Particle Swarm Optimization for Sorted Adapted Gaussian Mixture Models

Rahim Saeidi, Hamid Reza Sadegh Mohammadi, *Member, IEEE*, Todor Ganchev, *Member, IEEE*, and Robert David Rodman

***Abstract*—Recently, we introduced the sorted Gaussian mixture models (SGMMs) algorithm providing the means to tradeoff performance for operational speed and thus permitting the speed-up of GMM-based classification schemes. The performance of the SGMM algorithm depends on the proper choice of the sorting function, and the proper adjustment of its parameters. In the present work, we employ particle swarm optimization (PSO) and an appropriate fitness function to find the most advantageous parameters of the sorting function. We evaluate the practical significance of our approach on the text-independent speaker verification task utilizing the NIST 2002 speaker recognition evaluation (SRE) database while following the NIST SRE experimental protocol. The experimental results demonstrate a superior performance of the SGMM algorithm using PSO when compared to the original SGMM. For comprehensiveness we also compared these results with those from a baseline Gaussian mixture model–universal background model (GMM-UBM) system. The experimental results suggest that the performance loss due to speed-up is partially mitigated using PSO-derived weights in a sorted GMM-based scheme.**

***Index Terms*—Gaussian mixture model–universal background model (GMM-UBM), particle swarm optimization (PSO), sorted GMM, speed-up, text-independent speaker verification.**

I. Introduction

A Gaussian mixture model (GMM) is a common baseline system in speaker recognition applications [1]. Such a system is normally used as a reference when one needs to evaluate the effectiveness of novel algorithms or modeling approaches. The GMM is a statistical approach for text-independent speaker recognition with a high computational load during the test phase. A popular method for training GMMs is based on the maximum-likelihood (ML) criterion, which has been shown to outperform several other existing techniques. In state-of-the-art systems, speaker-dependent GMMs are derived from a speaker-independent universal background model (UBM) by adapting the UBM components with maximum *a posteriori* (MAP) adaptation using speakers' personal training data [2]. This method constructs a natural association between the UBM and the speaker models. For each UBM Gaussian component there is a corresponding adapted component in the speaker's GMM. In the verification phase each test vector is scored against all UBM Gaussian components, and a small number of the best-scoring components in the corresponding speaker-dependent adapted GMM are chosen. The decision score is computed as the log likelihood ratio (LLR) of the speaker GMM and the UBM scores. Because of the need to score twice—against both the UBM and speaker-dependent GMM, and the tendency to have large-sized GMMs—the computational load during the test phase is high.

Chan *et al.* have categorized the existing methods for fast GMM computation in four layers, referred to as: frame-layer, GMM-layer, Gaussian-layer, and component-layer [3]. Here, we consider the Gaussian-layer scheme of this categorization as a reference point to describe the SGMM algorithm [4]. The speed-up concept of a GMM-UBM-based system with preprocessing was previously investigated for speaker verification systems in [5], [6]. In the literature, various speed-up approaches were reported to reduce the computational complexity, such as the use of Gaussian selection (hash GMM) [7], vector quantization (VQ) pre-classifier [8], structural GMM [9], tree-structured Gaussian densities [10], pruning methods [11], [12], approximated cross entropy (ACE) [13], and discriminative mixture selection [14]. These methods degrade the system performance to some extent in exchange for gaining speed-up.

In the present contribution, we build on the SGMM algorithm [4], which belongs to the group of methods operating in the Gaussian-layer. Investigation of the SGMM speed-up concept for GMM-UBM-based classification schemes on the speaker verification task is documented in [5], [6]. Here, we elaborate further on the SGMM algorithm by introducing an efficient scheme for adjusting the parameters of the sorting function by means of the PSO algorithm [16].

The remainder of this paper is organized as follows. In Section II, the GMM-based classification method and its sorted variant, SGMM, are described. The PSO algorithm is explained in Section III. Section IV presents the computer simulation and experimental results. Finally, Section V concludes the paper.

Manuscript received March 26, 2008; revised October 05, 2008. Current version published January 14, 2009. This work was supported by the Iranian Research Institute for Electrical Engineering, ACECR. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Richard C. Rose.

R. Saeidi and H. R. S. Mohammadi are with the Iranian Research Institute for Electrical Engineering (IRIEE), Academic Center for Education, Culture, and Research (ACECR), Tehran, Iran (e-mail: rahim.saeidi@gmail.com; h.sadegh@ijece.org).

T. Ganchev is with the Wire Communications Laboratory, Electrical and Computer Engineering Department, University of Patras, 26 500 Rio-Patras, Greece (e-mail: tganchev@ieee.org).

R. D. Rodman is with the Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206 USA (e-mail: rodman@ncsu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASL.2008.2010278

II. GMM-UBM-Based Speaker Verification

A. Principals

Given a segment of speech $\mathbf{X}$ and a hypothesized speaker S, the task of speaker verification—also referred to as one-speaker detection—is to determine whether $\mathbf{X}$ was spoken by S. (Throughout this paper we refer to $\mathbf{X}$ as a speech segment or, alternatively, as a sequence of feature vectors extracted from a speech segment. In the later case, $\mathbf{x}_t$ is used to represent one of the feature vectors.) An implicit assumption often used is that $\mathbf{X}$ contains speech from only one speaker. The single-speaker detection task can be stated as a basic hypothesis test between two hypotheses.

H_0 $\mathbf{X}$ is from the hypothesized speaker S.

H_1 $\mathbf{X}$ is not from the hypothesized speaker S.

The optimum test to decide between these two hypotheses is a likelihood ratio (LR) test given by

$$\frac{p(\mathbf{X}|H_0)}{p(\mathbf{X}|H_1)} \begin{cases} \geq \theta, & \text{Accept } H_0 \\ < \theta, & \text{Accept } H_1 \end{cases} \tag{1}$$

where $p(\mathbf{X}|H_0)$ is the probability density function for the hypothesis H_0 evaluated for the observed speech segment $\mathbf{X}$, also referred to as the likelihood of the hypothesis H_0 given the speech segment. The likelihood function for H_1 is likewise $p(\mathbf{X}|H_1)$. The decision threshold for accepting or rejecting H_0 is θ. The main goal in designing a speaker detection system is to determine techniques for computing values for the two likelihoods $p(\mathbf{X}|H_0)$ and $p(\mathbf{X}|H_1)$ [17].

The output of the speech parameterization stage is typically a sequence of feature vectors. Mathematically, a model denoted by λ_{hyp} represents H_0, which characterizes the hypothesized speaker S in the feature space. $\lambda_{\overline{hyp}}$ represents the alternative hypothesis, H_1. The likelihood ratio statistic is then $p(\mathbf{X}|\lambda_{hyp})/p(\mathbf{X}|\lambda_{\overline{hyp}})$. Often, the logarithm of this statistic is used giving the log-likelihood ratio as

$$\Lambda(\mathbf{X}) = \log\left[p(\mathbf{X}|\lambda_{hyp})\right] - \log\left[p\left(\mathbf{X}|\lambda_{\overline{hyp}}\right)\right] \tag{2}$$

While the model for H_0 is well defined and can be estimated using training speech from S, the model for $\lambda_{\overline{hyp}}$ is less well defined since it potentially must represent the entire space of possible alternatives to the hypothesized speaker.

The widely used approach to the alternative hypothesis modeling is to pool speech from a large number of speakers and train a single model. Various terms for this single model are a *general model*, a *world model*, and a UBM [2]. Given a collection of speech samples from a large number of speakers which are representative of the population of speakers expected during verification, a single model λ_{UBM} is trained to represent the alternative hypothesis. The main advantage of this approach is that a single speaker-independent model can be trained once for a particular task and then used for all hypothesized speakers in the same task. It is also possible to use multiple background models tailored to specific sets of speakers. The use of a single background model has become the predominant approach used in speaker verification systems.

