Gaussian Mixture Modeling for Dynamic Particle Swarm Optimization of Recurrent Problems

Eduardo Vellasques
École de Technologie Supérieure
Université du Québec
1100 rue Notre-Dame Ouest
Montreal, Canada
evellasques@livia.etsmtl.ca

Robert Sabourin
École de Technologie Supérieure
Université du Québec
1100 rue Notre-Dame Ouest
Montreal, Canada
robert.sabourin@etsmtl.ca

Eric Granger
École de Technologie Supérieure
Université du Québec
1100 rue Notre-Dame Ouest
Montreal, Canada
eric.granger@etsmtl.ca

ABSTRACT

In dynamic optimization problems, the optima location and fitness value change over time. Techniques in literature for dynamic optimization involve tracking one or more peaks moving in a sequential manner through the parameter space. However, many practical applications in, e.g., video and image processing involve optimizing a stream of recurrent problems, subject to noise. In such cases, rather than tracking one or more moving peaks, the focus is on managing a memory of solutions along with information allowing to associate these solutions with their respective problem instances. In this paper, Gaussian Mixture Modeling (GMM) of Dynamic Particle Swarm Optimization (DPSO) solutions is proposed for fast optimization of streams of recurrent problems. In order to avoid costly re-optimizations over time, a compact density representation of previously-found DPSO solutions is created through mixture modeling in the optimization space, and stored in memory. For proof of concept simulation, the proposed hybrid GMM-DPSO technique is employed to optimize embedding parameters of a bi-tonal watermarking system on a heterogeneous database of document images. Results indicate that the computational burden of this watermarking problem is reduced by up to 90.4% with negligible impact on accuracy.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence—*Problem Solving, Control Methods, and Search*

General Terms

Experimentation

GECCO'12, July 7–11, 2012, Philadelphia, Pennsylvania, USA.

Keywords

Particle Swarm Optimization, Dynamic Optimization, Gaussian Mixture Models, Estimation of Distribution Algorithms

1. INTRODUCTION

In evolutionary computing (EC), populations of candidate solutions are evolved through a certain number of generations, driven by one or more fitness functions. Since EC allows the inherent optimization problem to be seen as a black box, it has led not only to an increased interest by researchers, but to a considerable increase in its practical applications. However, differently than the idealized static synthetic problems commonly seen in the literature, most practical applications have a dynamic nature.

In a dynamic optimization problem (DOP), the optimum (or optima) change with time. A change might be followed by a period of stasis [7]. As observed by Nickabadi et al [16], a change can be either of type I (optimum location changes with time), type II (optimum fitness changes with time but location remains fixed) or type III (both, the location and fitness change with time). A change in such context is subject to *temporal severity* and/or *spatial severity.* Some applications involve tracking one or more peaks moving in a sequential manner through the environment. In such scenario, changes can be of any of the three types above but the temporal severity is usually high (change occurs in short intervals of time) while spatial severity is usually low (peaks move in a smooth fashion, therefore stasis is small or inexistent).

Control applications usually involve such type of change. For example, moving a robot through a 3D space based on sensor data usually involves tracking small variations on the signals occurring in short intervals). In such cases, the most important issues to be addressed are diversity loss (in static optimization, the population tends to collapse towards a narrow region of the fitness landscape, making adaptation difficult) and outdated memory (past knowledge is very useful in EC, but it might be useless when a change occurs). Diversity loss can be mitigated by three different approaches – introducing diversity after a change occurs, maintaining diversity throughout the run or using multi-population – while outdated memory can be tackled by either erasing memory or re-evaluating memory and setting it to either previous or current value (which one is better) [4]. Another important issue for such type of DOP is detecting when a change

occurs. The most common approach is to track the fitness value of one or more sentry particles [5, 10]. An alternative is to compute a running average of the fitness function for the best individuals over a certain number of iterations [23].

However, in some other applications (those that involve cyclic changes), the spatial severity is high (peaks "jump" from their previous locations) but the temporal severity is low. Moreover after a certain number of instances, a problem instance seen before can re-appear (in some cases with small distortions). This type of DOP is known as a recurrent (or cyclic) problem [1, 25, 13]. Such type of DOP is usually relied to applications involving optimizing system parameters for streams of data like machine learning [11] or digital watermarking [22].

