not as significant. This suggests that the content and nature of the original FV indeed has an effect on the synthesis process output both with ANNs and EFS.

Strictly speaking, performing several EFS runs in a row is analogous to concatenating several SLP classifiers such that the output class vector of a previous SLP is considered as an input vector for the next one. To see whether such an arrangement makes a difference in retrieval performance, Figure 8 shows the evolution of AP values of ten concatenated SLP classifiers trained with the MD PSO. The obtained results are shown also numerically for all FSs in Table 13. Compared to single SLP evaluations, no significant improvements were achieved by concatenating the SLP classifiers. Rather, retrieval performance begins to decrease either immediately or after two or three SLP classifiers. This leads to a conclusion that, at least with the databases and parameters used in our study, SLP (as a feature synthesizer) achieves its best performance almost right away, whereas the proposed EFS framework can improve its performance over several additional synthesis runs. Moreover, in most cases the EFS results are clearly better than those obtained with MLPs, which supports the EFS approach of performing the fitness evaluation after every run.

Finally, the evaluation shown in Figure 7 was repeated for segment-based features so that every EFS run was repeated *three times,* with only the best synthesis solution (i.e., the one maximizing the retrieval performance in the training dataset) being used. This was done in order to demonstrate the full potential of the EFS technique; in this way the occasional sub-optimal solutions (caused by the stochastic nature of the search process) can be reduced efficiently. Note that unless grid computing with parallel processes can be used, the processing time would be three times longer. Hence, such a demonstration is more about providing an idea of EFS's potential than a practical application. Nevertheless, Figure 9 shows additional improvements in AP scores, especially in the case of the extended 12-class database where performance increases notably. With the basic database, the optimal performance level (in the sense of AP values) is reached after 5–7 runs. Note that improved retrieval