B. Gaussian Mixture Models

Having $\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^{T}$ where $\mathbf{x}_t$ is a D-dimensional feature vector at instance t and T is the total number of feature vectors, an important step in the implementation of the aforementioned likelihood ratio is the selection of the actual likelihood function $p(\mathbf{x}_t|\lambda)$. The choice of this function largely depends on the features being used as well as on the specifics of the application. The mixture density used for the likelihood function is defined [2] as follows:

$$p(\mathbf{x}_t|\lambda) = \sum_{i=1}^{M} w_i p_i(\mathbf{x}_t). \tag{3}$$

The density is a weighted linear combination of M unimodal Gaussian densities $p_i(\mathbf{x}_t)$, each parameterized by a $D\times 1$ mean vector $\boldsymbol{\mu}_i$ and a $D \times D$ covariance matrix Σ_i. Here, $p_i(\mathbf{x}_t)$ are defined as

$$p_i(\mathbf{x}_t) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \times \exp\left\{-\frac{1}{2}(\mathbf{x}_t - \boldsymbol{\mu}_i)'\Sigma_i^{-1}(\mathbf{x}_t - \boldsymbol{\mu}_i)\right\}. \tag{4}$$

The mixture weights, w_i, further satisfy the constraint $\sum_{i=1}^{M} w_i = 1$. While the general form of the model supports full covariance matrices; that is, a covariance matrix with all its elements, typically only diagonal covariance matrices are used in order to reduce the number of adjustable parameters, hence the amount of computation [2]. Collectively, the parameters of the density model are denoted as $\Sigma = \{w_i, \boldsymbol{\mu}_i, \boldsymbol{\sigma}_i\}$, where $i= (1,\ldots,M)$.

Given a collection of training vectors, maximum-likelihood model parameters are estimated using the expectation-maximization (EM) algorithm. The EM algorithm iteratively refines the GMM parameters to monotonically increase the likelihood of the estimated model for the observed feature vectors. Under the assumption of independent feature vectors, the log-likelihood of a model λ for a sequence of T feature vectors is computed as follows:

$$L(\lambda) = \frac{1}{T}\sum_{t=1}^{T} \log[p(\mathbf{x}_t|\lambda)]. \tag{5}$$

C. Adapted GMM

As discussed earlier, the dominant approach to background modeling is to use a single, speaker-independent background model to represent $p(\mathbf{X}|\lambda_{\overline{hyp}})$. Using a GMM as the likelihood function, the background model is implemented typically as a large GMM trained to represent the speaker-independent distribution of features. Specifically, the training set should be selected in such a way as to reflect the expected alternative speech to be encountered during recognition.

In the original GMM scheme [1], the speaker model is trained using the EM algorithm on the speaker's enrollment data. In the adapted GMM, the speaker model is derived by adapting the parameters of the background model using the speaker's training speech and a form of Bayesian adaptation known as maximum *a posteriori* (MAP) estimation [18]. The rationale underlying this method is to derive the speaker's model by updating the well-trained parameters in the background model via adaptation as opposed to the standard approach, where the maximum likelihood training of a model for the speaker occurs independently of the background model [1]. This provides a tighter association between the speaker's model and the background model that not only produces better performance than decoupled models, but, as discussed later in this section, also allows for a fast-scoring technique. Evidence shows that instead of adapting all UBM parameters, adapting only the mean vectors provides the best performance [2].

The adapted GMM approach also leads to a fast-scoring technique. Computing the LLR requires computing the likelihood for the speaker and background models for each feature vector, which can be computationally expensive for large mixture orders. However, the fact that the hypothesized speaker model was adapted from the background model allows a faster scoring method because the components of the adapted GMM retain a correspondence with the mixtures of the background model so that vectors close to a particular mixture in the background model will also be close to the corresponding mixture in the speaker model.

The fast-scoring procedure operates as follows. For each feature vector, determine the top-C scoring mixtures in the background model and compute the background model likelihood using only these top-C mixtures. Next, score the vector against only the corresponding C components in the adapted speaker model to evaluate the speaker's likelihood. For a background model with M mixtures, this requires only $M + C$ Gaussian computations per feature vector compared to $2M$ Gaussian computations for normal likelihood ratio evaluation. When there are multiple hypothesized speaker models for each test segment, the savings become even greater. Typically, a value of $C = 5$ is used.

D. Sorted GMM

The sorted Gaussian mixture model (refer to Fig. 1) is a recently reported method for the fast scoring GMM [4], [6]. Given a D-dimensional feature vector $\mathbf{x}_t = [x_{1t}, x_{2t}, \ldots, x_{Dt}]^T$ related to the speech frame at the time instance t, and a GMM of order M, we define a *sorting parameter* $s_t = f(x_{1t}, x_{2t}, \ldots, x_{Dt})$, which is a scalar. Here, $f(\cdot)$ is a *sorting function*, chosen in such a way that neighboring target feature vectors provide almost neighboring values of s_t. The mixtures of the GMM are sorted in ascending order of the associated sorting parameter of their mean vectors according to the vector $\mathbf{S}_{\mathrm{UBM}} = [s_1, s_2, \ldots, s_M]^T$ with $s_1 \leq s_2 \leq \cdots \leq s_M$.

To compute the likelihood of an unknown input feature vector we first scalar quantize s_t by $\mathbf{S}_{\mathrm{UBM}}$ giving s_i $1 \leq i \leq M$. We name i central index. Next, we evaluate the input feature vector's likelihood using the ordinary method by an extensive local search in the neighborhood of the central index i which includes a subset of M_s mixtures where $M_s < M$. For example, only the mixtures with indices within the range of $i - k + 1$ to $i + k$ may be searched, where k is an offset value ($k = M_s/2$). M_s is called the search width.

To achieve a better performance for the SGMM, we always search $2k$ mixtures. For example, for the case $i \leq k$, the first $2k$ mixtures in the GMM are considered for local search, and for $i \geq M - k$ the last $2k$ mixtures are evaluated for the likelihood calculation. Generally, the computational complexity of this method grows linearly with M_s, which normally is set to be less than M. In Fig. 1, we summarize the structure of a SGMM system, and in Fig. 2 we illustrate the mixture selection process.

In previous work [4], [6], we relied on the sorting function defined as follows:

$$s_t = f(\mathbf{x}_t) = \sum_{i=1}^{D} x_{it}. \tag{6}$$

This sorting function will be considered as the baseline for comparison with the PSO-optimized one.

The SGMM method can be applied to any GMM, such as a UBM, without any further training process. However, the overall performance can be further enhanced if the following optimization algorithm is used to optimize the GMM.

Step 1) *Initialization*: Set $k = 0$, $\lambda_k = \lambda_{\mathrm{UBM}}$, where λ_{UBM} is the EM derived GMM. Calculate the sorting parameter related to each mixture and sort the GMM in ascending order of the sorting parameter.

Step 2) *Likelihood Estimation*: Calculate the likelihood of the entire training database with λ_k mixtures using the SGMM method.

Step 3) *GMM Adaptation*: Compute the λ_{k+1} GMM. This is done simply by adapting each mixture from λ_k using associated training vectors found in the previous step.

Step 4) *Sorting*: Recalculate the sorting parameters related to the mixtures of the new GMM λ_{k+1} and sort the GMM in ascending order of the sorting parameter. Also, set $k = k + 1$.