In [22], it was observed that computational burden can be decreased for DOP involving recurrent problems with the use of a memory of selected solutions. Costly re-optimization operations can be avoided in some cases by properly matching new problems with their corresponding memory solutions and employing these solutions directly. The main reason is that a recurrent problem is a special case of a type III change where the spatial severity is minimal (or inexistent) in the parameter space and small in the fitness space. Moreover, time severity is small or inexistent – the exact moment a new problem instance arrives is known beforehand, e.g. the moment a new image is picked from a stream – that is, stasis can have an indefinite length. Such case of type III change can be considered as a pseudo-type II change. Therefore, even though an optimal solution obtained for a similar previous problem might not be exactly optimal for the new problem (they are near-optimal), there are several practical applications where the computational burden is an important constraint and an insignificant loss in precision is to be preferred as long as it results in a significant decrease in computational burden. Due to its fast convergence, Particle Swarm Optimization (PSO) is preferred in such time-constrained applications.

In this paper, a novel memory-based Dynamic PSO (DPSO) technique is proposed for fast optimization of recurrent dynamic problems. In this approach, a two-level memory of selected solutions and Gaussian Mixture Models (GMM) of their corresponding environments is incrementally built. For each new problem instance, solutions are sampled from this memory and re-evaluated. A statistical test compares the distribution of both fitness values and the best re-evaluated solutions is employed directly if both distributions are considered similar. Otherwise, DPSO [11] is employed in order to optimize parameters. The main research problem addressed in the given work is how to build a comprehensive model of the stream of optimization problems using representative instances of these problems. The research hypothesis is that density estimates of historical solutions found during the optimization phase provide a compact yet flexible representation of a given optimization problem and for this reason, are more suitable for high data rate applications where change in the streams of problems might require a fast adaptation of the given memory. Such flexibility is expected because a density estimate allows for more dynamic update operators than would be possible with a memory of static solutions. Moreover, sentry solutions sampled from this density estimate should provide a more general coverage of a given landscape than selected isolated solutions and are thus more appropriate for matching two problem instances.

A review of DPSO is provided in Section 2. The proposed hybrid GMM-DPSO technique for fast dynamic optimization of long streams of recurrent problems is proposed in Section 3. Section 4 provides proof of concept simulation results and discussions.

2. DYNAMIC PSO

PSO [12] is an optimization heuristics based on the concept of swarm intelligence. In its canonical form, a population (swarm) of candidate solutions (particles) is evolved through a certain number of generations. As its physics equivalent, each particle i in PSO has a position ($\mathbf{x}_i$) in a multidimensional search space and a velocity ($\mathbf{v}_i$). The velocity of the i^{th} particle is adjusted at each generation according to the best location visited by that particle ($\mathbf{p}_i$) and the best location visited by all neighbors of particle i ($\mathbf{p}_g$):

$$\mathbf{v}_i = \chi \times (\mathbf{v}_i + c_1 \times r_1 \times (\mathbf{p}_i - \mathbf{x}_i) + c_2 \times r_2 \times (\mathbf{p}_g - \mathbf{x}_i)) \quad (1)$$

where χ is a constriction factor, chosen to ensure convergence [2], c_1 and c_2 are respectively the cognitive and social acceleration constants (they determine the magnitude of the random forces in the direction of $\mathbf{p}_i$ and $\mathbf{p}_g$ [19]), r_1 and r_2 are two different random numbers in the interval $[0, 1]$. PSO parameters c_1 and c_2 are set to 2.05 while χ is set to 0.7298 as it has been demonstrated theoretically that these values guarantee convergence [19]. The neighborhood of a particle can be restricted to a limited number of particles (L-Best topology) or the whole swarm (G-Best topology).

After that, the velocity is employed in order to update the position of that same particle:

$$\mathbf{x}_i = \mathbf{x}_i + \mathbf{v}_i \quad (2)$$

Issues like outdated memory, lack of change detection mechanism and diversity loss limit the use of the canonical PSO to static problems [3, 5]. It is important to mention that outdated memory can be easily tackled by re-evaluating the fitness function for the new problem instance [4, 10] while change detection can be tackled by the use fixed sentry particles, re-evaluated at each generation [5]. However, tackling diversity loss requires more elaborate techniques, which can be categorized as [24]: increasing diversity after a change, maintaining diversity throughout the run, memory-based schemes and multi-population approach.