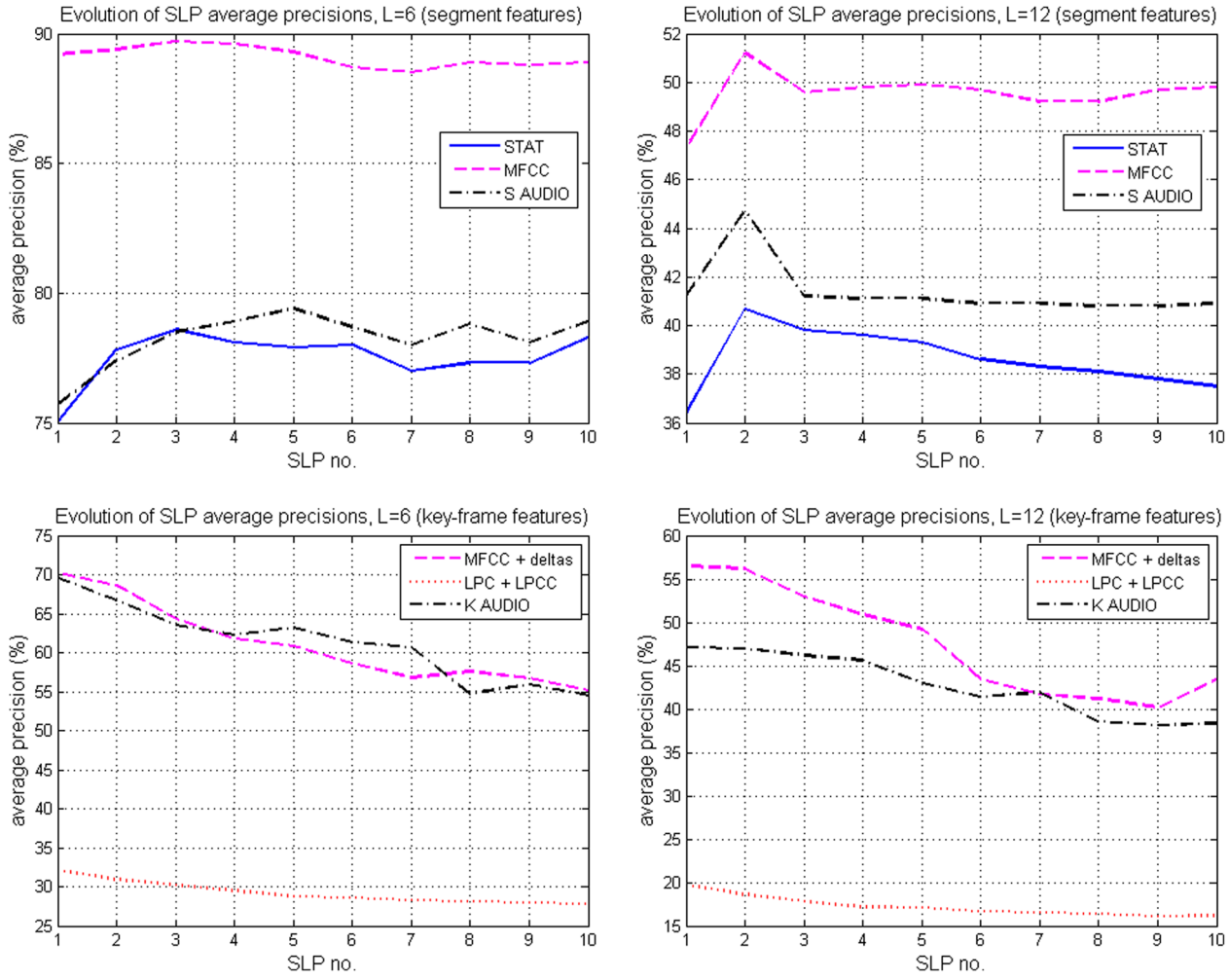

Figure 8 The AP performances obtained by consecutive SLP classifiers. In general, the results cannot be further improved by concatenating the SLPs, except with the segment features for the first few classifiers.

performance was also obtained using lower FV dimensions than the original ones, which is an attractive property considering the computational requirements for processing the vectors. For example, the dimensionality of the synthesized vectors can significantly be decreased for STAT and S_AUDIO features. However, the best FV dimensions found may vary depending on the FS. Specifically, it was observed that synthesizing the 26-dimension MFCC features of the extended database into a lower dimensionality (and yet obtain improved retrieval performance) is a challenging task. Nonetheless, allowing a dynamic output FV dimensionality is an advantage, as specific applications (or classifiers) requiring synthesized features may have strict computational (or other) constraints on the FS dimensions. Such requirements can be met with the EFS technique by setting the output dimension range equal to the requested one, or by setting it to a single specific value. Furthermore, by allowing the output vector dimension vary during the synthesis process, the most suitable FV dimension can be found *concurrently* in a single experiment, obviating the need to perform separate experiments with different (fixed) FV dimensions.

Computational complexity

Considering the computational complexity of the SLP, MLP, and EFS approaches, the most important factor is the number of particles and iterations applied to the underlying MD PSO algorithm. Their selection is influenced by the number and identity of the FV dimensions synthesized (i.e., database size, feature extraction approaches, and the original FSs). In the case of EFS, the synthesis depth, K, is another important factor, as in every particle element the synthesis process consists of $K - 1$ input features. There is usually a trade-off between computation time and performance attained, so that the parameters affecting processing time should be set depending on the specific task.

Table 13 The best retrieval performances obtained by ten consecutive SLP classifiers

FS	Basic database (6 classes)		Extended database (12 classes)	
	ANMRR (run no.)	AP (%) (run no.)	ANMRR (run no.)	AP (%) (run no.)
Segment				
STAT	0.206 (run 3)	78.6 (run 3)	0.579 (run 2)	40.7 (run 2)
MFCC	0.109 (run 3)	89.7 (run 3)	0.479 (run 2)	51.2 (run 2)
Δ-MFCC	0.295 (run 2)	69.8 (run 2)	0.534 (run 3)	45.2 (run 3)
ΔΔ-MFCC	0.269 (run 10)	72.4 (run 10)	0.579 (run 4)	41.3 (run 4)
LPC	0.483 (run 2)	50.9 (run 2)	0.667 (run 4)	32.3 (run 4)
LPCC	0.550 (run 2)	44.2 (run 2)	0.714 (run 1)	27.4 (run 1)
S_AUDIO	0.202 (run 5)	79.4 (run 5)	0.541 (run 2)	44.7 (run 2)
Key-frame				
MFCC + deltas	0.279 (run 1)	70.2 (run 1)	0.414 (run 1)	56.5 (run 1)
LPC + LPCC	0.669 (run 1)	32.1 (run 1)	0.794 (run 1)	19.7 (run 1)
K_AUDIO	0.277 (run 1)	69.6 (run 1)	0.504 (run 1)	47.1 (run 1)

The structure of the proposed feature synthesis technique is particularly designed for *parallel* computing, so that grid computing[d] was utilized in computing the results. Here, each FS was processed on its own, as shown in Figure 4. With the applied MD PSO and EFS parameters, the computational time of a single EFS run varied between 30 min (segment features of the basic database) and 3.5 h (key-frame features of the extended database). The corresponding computation times for the SLP and MLP classifiers trained with MD PSO were more or less the same. However, it should be kept in mind that once the synthesis (or network) parameters are found, features of any previously unseen data can be synthesized afterwards with no need for any further search or training processes.