Step 5) *Termination*: If the total likelihood is higher than a certain threshold the algorithm terminates; otherwise, go back to Step 2.

After the optimization stage with the optimized UBM at hand, the speakers' GMMs are simply adapted from the optimized UBM in the same way the ordinary GMM-UBM training is performed. The memory storage required for the SGMM is $(2D + 2)/(2D + 1)$ times that needed for the ordinary GMM. The negligible extra storage is required to store the sorting parameter quantization table. On the other hand, the number of Gaussian computations (which is used as a measure of speed-up) is reduced to $M_s + C$ (where C is the number of top scoring mixtures whose corresponding mixtures are evaluated in the speaker GMM). This is less than $M + C$ which is the number of Gaussian computations in the baseline system. Thus, the speed-up factor of the SGMM algorithm is approximately equal to $(M + C)/(M_s + C)$. Here we ignore the computations used to find the M_s Gaussians which are negligible compared to that of the Gaussian evaluation computations. To incorporate

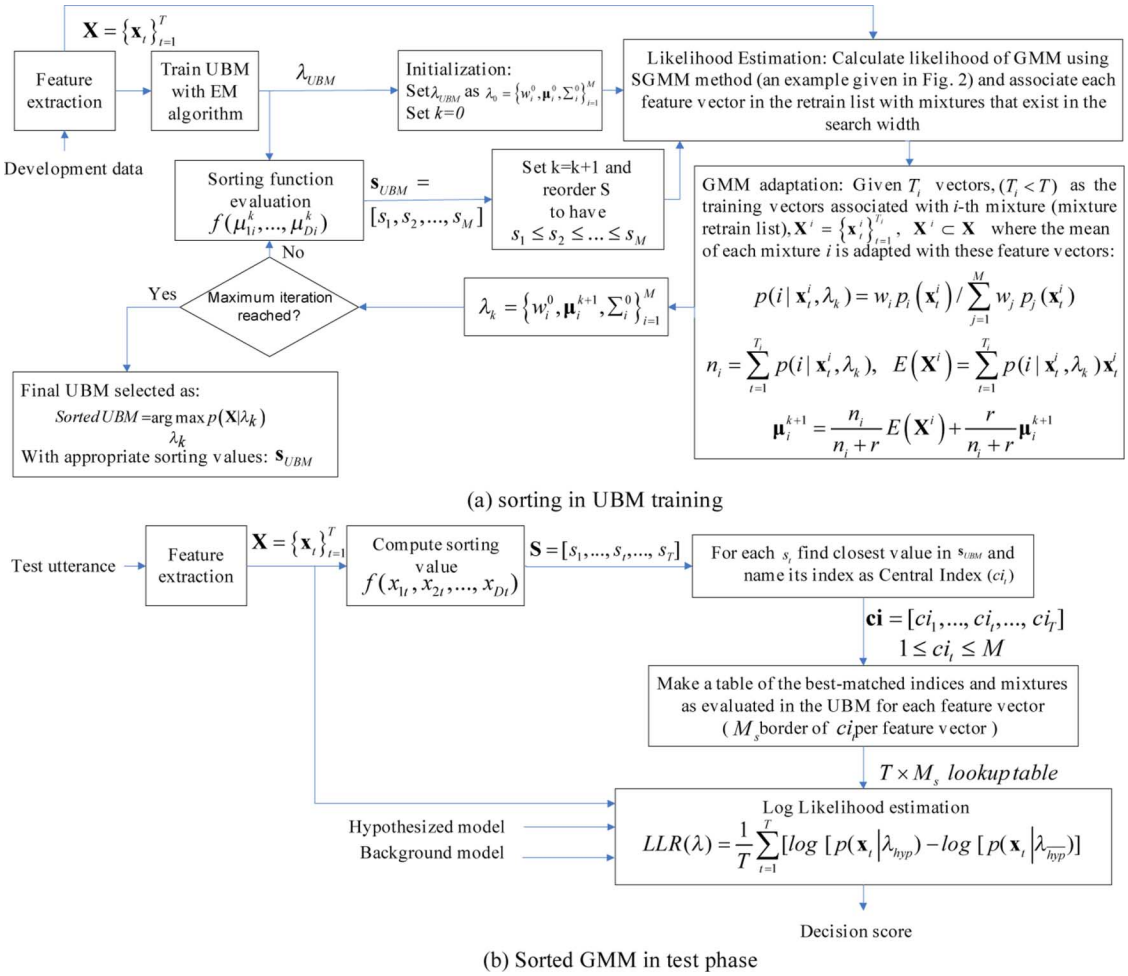

Fig. 1. Simplified flow diagram of the SGMM algorithm. (a) Training stage. (b) Test stage.

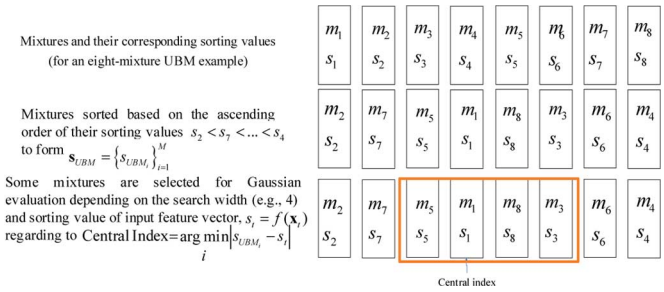

Fig. 2. Example of SGMM algorithm performance for a simple GMM of order 8.

Tnorm in the score calculation, N Tnorm speakers are considered; therefore, the speed-up factor of SGMM is reduced to $(M + NC)/(M_s + NC)$.

III. Particle Swarm Optimization

Swarm intelligence (SI) is an innovative distributed intelligence paradigm for solving optimization problems that originally took its inspiration from the biological phenomena of swarming, flocking, and herding [19]. Particle swarm optimization (PSO) incorporates group behaviors observed in flocks of birds, schools of fish, swarms of bees, and even human social behavior, from which the idea emerged. PSO is a population-based optimization tool which can be easily implemented and applied to solve various function optimization problems, or the problems that can be transformed into function optimization problems. As an algorithm, the main strength of PSO is its fast convergence, which compares favorably with many global optimizations. Following the original PSO [15], various modifications and improvements have been reported in the literature to enhance its stability and performance. In the present work, we rely on the PSO of Clerc [16], which is known with its robust operation.

The PSO algorithm is employed here in a scheme for adjustment of the weight coefficients of the sorting function outlined in Section II-D. With this optimization scheme, we aim at estimating data-driven importance factors for the different members of the speech feature vector, hoping to enhance the overall speaker verification performance. Before describing in details the optimization of the sorting function (Section III-B), for the purpose of comprehensiveness we will outline briefly the basics of the PSO algorithm.

A. PSO Algorithm Description

We adhere to standard PSO which is described in [16]. Let $h(\mathbf{y}) : \Re^m \rightarrow \Re$ be the cost function. Let there be n particles, each with associated positions $\mathbf{y}_i \in \Re^m$ and velocities $\mathbf{v}_i \in$

$\Re^m, i = 1, \ldots, n$. Let $\hat{\mathbf{y}}_i$ be the current best position (PBest) of each particle and let $\hat{\mathbf{g}}_i$ be the global best (GBest). In summary the PSO algorithm consists of the following steps.