The choice of diversity enhancement strategy is tied to properties of the dynamic optimization problem. Problems involving a single optimum drifting in the fitness landscape can be tackled by either increasing diversity after a change (e.g. re-initializing part of the swarm, a technique known as random immigrants [23]) or by maintaining diversity throughout the run. Problems involving multiple peaks drifting in the fitness landscape with the optimal position shifting between these peaks can be tackled with the use of multi-population and is for example, the assumption of the technique proposed by Parrot and Li [17].

Problems involving one or more states re-appearing over time are better tackled through the use of memory-based schemes [25]. The main reason is that in such type of DOP, the transition between one or more problem instances is not smooth as assumed in [17] and the three strategies described above are of no use when the optimum moves away from the area surveyed by the swarm. As observed by Yang and Yao [25], the main strategy to tackle such type

of DOP is to preserve relevant solutions in a memory either by an implicit or an explicit memory mechanism and then, recall such solutions for similar future problems. In an implicit memory mechanism, redundant genotype representation (i.e. diploidy-based GA) is employed in order to preserve knowledge about the environment for future similar problems. In an explicit mechanism, precise representation of solutions is employed but an extra storage space is necessary to preserve these solutions for future similar problems. The three major concerns in memory-based optimization systems are: (1) what to store in the memory?; (2) how to organize and update the memory?; (3) how to retrieve solutions from the memory?

In this paper we propose a technique which is based on storing Gaussian Mixture Model of solutions in the optimization space along with their respective global best solutions. The use of probabilistic models in EC is not new. Such approach, known as Estimation of Distribution Algorithms (EDA) [18] is an active research topic. The main advantage of EDA is that such probabilistic models are more effective in preserving historical data than a few isolated high evaluating solutions.

3. FAST OPTIMIZATION OF RECURRENT PROBLEMS USING GAUSSIAN MODELING OF DPSO POPULATIONS

Figure 1 illustrates the proposed memory-based method. In the proposed approach, the basic memory unit is a **probe** and it contains a density estimate of solutions plus the global best solution, obtained after the optimization of a given problem instance. The first memory level is the Short Term Memory (STM) which contains a single probe, obtained during the optimization of a single problem instance and provides fast recall for situations where a block of similar problem instances appear sequentially (e.g. a sequence of practically identical frames in a video sequence, except for noise). The second memory level is the Long Term Memory (LTM) which contains multiple probes obtained in the optimization of different problem instances and allows recalling solutions for problems reappearing in an unpredictable manner.

Given a stream of optimization problems, an attempt to recall the STM is performed first. During a recall, solutions are re-sampled from the density estimate and re-evaluated (along with the global best solution) in the new problem instance. The Kolmogorov-Smirnov statistical test is employed in order to measure the difference between the sampled and re-evaluated sets of fitness values. If they are similar, this means that the given problem instance is a recurrent one and therefore, the best re-evaluated solution is employed right away, avoiding a costly re-optimization operation. If the distributions are not similar, the same process (sampling/re-evaluation/statistical test) is repeated for each probe in the LTM until either a similarity between both distributions of fitness values is found or all probes have been tested.

In such case, the solutions sampled from the STM probe are used as a starting point for a new round of optimization. The L-best DPSO algorithm described in [11] is used for this purpose. The change detection module was adapted to the memory recall mechanism employed in the proposed approach. Although each problem instance is static, this DPSO variant allows tackling multi-modal optimization problems and the motivation behind the proposed technique is tackling dynamic optimization of recurrent problems in practical applications (which might imply in multi-modal landscapes). After that, a mixture model of the fitness landscape is estimated using the GMM approach of Figueiredo and Jain [8]. The **position** and **fitness** data of all intermediary solutions, found in all generations are employed for this purpose. The reason for using all intermediary solutions and not selected solutions (e.g. best particle positions) is that these solutions allow a more general model of the fitness landscape. That is, local best data usually results in density estimates that are over-fit to a specific problem.

This mixture model along with the global best solution will form a probe, to be stored in the STM (replacing the previous probe) and updated into the LTM. The LTM update consists of either a merge between the new probe and a probe in the memory (if they are similar) or an insert (if either the LTM is empty or no similar probe has been found). In such case, if the memory limit has been reached, an older probe is deleted (we propose deleting the probe which resulted in the smallest number of successful recalls up to that instant).