Conclusions

A method for transforming and modifying traditional audio features by an evolutionary optimization algorithm is proposed, one that improves feature discrimination as

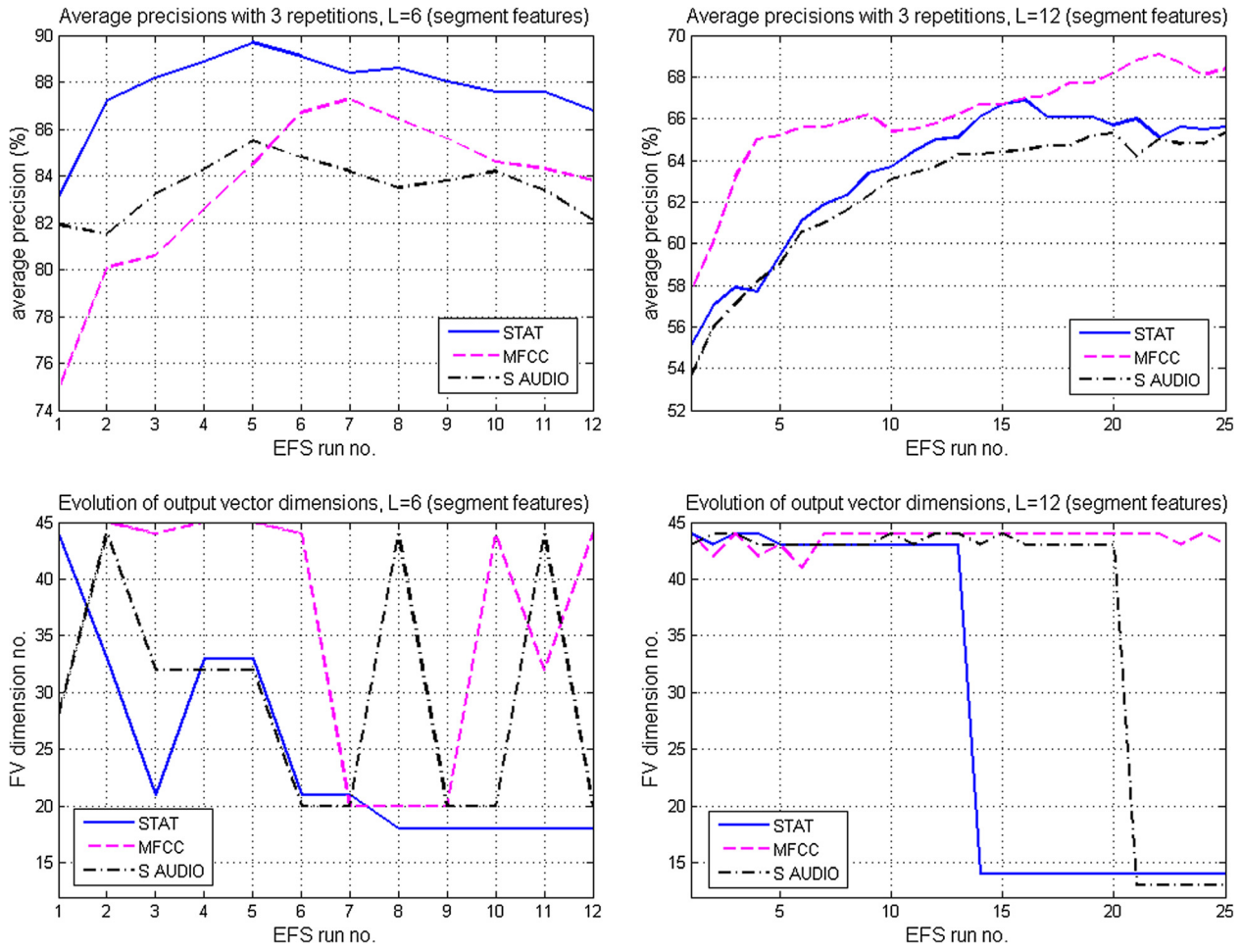

Figure 9 The consecutive EFS runs with three repetitive optimization performances within each run. Significant further improvements were obtained to the AP results also with low output FV dimensions.

well as audio classification and retrieval performance. The process, EFS, is a *generalized form* of feed-forward neural networks. In addition to traditional ANNs, the EFS technique provides (1) numerous linear/nonlinear attribute operators, (2) a built-in feature selection scheme, and (3) dynamic output (layer) dimension.