- Initialize $\mathbf{y}_i$ and $\mathbf{v}_i$ for all i. One common choice is to take $\mathbf{y}_{ij} \in U(a_j, b_j)$ and $\mathbf{v}_i = 0$ for all i and $j = 1, \ldots, m$, where U is a uniform pdf, and a_j, b_j are the limits of the search domain in each dimension.
- $\hat{\mathbf{y}}_i \leftarrow \mathbf{y}_i$ and $\hat{\mathbf{g}} \leftarrow \arg\min_{y_i} h(\mathbf{y}_i), i = 1, \ldots, n$.
- While not converged:
  For $1 \leq i \leq n$
  - $\mathbf{y}_i \leftarrow \mathbf{y}_i + \mathbf{v}_i$.
  - $\mathbf{v}_i \leftarrow \omega \mathbf{v}_i + c_1 \mathbf{r}_1 \otimes (\hat{\mathbf{y}}_i - \mathbf{y}_i) + c_2 \mathbf{r}_2 \otimes (\hat{\mathbf{g}} - \mathbf{y}_i)$.
  - If $h(\mathbf{y}_i) < h(\hat{\mathbf{y}}_i)$ Then $\hat{\mathbf{y}}_i \leftarrow \mathbf{y}_i$.
  - If $h(\mathbf{y}_i) < h(\hat{\mathbf{g}})$ Then $\hat{\mathbf{g}} \leftarrow \mathbf{y}_i$.

In the above equations, the symbol " $\otimes$ " stands for element-by-element vector multiplication.

- ω is an inertia constant. It is usual to decrease ω linearly from an initial value to a final value while running iterations.
- c_1 and c_2 are constants that define the extent to which the particle is directed towards improved positions. They represent a "cognitive" and a "social" component, respectively, in that they affect the extent to which the particle's individual best and its global best influence its movement.
- $\mathbf{r}_1$ and $\mathbf{r}_2$ are two random vectors, whose elements are random numbers uniformly distributed between 0 to 1.

The particles search the space for better solutions by learning from their own and their neighbors' experiences. The learning factors c_1 and c_2 represent the weights of the stochastic acceleration terms that pull each particle toward PBest (cognitive behavior) and GBest (social behavior) positions. The velocities of the particles are clamped to a maximum velocity $\mathbf{v}_{\max}$, which serves as a constraint on the speed of the particles to avoid global explosion. It limits the maximum step change of the particle, thus constraining the moving speed of the entire population in the hyperspace. Generally, $\mathbf{v}_{\max}$ is set to the value of the dynamic range of each variable to prevent particles from leaving the problem space, so that no inherent limitations are introduced. If $\mathbf{v}_{\max}$ was set to a lower value, it might slow down the convergence speed of the algorithm, although, on the other hand, it might help prevent PSO from local convergence. To prevent dealing with this, we used the dynamic range of each variable as maximum speed. The PSO terminates when the prespecified number of iterations has taken place or when no improvement of GBest has been achieved within a specified number of iterations. As well, iterations may terminate when a satisfactory value of the fitness function is reached.

B. PSO of the Sorting Function

Suppose a sequence of feature vectors is given as $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$. One can write this sequence in the form of $\mathbf{X} = \left[\mathbf{c}_1^T, \ldots, \mathbf{c}_{D/3}^T, \Delta\mathbf{c}_1^T, \ldots, \Delta\mathbf{c}_{D/3}^T, \Delta\Delta\mathbf{c}_1^T, \ldots, \Delta\Delta\mathbf{c}_{D/3}^T\right]^T$ (superscript T stands for matrix Transpose and in all other cases T indicates the number of feature vectors per utterance) where $\mathbf{c}_i = [c_{i1}, \ldots, c_{iT}]$, $\Delta\mathbf{c}_i = [\Delta c_{i1}, \ldots, \Delta c_{iT}]$ and $\Delta\Delta\mathbf{c}_i = [\Delta\Delta c_{i1}, \ldots, \Delta\Delta c_{iT}]$ are the ith Mel-frequency cepstral coefficent (MFCC), ΔMFCC and $\Delta\Delta$MFCC over all feature vectors, respectively. For convenience, we rewrite the feature vectors as $\mathbf{X} = \left[\mathbf{x}_1'^T, \ldots, \mathbf{x}_D'^T\right]^T$, where the $\mathbf{x}_i'$s represent the MFCCs for $1 \leq i \leq D/3$, ΔMFCCs for $D/3 < i \leq 2D/3$ and $\Delta\Delta$MFCCs for $2D/3 < i \leq D$ over all feature vectors.

The introduction of a sorting function as the sum of feature vector elements is based on correlations between sorting function values $\mathbf{s} = [s_1, \ldots, s_T]$ and $\mathbf{x}_i'$s, where they are highly correlated for low index values such as $\mathbf{x}_1'$, $\mathbf{x}_2'$ and $\mathbf{x}_3'$ with correlations decreasing to $\mathbf{x}_D'$. It is straightforward that a sorting function which generates sorting values better correlated with $\mathbf{x}_i'$s would offer better results in the Gaussian selection stage.

If we consider $\mathbf{X} = [x_{it}], 1 \leq i \leq D, 1 \leq t \leq T$ (x_{it} stands for ith MFCC in the tth feature vector) and $\mathbf{s}' = [s_t'], 1 \leq t \leq T$ then we have $\mathbf{x}_i' = [x_{it}], 1 \leq t \leq T$ and $\mathbf{x}_t = [x_{it}], 1 \leq i \leq D$, so that the sorting function (6) can be redefined as

$$s_t' = f'(\mathbf{x}_t) = \mathbf{a}.\mathbf{x}_t = \sum_{i=1}^{D} a_i x_{it}. \tag{7}$$

The new sorting function is considered as a weighted sum of the elements x_{it} of a feature vector, while in the earlier formulation (6), unit weights were assumed. In the new more general form of the sorting function the adjustable weights $\mathbf{a}$ can be learned in a data-dependent manner. The fitness function that is function of the weights $\mathbf{a}$ is defined as follows:

$$Fitness(\mathbf{a}) = \sum_{i=1}^{D} \frac{E\{(\mathbf{x}_i' - E(\mathbf{x}_i'))\,(\mathbf{s}' - E(\mathbf{s}'))\}}{\sqrt{E\{(\mathbf{x}_i' - E(\mathbf{x}_i'))^2\} E\{(\mathbf{s}' - E(\mathbf{s}'))^2\}}}. \tag{8}$$

Here, $E\{.\}$ stands for mathematical expectation. The weights $\mathbf{a}$ should be chosen in such a way that the defined fitness function is maximized. This definition of the fitness function is motivated by previous experience [4], [20], where summing all correlations between s' and $\mathbf{x}_i'$s led to positive results. The optimization problem defined so far has the goal of finding out the optimal weights for the sorting function $\mathbf{a}_{\text{opt}}$ so as to attain the maximum correlations.

We expect that by maximizing the fitness function (8), we optimize the weights of the sorting function (7) in such a way that neighboring feature vectors would result in almost-neighboring sorting values. These would consequently correspond to the most valuable mixtures for this purpose in that they would provide a level of discrimination comparable to top-C selection in conventional GMM-UBM systems. (Here, we assume that if we chose the appropriate sorting function, we would reach the same likelihood value of UBM, as well as the same log likelihood ratio for a specific speaker, by evaluating a smaller number of Gaussians M_s when compared to the model order M). Because the search space is unknown, such an optimization algorithm is needed that is capable of searching a wide area while avoiding local maximums.