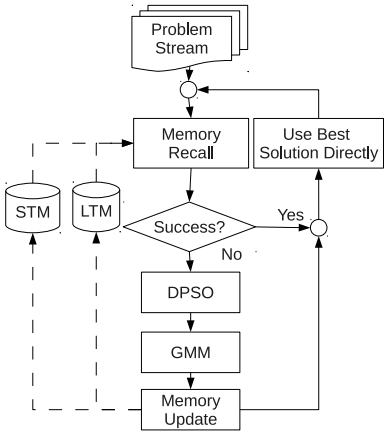


Figure 1: Diagram of the proposed method.

3.1 Gaussian mixture modeling of fitness landscapes

The main reason for building and storing a model of all solutions found during the optimization of a problem instance rather than storing selected solutions is that the former allows a more compact and precise representation of the fitness landscape than individual solutions. As explained before, GMM [8] is employed for this purpose since it is a powerful tool for modeling multi-modal data (which makes the proposed technique robust for multi-modal optimization as well). Different than other clustering techniques such as k-means, GMM allows to model overlap among data densities, and provides more accurate (complex) modeling of data distributions. A mixture model is a linear combination of a finite number of models

$$p(\boldsymbol{x}|\Theta) = \sum_{j=1}^{K} \alpha_j p(\boldsymbol{x}|\theta_j) \quad (3)$$

where $p(\boldsymbol{x}|\Theta)$ is the probability density function (pdf) of a continuous random vector $\boldsymbol{x}$ given a mixture model Θ,

K is the number of mixtures, α_j and θ_j are the mixing weights and parameters of the j^{th} model (with $0 < \alpha_j \leq 1$ and $\sum_{j=1}^{K} \alpha_j = 1$). The mixture model parameters $\Theta = \{(\alpha_1, \theta_1), ..., (\alpha_K, \theta_K)\}$ are estimated using the particle position ($\boldsymbol{x}$) and fitness ($f(\boldsymbol{x})$) of all particles through all generations.

The use of density models in EC is not new. However, such strategy of using both phenotypic and genotypic data to estimate the models is novel. Another major difference between the proposed use of probabilistic models and those seen in the EDA literature is that in the proposed approach, all solutions found in the course of an optimization task are employed in order to estimate the model of the fitness landscape. In the EDA literature instead, selected (best fit) solutions at each generation are employed in model estimation, as a selection strategy. In the proposed approach, a density estimate is employed as a mean of matching new problems with previously seen problems.

3.2 Memory update

The memory is due to be updated after the mixture model has been created. As mentioned before, the basic memory element in the proposed approach is a probe. Updating the STM is trivial, it requires basically deleting the current probe and inserting the new probe since the STM should provide means of recalling the last case of optimization, for situations involving a block of similar optimization problems appearing in sequence.

The LTM instead, must provide a general model of the stream of optimization problems. Since all LTM probes must be recalled before optimization is triggered, the size of the LTM must be kept to a minimum in order to avoid a situation where the cost of an unsuccessful recall is greater than the cost of full optimization. For this reason, we propose an adaptive update mechanism. In the proposed mechanism, when the LTM is due to be updated, the $C2$ distance metric [21] (which provides a good balance between computational burden and precision) is employed in order to measure the similarity between the GMM of the new probe and that of each of the probes in the LTM. The new probe is merged with the most similar probe in LTM if this distance is smaller than a given threshold. Otherwise, the new probe is inserted (the probe with smallest number of successful recalls is deleted if the LTM size limit has been reached). The insert threshold is computed based on the mean minimum distance between new probes and probes on the LTM for the last T LTM updates (μ_δ^t). That is, an insert will only occur if $C2 - \mu_\delta^t$ is greater than the standard deviation for the same time-frame (σ_δ^t).

The Hennig [9] technique, which is based on the use of Bhattacharyya distance and does not require the use of historical data, is employed in order to merge two GMMs. The Bhattacharyya distance is defined as:

$$\bar{\boldsymbol{\Sigma}} = \frac{1}{2}(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2) \tag{4}$$

$$d_B(\Theta_1, \Theta_2) = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \bar{\boldsymbol{\Sigma}}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + \frac{1}{2}\log\left(\frac{|\frac{1}{2}(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)|}{\sqrt{|\boldsymbol{\Sigma}_1||\boldsymbol{\Sigma}_2|}}\right) \tag{5}$$

where μ_i is a mean vector and Σ_i is a covariance matrix.