The experimental results obtained from clustering, *K*-means classification, and several audio retrieval tasks demonstrate the ability of the technique to provide substantial improvements compared with the original features. It was also shown that the audio retrieval performance can be further improved by performing several synthesis runs, whereas comparable performance could not be achieved using concatenated SLP evaluations. Based on these experiments, the synthesis approach appears capable of producing more descriptive/discriminative (artificial) features than those designed and selected by humans (i.e., the traditional low-level audio features).

The underlying optimization algorithm used to discover the optimal feature synthesis parameters is based on PSO, extended in our study by the addition of three properties previously introduced in the literature. First, an MD PSO was applied, which enables optimizing the output FV dimension during the synthesis process. Second, an FGBF technique was used to increase the probability of converging to the global optimum of the search space and third, the heterogeneous particle behavior was addressed via the swarm in order to avoid local optima in the fitness function surface.

In this study, the optimization of output vector dimension was considered only with respect to classification and retrieval performance. As a result, the best output vector dimension found by the EFS was usually higher than that of the original FS. This may be an important aspect for certain applications where there is considerable post-processing for the output vectors. However, an appropriate upper limit for the output vector dimensionality can be determined and tuned to the specific problem to be solved. Some of the experiments using the basic 6-class database suggest that when it comes to discriminating features of a relatively small and distinguishable database, the EFS technique may not be worth implementing. However, with the larger and more overlapping (in terms of the pre-determined data classes) 12-class database, regular neural networks were not able to achieve as significant improvements to the retrieval performance as those obtained with the EFS.

In general, the EFS framework can be used in any such tasks in which it is applicable to take advantage of enhanced feature discrimination. For example, content-based classification and knowledge mining are suitable venues for the proposed framework, as are those in which the quality and description power of the applied features play an essential role. In future research, the synthesis performance may be enhanced by experimenting with other optimization techniques (such as simulated annealing and GAs) or by optimizing the mathematical operators used. This would require analyzing operator selection during the synthesis process so that some statistical conclusions could be drawn about the selection behavior. A similar analysis could be applied, with caution, to the selection of the original features in order to avoid using "vain" (or very rarely selected) features. Unlike general feed-forward ANNs, where regular gradient-descent training methods (such as back-propagation) are designed to minimize fixed and differentiable fitness functions (such as MSE), the EFS technique can be used to minimize any types of fitness functions. In this way, the optimization tasks can be attuned to the specific goals of the research, such as audio retrieval, as closely as possible. Designing additional fitness functions for particular problems is thus one of the main advantages of the proposed feature synthesis scheme. Such issues will be addressed in future research.

Endnotes

[a]TIMIT (Texas Instruments, Massachusetts Institute of Technology) is a corpus of phonemically and lexically transcribed speech of American English male and female speakers of different dialects. [b]RWC (Real World Computing) Music Database is a copyright-cleared music database that is available to researchers as a common foundation for research (http://staff.aist.go.jp/m.goto/RWC-MDB/). [c]The *StockMusic.com* web shop (http://www.stockmusic.com/). [d]Techila Technologies Ltd., *Techila Grid*, http://www.techila.fi/.

Competing interests

Toni Mäkinen is a PhD student at the Tampere University of Technology (TUT), from where he receives salary. In order to graduate, scientific publications are required. The TUT will gain financially from the graduation itself in the future, but not separately from any specific publication (such as this one). The authors thus declare that they have no competing interests.

Acknowledgments

The authors would also like to thank Toni Heittola, Anna-Maria Mesaros, and Tuomas Virtanen from the Tampere University of Technology for kindly providing several audio databases for testing during this research.

Received: 13 March 2012 Accepted: 13 August 2012
Published: 11 September 2012

References

1. R.O. Duda, P.E. Hart, D.G. Stork, *Pattern Classification*, 2nd edn. (Wiley-Interscience Publication, 2001)
2. R.E. Bellman, *Adaptive Control Processes—A Guided Tour (Princeton University Press* (Princeton, NJ, 1961)
3. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st edn. ((Addison-Wesley Longman Publishing Co, Boston, MA, 1989)
4. J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press, Cambridge, MA, 1992)