Compared with basic sorted GMM where only M_s must be estimated for system performance, in PSO optimization of SGMM weights we need to estimate other parameters of PSO such as inertia weight and learning rates. This may be viewed as a weak point for using PSO in SGMM as we used standard

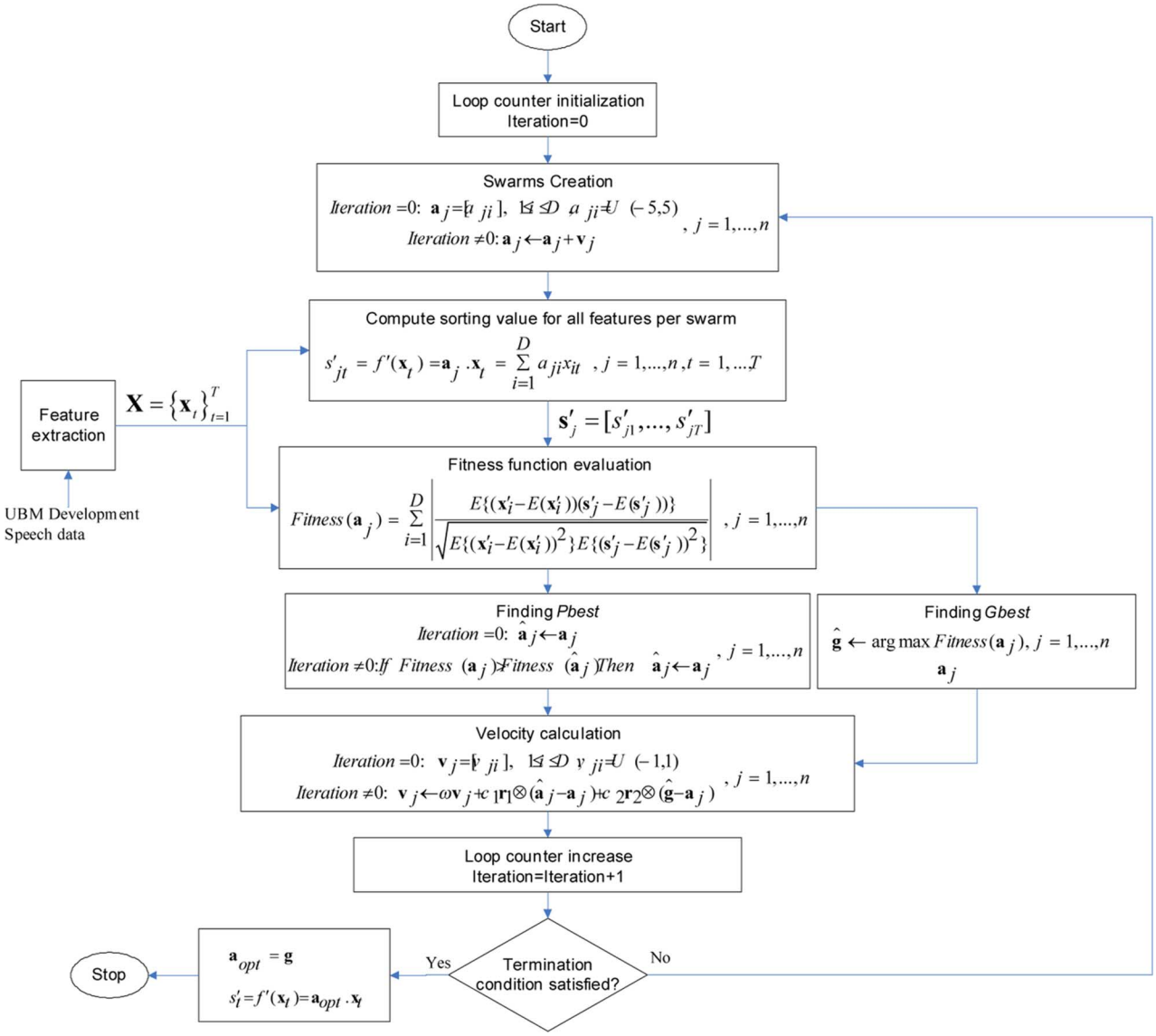

Fig. 3. Block diagram of fitness function optimization via PSO iterations.

PSO with recommended parameters and achieved good results without further tuning.

C. PSO Utilization

In the experiments, we set $\mathbf{X}$ to be the total feature vector for the UBM training, which is a $D \times T$ matrix (D and T values were used as 36 and 122 364 for males and 36 and 118 370 for females, respectively). In Fig. 3, we present a block diagram of the operation of the PSO-based weights adjustment. We relied on a swarm of one hundred particles, whose elements were randomly initialized to take values in the range of $(-5, 5)$. The inertia weight was set to linearly decrease from 0.94 to 0.4 during the first 70% of 1000 iterations of optimization, and then remain constant. Both learning factors were set to 1 based on initial experiments on the dataset. Termination occurred when the algorithm reached its maximum number of iterations or when the fitness function value of the best particle remained constant for 250 iterations. The maximum value of particles' velocity was confined to the default range of $(-5, 5)$ to avoid divergence of the swarm. To obtain a different set of optimal weights for males and females this process was repeated for each gender-specific UBM.

In Fig. 4, we show the result of the PSO optimization on the value of the fitness function for the male and female cases. (The evolution of the values of the multidimensional weights vector $\mathbf{a}$ is difficult to illustrate). Initial values shown in Fig. 4 are for unit weights, as they are in the basic formulation of the sorting function (6). The final values are those estimated via the PSO. The figure indicates that the PSO enhanced the fitness function significantly and therefore we assume that the adjustment of the weights $\mathbf{a}$ is beneficial.

IV. Performance Evaluation

To evaluate the practical significance of the proposed PSO-based SGMM speed-up method, we performed a series of experiments. After detailing the experimental setup, we explain different aspects of these trials.

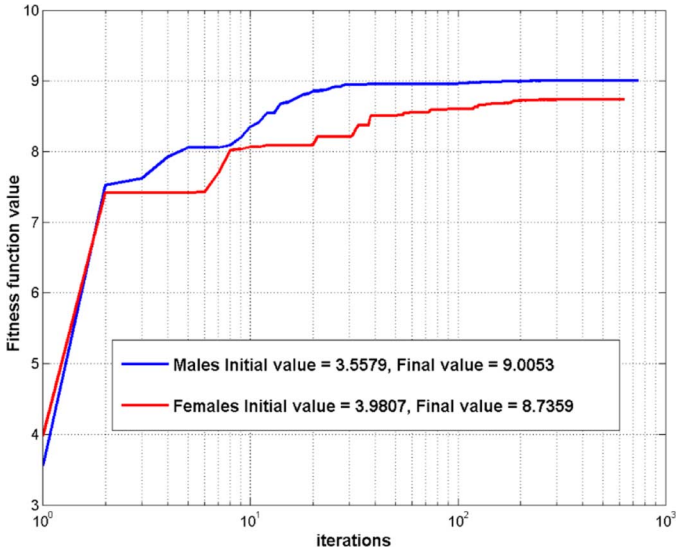

Fig. 4. Fitness function optimization during PSO iterations.

A. Speech Database

The speaker recognition experiments were conducted on the NIST 2002 SRE data [21], which consist of cellular telephone conversational speech and excerpts from the Switchboard corpus. The NIST SRE experimental protocol as defined in the one-speaker detection task was used. In brief, given a speech segment of about 30 s, the goal is to decide whether this segment was spoken by a specific target speaker or not. For each of 330 target speakers (139 males and 191 females), two minutes of untranscribed, concatenated speech is available for the training of the target model. Overall 3570 test segments (1442 males and 2128 females), mainly lasting between 15 and 45 s, must be scored against roughly ten gender-matching impostors and against the true speaker. The gender of the target speaker is known. A subset of the NIST 2001 speech data was used to train the universal background model and impostor speakers' model for use in Tnorm score normalization [22]. This subset consists of 174 target speakers, 74 males and 100 females. To avoid any bias in results, the speakers from NIST 2001 SRE who are present in the 2002 evaluation were discarded (14 males and 14 females were excluded in this manner). Consequently, the speech of 30 males and 30 females chosen from 60 males and 86 females, respectively, were used to train gender specific UBMs. The speech of the remaining 30 males and 56 females was used to create Tnorm speakers models.