In Hennig's approach, given a tuning constant $d^* < 1$, the two components with maximum Bhattacharyya distance are merged iteratively as long as $e^{-d_B} < d^*$ for at least one component. We propose a slight modification, which is to merge the two components with minimum distance instead, in order to get a more incremental variation in the mixture components.

If the number of mixture components after the merge operation is still greater than a limit, unmerged components from the older mixture are deleted (the old unmerged component with the highest Bhattacharyya distance from all other components is delete iteratively until the limit has been achieved).

Algorithm 1 summarizes the memory update mechanism. After the end of a round of re-optimization, a new mixture (Θ_N) is estimated based on position and fitness values of all particles from all generations during the optimization process (step 1). The estimated mixture plus the global best solution will form a probe to be added to the STM (any previous STM probe is deleted, step 2). Then, if the length of the vector containing the last n minimum $C2$ distances between new probes and probes in the LTM ($\boldsymbol{\delta}$) is smaller than T (step 3), its mean and standard deviation ($\mu_{\boldsymbol{\delta}}^t$ and $\sigma_{\boldsymbol{\delta}}^t$) are initialized based on pre-defined values ($\mu_{\boldsymbol{\delta}}^0$ and $\sigma_{\boldsymbol{\delta}}^0$, steps 4 and 5). Otherwise, $\mu_{\boldsymbol{\delta}}^t$ and $\sigma_{\boldsymbol{\delta}}^t$ are computed based on $\boldsymbol{\delta}$ (steps 7 and 8). After that, the minimum $C2$ distance between new probe and probes in the LTM is added to $\boldsymbol{\delta}$ (steps 10 and 11). The new probe is inserted into the memory if the difference between the minimum $C2$ distance and $\mu_{\boldsymbol{\delta}}^t$ is greater than the standard deviation ($\sigma_{\boldsymbol{\delta}}^t$, steps 12 to 16). It important to notice that before the insert, the LTM probe with smallest number of recalls is deleted if memory limit has been reached. Otherwise the new probe is merged with the most similar probe in the LTM (steps 18 and 19). If the limit of vector $\boldsymbol{\delta}$ has been reached, its first element is deleted (steps 21 to 23).

3.3 Memory recall

Basically, the recall mechanism is the same for both levels of memory. The only difference is that the LTM contains more probes than the STM and for this reason, this process might be repeated for many LTM probes until either all probes have been tested of a successful recall has occurred. For a given probe, N_s solutions are sampled from its mixture model:

$$\boldsymbol{X}_s = \boldsymbol{\mu}_j + \boldsymbol{\Lambda}_j^{\frac{1}{2}} \boldsymbol{U}_j \boldsymbol{R}_s \tag{6}$$

where $\boldsymbol{X}_s$ is a sampled solution, s is the index of a solution sampled for the component j in the mixture ($\lfloor (N_s \alpha_j) + 0.5 \rfloor$ solutions are sampled per component), $\boldsymbol{\Lambda}_j$ and $\boldsymbol{U}_j$ are the eigen-decomposition of $\boldsymbol{\Sigma}_j$ ($\boldsymbol{\Sigma}_j = \boldsymbol{U}_j \boldsymbol{\Lambda}_j \boldsymbol{U}_j^{-1}$) and $\boldsymbol{R}_s$ is a vector with the same length as $\boldsymbol{\mu}_j$ whose elements are sampled from a normal distribution $N(0, \mathbf{I})$, being $\mathbf{I}$ the identity matrix.

It is important to observe that both, position and fitness values are sampled simultaneously. Then, the sampled solutions (along with the corresponding global best) are reevaluated for the new problem instance. A Kolmogorov- Smirnov statistical test is employed in order to compare the sampled and re-evaluated fitness values. If they are below a critical value for a given confidence level, the recall is considered to be successful and the best recalled solution is employed right away, avoiding a costly re-optimization. If no probe (neither in the STM nor in the LTM) results in a successful recall, a

Algorithm 1 Memory update mechanism.

Inputs:
k_{max} – maximum number of components with $\alpha_j > 0$.
$\mathfrak{M}_S$ – Short Term Memory.
$\mathfrak{M} = \{\mathfrak{M}_1, ..., \mathfrak{M}_{|\mathfrak{M}|}\}$ – Long Term Memory.
$\mathfrak{D}$ – optimization history (set of all particle positions and fitness values for new problem instance).
$L_{\mathfrak{M}}$ – maximum number of probes in LTM.
$\boldsymbol{\delta}$ – last T minimum $C2$ distances between a new probe and probes in the LTM.
$|\boldsymbol{\delta}|$ – number of elements in $\boldsymbol{\delta}$.
T – maximum size of $\boldsymbol{\delta}$.
$\mu^0_{\boldsymbol{\delta}}$, $\sigma^0_{\boldsymbol{\delta}}$ – initial mean and standard deviation of $\boldsymbol{\delta}$.
Output:
Updated memory.