B. Evaluation Measure

The evaluation of the speaker verification system is based on detection error tradeoff (DET) curves, which show the tradeoffs between false alarm (FA) and false rejection (FR) errors. We also used the detection cost function (DCF) defined in [21]

$$DCF = C_{\text{miss}} E_{\text{miss}} P_{\text{target}} + C_{\text{fa}} E_{\text{fa}} (1 - P_{\text{target}}) \quad (9)$$

where P_{target} is the *a priori* probability of target tests (i.e., 0.01), the specific cost factors $C_{\text{miss}} = 10$ and $C_{\text{fa}} = 1$, so the point of interest is shifted towards low FA rates. We also calculated an equal error rate (EER) as a more intuitive measure of system performance.

C. Experimental Setup

A 30-ms sliding Hamming window with a 15-ms skip rate was used to obtain a sequence of frames for which speech feature vectors were estimated. The feature vector consisted of 12-dimensional MFCCs concatenated with 12Δ-MFCC and $12\Delta\Delta$-MFCC. Thus, a 36-dimensional feature vector was formed. The 0th cepstral coefficient was excluded.

It is common to postprocess the MFCCs to compensate for channel-related undesirable effects. Based on results given in Table I of Burget, *et al.* [23], where performance improvement could be attained by applying different types of channel compensation methods were studied, we choose RASTA filtering for feature postprocessing because of its major contribution to performance improvement. This choice is also made in [24], where intersession variability is of interest. Cepstral mean and variance normalization was performed afterwards. Following the system setup described in [25], we trained gender-dependent UBMs and investigated the effect of the Tnorm score normalization [22] on system performance.

For each target speaker, a speaker-specific GMM with diagonal covariance matrices was trained via MAP adaptation of the Gaussian means of the UBM. The relevance factor, which is a parameter for determining the impact of new data in model adaptation from the UBM, was set to 16. (In this study the UBM model order was 1024.) The UBM was trained with a minimum of 100 iterations of the EM algorithm. Variance flooring of 0.01 was imposed to avoid singularity.

For each verification test, i.e., a pair of a test segment and a target speaker, the test segment is scored against both the target model and the background model matching the target gender, ignoring low energy frames (silence removal was performed by applying a bi-Gaussian modeling of frames energy and discarding frames with low mean values). Search width of the SGMM algorithm was varied from 8 to 512.

D. Experiments and Results

We compared the performance of the PSO-optimized SGMM using s'_t against the nonoptimized SGMM that uses the basic sorting function s_t. The GMM-UBM system was used as a baseline in both cases. The results are summarized in Figs. 5 and 6 with DET plots for the SGMMs, where the search width was selected as powers of 2. In Fig. 7, we present the speed up factor for each value of M_s. The effect of Tnorm score normalization can be seen in Figs. 5 and 6 which include the results both with (left) and without (right) Tnorm score normalization.

As expected, the speaker verification performance degradation is small when the search width values M_s are close to the model order. Performance degradation increases as M_s approaches C. However, the acceleration rate becomes larger in this region. As shown in the DET plots in Figs. 5 and 6, the performance of the SGMM algorithm improves substantially by using optimal weights. In addition, applying Tnorm degrades system performance from an EER perspective but enhances it if we focus our regard on the low false alarm region which is designated by "min. DCF." From the DET plots for the optimized

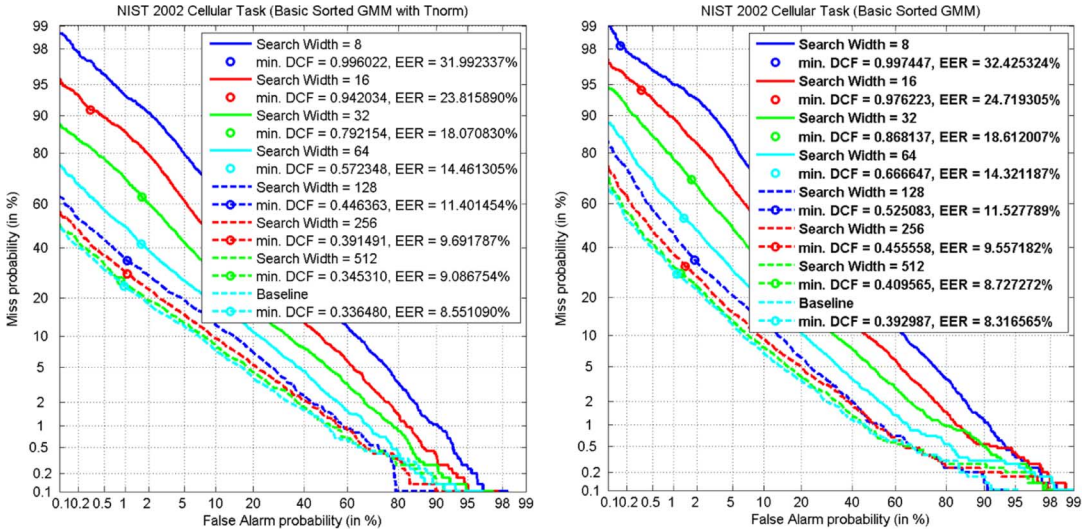

Fig. 5. Performance comparison of the baseline versus basic SGMM for multiple values of M_s.

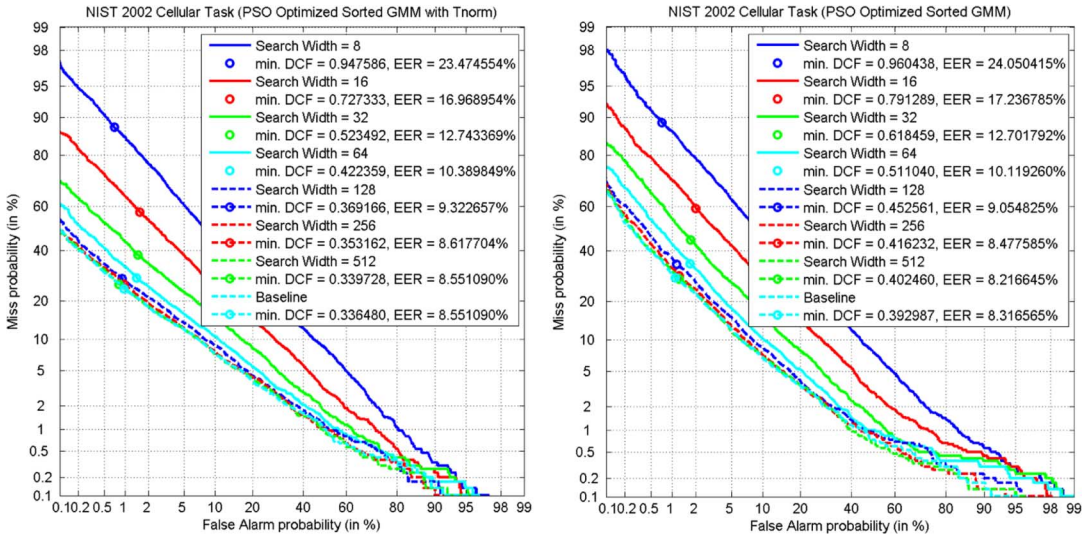

Fig. 6. Performance comparison of the baseline GMM-UBM versus PSO optimized SGMM for multiple values of M_s.

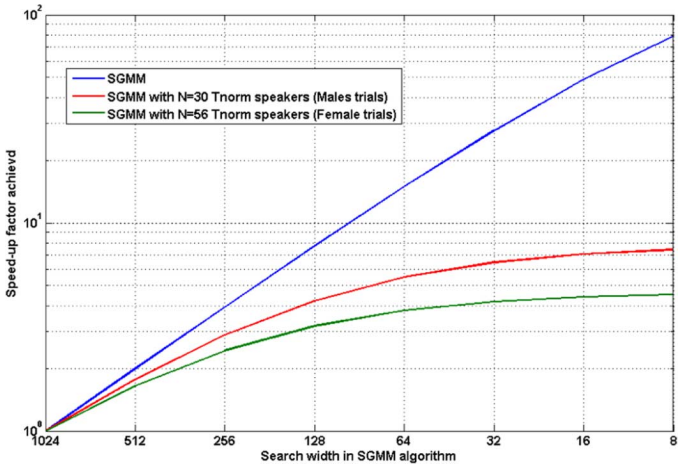

Fig. 7. Speed-up factor achieved in the SGMM versus the search width.

SGMM, we can see that $M_s = 128$ is the knee of performance degradation for SGMM. Considering the baseline system performance, it is noticeable that the main effect of using optimized weights in SGMM is to partially compensate the system degradation (performance degradation in EER from 8.32% to 9.06% and in min. DCF from 0.393 to 0.453 in PSO-based SGMM compared with performance degradation in EER from 8.32% to 11.53% and in min. DCF from 0.393 to 0.526 in basic SGMM). It provides a speed-up factor of 7.74, as we see from the knee of the performance plot. Also, it can be stated that the knee of performance changed to $M_s = 64$ by using optimized weights in SGMM since it has a minor degradation in performance compared to that of the baseline system (degradation in EER from 8.32% to 10.12% and in min. DCF from 0.393 to 0.511) while reaching a speed-up factor of 14.9. Although adding Tnorm to PSO optimized SGMM enhances system performance to some extent, when compared to the loss in speed-up factor (for example, from 14.9 to 4.29 for $M_s = 64$) this is insignificant. To clarify the performance improvement attained using optimized weights in SGMM, detailed results of all experiments are reported in Table I, partitioned to male and female trials. There are some interesting observations in this table. First, baseline system performance is better for males than females. Second,

TABLE I
SPEAKER RECOGNITION PERFORMANCE FOR SGMM DIFFERENT SYSTEMS (MALES)

(MALES)

Evaluation Point	EER				Min. DCF			
	With Tnorm		Without Tnorm		With Tnorm		Without Tnorm	
System Type	Basic sort	PSO sort	Basic sort	PSO sort	Basic sort	PSO sort	Basic sort	PSO sort
Baseline	8.60	8.60	8.27	8.27	0.33	0.33	0.36	0.36
Ms = 512	9.01	8.60	8.52	8.26	0.34	0.34	0.38	0.37
Ms = 256	9.57	8.84	9.41	8.27	0.38	0.35	0.44	0.38
Ms = 128	11.52	9.34	11.20	8.94	0.41	0.36	0.48	0.43
Ms = 64	13.31	9.82	12.97	9.90	0.50	0.41	0.57	0.48
Ms = 32	16.23	11.59	16.47	11.68	0.67	0.52	0.74	0.60
Ms = 16	20.86	15.83	21.10	16.63	0.82	0.69	0.87	0.74
Ms = 8	27.84	23.13	28.33	23.68	0.99	0.89	0.99	0.91

(FEMALES)

Evaluation Point	EER				Min. DCF			
	With Tnorm		Without Tnorm		With Tnorm		Without Tnorm	
System Type	Basic sort	PSO sort	Basic sort	PSO sort	Basic sort	PSO sort	Basic sort	PSO sort
Baseline	8.51	8.51	8.34	8.34	0.32	0.32	0.40	0.40
Ms = 512	9.09	8.40	8.91	8.17	0.34	0.33	0.41	0.41
Ms = 256	9.77	8.50	9.59	8.62	0.38	0.35	0.45	0.43
Ms = 128	11.53	9.25	11.59	9.14	0.46	0.36	0.54	0.46
Ms = 64	15.42	10.74	15.25	10.28	0.61	0.41	0.71	0.51
Ms = 32	18.96	13.60	19.93	13.37	0.86	0.52	0.90	0.62
Ms = 16	25.93	17.59	26.57	17.77	0.96	0.73	0.97	0.82
Ms = 8	34.45	23.83	34.97	24.34	0.99	0.97	0.99	0.97

SGMM system performance in the case for the male speakers is less sensitive to speed-up factor increase than that of the one for females. This is shown in Table I with shadowed cells, where $M_s = 128$ is considered as the knee of performance for basic SGMM, and $M_s = 64$ is the knee for PSO optimized SGMM. Clearly, when compared to the corresponding baseline, performance degradation due to SGMM in males trials is far less than in females trials.

E. Comparison With Other Accelerating Techniques

In hash GMM [7], a smaller UBM (named hash UBM) is created initially, based on those training data used for main UBM training. Then, hash UBM is used to address the mixtures of the main UBM. Each mixture of hash UBM has a short list of mixtures of main UBM that occurred with more frequency during the simultaneous evaluation of the main UBM and hash on the training database. In hash GMM, it was found a speed-up factor of 16 to be a compromise between the reduction of computational demands and an increase in verification error. However, hash GMM needs nearly 4% memory overhead compared to the baseline system in order to store the hash model, while SGMM has merely 0.1% memory overhead to store sorting function parameters. In [8], a VQ preclassifier is used in a GMM-based speaker identification. A VQ codebook is constructed by pooling all the different speakers' GMM mean vectors together and running the K-means. Each VQ partition is associated with a shortlist of Gaussians to be evaluated. The shortlists contain Gaussians from *all* models. In the testing phase, VQ codebook is searched for the nearest neighbor for each test vector, and the Gaussians associated with this partition are scored from different models. All test vectors are processed, and the N-top scoring models are retained for final scoring. It was reported that this method provides a speed-up factor of 4 compared to the baseline with some degradation in performance. Structural GMM [9], [10] was reported to have the ability of reaching a speed-up factor of 17 while improving the system performance when combining with a multilayer perceptron neural network. This superior performance was achieved by modeling feature space in different resolutions and constructing a tree on them, and then finally constructing appropriate structural GMMs for speakers with multilevel MAP adaptation. Algorithm complexity and memory overhead are not comparable with SGMM, so that consideration of neural network computations may be excluded insofar as speed-up factors are concerned.

In [12], the authors examined multiple pruning methods for speaker identification based on GMM and vector quantization techniques. Pruning methods were used to improve efficiency in the test score normalization stage resulting in fast cohort selection. As authors claimed, a speed-up factor of 23 could be reached, with improved performance. Techniques presented in this work could be effectively cascaded with SGMM.

In [13], a new framework was introduced for constructing a speaker recognition system based on GMMs that used Gaussian pruning based on approximated cross entropy (ACE). ACE is reported to have a speed-up factor of 3 with a statistically insignificant degradation in accuracy. Furthermore, by using two-phase scoring they reported a speed-up factor of 5 with minor degradation in performance. In discriminative mixture selection [14], the most discriminative of all UBM mixtures are selected for use by identifying ambiguous regions within the mixtures and eliminating them from consideration by a minimum classification error (MCE) training algorithm. This technique was applied to a dialect classification task where it was shown that by selecting only 65% of mixtures the system performance could be improved.

V. CONCLUSION

We presented an efficient PSO-based scheme to enhance the performance of SGMMs by means of fine-tuning the sorting function parameters. A comparative evaluation with the baseline approach on the one-speaker detection task, as defined in the NIST 2002 experimental protocol, confirmed the superior performance of the proposed approach. In this paper, only PSO optimization for the proposed sorting function was considered. Finding a more relevant fitness function or using a type of discriminative training may also improve the results and achieve superior performance.

REFERENCES

[1] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using Gaussian mixture speaker models," *IEEE Trans. Speech Audio Process.*, vol. 3, no. 1, pp. 72–83, Jan. 1995.
[2] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Process.*, vol. 10, no. 1–3, pp. 19–41, Jan. 2000.

[3] A. Chan, R. Mosur, A. Rudnicky, and J. Sherwani, "Four-layer categorization scheme of fast GMM computation techniques in large vocabulary continuous speech recognition systems," in *Proc. Interspeech'04*, 2004, pp. 689–692.
[4] H. R. S. Mohammadi and R. Saeidi, "Efficient implementation of GMM based speaker verification using sorted Gaussian mixture model," in *Proc. EUSIPCO'06*, Florence, Italy, Sep. 4–8, 2006, CD-ROM.
[5] R. Saeidi, H. R. S. Mohammadi, R. D. Rodman, and T. Kinnunen, "A new segmentation algorithm combined with transient frames power for text independent speaker verification," in *Proc. ICASSP'07*, Apr. 2007, vol. 1, pp. 305–308.
[6] H. R. S. Mohammadi, R. Saeidi, M. R. Rohani, and R. D. Rodman, "Combined inter-frame and intra-frame fast scoring methods for efficient implementation of GMM-based speaker verification systems," in *Proc. ICASSP'07*, Apr. 2007, pp. 309–312.
[7] R. Auckenthaler and J. Mason, "Gaussian selection applied to text-independent speaker verification," in *Proc. "A Speaker Odyssey," Speaker Recognition Workshop*, 2001, pp. 83–86.
[8] M. Roch, "Gaussian-selection-based non-optimal search for speaker identification," *Speech Commun.*, vol. 48, pp. 85–95, 2006.
[9] B. Xiang and T. Berger, "Efficient text-independent speaker verification with structural Gaussian mixture models and neural networks," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 5, pp. 447–456, Sep. 2003.
[10] Z. Xiong, T. F. Zheng, Z. Song, F. Soong, and W. Wu, "A tree-based kernel selection approach to efficient Gaussian mixture model-universal background model based speaker identification," *Speech Commun.*, vol. 48, pp. 1273–1282, 2006.
[11] B. L. Pellom and J. H. L. Hansen, "An efficient scoring algorithm for Gaussian mixture model based speaker identification," *IEEE Signal Process. Lett.*, vol. 5, no. 11, pp. 281–284, Nov. 1998.
[12] T. Kinnunen, E. Karpov, and P. Fränti, "Real-time speaker identification and verification," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 1, pp. 277–288, Jan. 2006.
[13] H. Aronowitz and D. Burshtein, "Efficient speaker recognition using approximated cross entropy (ACE)," *IEEE Trans. Audio, Speech and Language Processing, Special Issue on Speaker and Language Recognition*, vol. 15, no. 7, pp. 2033–2043, 2007.
[14] R. Huang and J. H. L. Hansen, "Unsupervised discriminative training with application to dialect classification," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 8, pp. 2444–2453, Nov. 2007.
[15] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, vol. IV, pp. 1942–1948.
[16] M. Clerc, "The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, 1999, vol. 3, pp. 1951–1957.
[17] P. Rose, *Forensic Speaker Identification*, ser. Forensic Science Series. New York: Taylor & Francis, 2002.
[18] J.-L. Gauvain and C.-H. Lee, "Maximum *a posteriori* estimation for multivariate Gaussian mixture observations of Markov chains," *IEEE Trans. Speech Audio Process.*, vol. 2, no. 2, pp. 291–298, Apr. 1994.
[19] A. Abraham, H. Guo, and H. Liu, "Swarm intelligence: Foundations, perspectives and applications, swarm intelligent systems," in *Studies in Computational Intelligence*. Berlin, Germany: Springer Verlag, 2006, pp. 3–25.
[20] H. R. S. Mohammadi and W. H. Holmes, "Low cost vector quantization methods for spectral coding in low rate speech coder," in *Proc. ICASSP'95*, Detroit, MI, 1995, vol. 1, pp. 720–723.
[21] "The NIST Year 2002 Speaker Recognition Evaluation." [Online]. Available: http://www.nist.gov/speech/tests/
[22] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas, "Score normalization for text-independent speaker verification systems," *Digital Signal Process.*, vol. 10, pp. 42–54, 2000.
[23] L. Burget, P. Matejka, P. Schwarz, O. Glembek, and J. Cernocky, "Analysis of feature extraction and channel compensation in a GMM speaker recognition system," *Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 7, pp. 1987–1998, Sep. 2007.
[24] D. Garcia-Romero and C. Y. Espy-Wilson, "Intersession variability in speaker recognition: A behind the scene analysis," in *Proc. Interspeech'08*, Melbourne, Australia, 2008, pp. 1413–1416.
[25] C. Barras and J.-L. Gauvain, "Feature and score normalization for speaker verification of cellular data," in *Proc. ICASSP*, May 2003, vol. 2, pp. 49–52.

Rahim Saeidi received the B.S. degree in electrical engineering from Azad University—Save branch, Saveh, Iran, in 2002, and the M.Sc. degree in telecommunication systems engineering from the Iran University of Science and Technology, Tehran, Iran, in 2005.

He joined the Iranian Research Institute for Electrical Engineering, Jahade Daneshghahi, in 2006 where he is currently Research Assistant involved in digital signal processing for speech applications. His research interests include speech processing, machine learning, neuroscience and pattern recognition.

Hamid Reza Sadegh Mohammadi (M'96) was born in Tehran, Iran, in 1960. He received the B.Sc. degree in electrical engineering in 1984, the M.Sc. degree in electronics in 1988 both from the Iran University of Science and Technology, Tehran, and the Ph.D. degree in electrical engineering from the University of New South Wales, Sydney, Australia, in 1997.

He has been with the Academic Center for Education, Culture, and Research (ACECR), Tehran, Iran, since 1982. Currently, he is with the Iranian Research Institute for Electrical Engineering (IRIEE) as an Assistant Professor and Head of the Institute. He has published more than 25 scientific papers. He is the founding editor of the *Iranian Journal of Electrical and Computer Engineering* (IJECE). His research interests include speech and signal processing.

Todor Ganchev (S'96–A'99–M'02) received the Diploma Engineer degree in electrical engineering from the Technical University of Varna, Varna, Bulgaria, in 1993 and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Patras, Patras, Greece, in 2005.

From February 1994 to August 2000, he consequently held Engineering, Research, and Teaching Staff positions at the Technical University of Varna. Since September 2000, he has been a Research Fellow at the Wire Communications Laboratory, University of Patras. Currently, he holds a Senior Researcher position at the Wire Communications Laboratory. His research interests are in the areas of pattern recognition, signal processing, and applications.

Robert David Rodman received the B.A. degree in mathematics in 1961, the M.A. degree in mathematics in 1965, the M.A. degree in linguistics in 1971, and the Ph.D. degree in linguistics in 1973, all from the University of California, Los Angeles.

He has been on the faculties of the University of California at Santa Cruz, the University of North Carolina at Chapel Hill, Kyoto Industrial College, Kyoto, Japan, and North Carolina State University, Raleigh, where he is currently a Professor of computer science. He is also a consulting forensic linguist. His research areas are forensic linguistics and computer speech processing.