Estimate Θ_N using $\mathfrak{D}$ [8].
Add Θ_N and $\boldsymbol{p}_g$ to $\mathfrak{M}_S$.
if $|\boldsymbol{\delta}| < T$ **then**
  $\mu^t_{\boldsymbol{\delta}} \leftarrow \mu^0_{\boldsymbol{\delta}}$
  $\sigma^t_{\boldsymbol{\delta}} \leftarrow \sigma^0_{\boldsymbol{\delta}}$
else
  $\mu^t_{\boldsymbol{\delta}} \leftarrow \frac{1}{|\boldsymbol{\delta}|}\sum_{i=1}^{|\boldsymbol{\delta}|}\delta_i$
  $\sigma^t_{\boldsymbol{\delta}} \leftarrow \sqrt{\frac{\sum_{i=1}^{n}(\delta_i - \mu^t_{\boldsymbol{\delta}})^2}{|\boldsymbol{\delta}|}}$
end if
$i^* \leftarrow argmin_i\{C2(\Theta_N, \Theta_i)\},\ \forall_{\Theta_i} \in \mathfrak{M}$
$\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} \cup C2(\Theta_N, \Theta_{i^*})$
if $C2(\Theta_N, \Theta_{i^*}) - \mu^t_{\boldsymbol{\delta}} > \sigma^t_{\boldsymbol{\delta}}$ **then**
  if $|\mathfrak{M}| = L_{\mathfrak{M}}$ **then**
    Remove LTM probe with smallest number of successful recalls.
  end if
  Add Θ_N and $\boldsymbol{p}_g$ to $\mathfrak{M}$
else
  $Merge(\Theta_{i^*}, \Theta_N)$ (section 3.2)
  Purge merged mixture in case number of elements exceed k_{max}.
end if
if $|\boldsymbol{\delta}| > T$ **then**
  Remove δ_1.
end if

part of STM re-sampled solutions is injected into the swarm and optimization is triggered for that problem instance.

4. EXPERIMENTAL RESULTS

4.1 Application

The proposed fast optimization technique will be validated in the optimization of embedding parameters for a bi-tonal watermarking system [22]. This is an interesting problem because a given watermark needs to be robust against attacks but at the same time result in minimum visual interference in the host image and this trade-off can be manipulated by carefully adjusting heuristic parameters in the watermark embedder. Generally speaking, in this watermarking system, a cover bi-tonal image is partitioned into several blocks, the flippability score of each pixel is computed using a moving window, the pixels are shuffled according to shuffling seed and each bit of the given bit stream is embedded into each block of the cover image through manipulation of the quantized number of black pixels. The quantization step size (Q) determines the robustness of the watermark and since this watermarking technique allows the embedding of multiple watermarks (with different levels of robustness), we will embed two watermarks a fragile one (which can be employed to enforce image integrity) with a fixed value for its quantization step size of ($Q_F = 2$) and robust one (which can be employed to enforce image authenticity) with an adjustable quantization step size $Q_R = Q_F + \Delta_Q$ where Δ_Q is a parameter to be optimized. More details can be found in [22]. Four parameters need to be optimized: the partition block size which is an integer between 1 and the maximum attainable size for the given image as seen in [15]; the size of the window used in the flippability analysis, which can be either 3×3, 5×5, 7×7 or 9×9; the difference between the quantization step size for the robust and fragile watermarks (Δ_Q), which is an even number between 2 and 150; and the index of the shuffling key (we proposed using a pool of 16 possible shuffling keys, thus this index is an integer between 1 and 16).

The fitness function is a combination of the Bit Correct Ratio (BCR) between the embedded and detected watermarks (both, robust and fragile), and Distance Reciprocal Distortion Measure (DRDM) [14], which measures the quality of the watermarked image (more details can be found in [22]). The three fitness values are aggregated using Chebyshev approach [6]:

$$F(\mathbf{x}) = max_{i=1,..,3}\{(1-\omega_1)(\alpha_s DRDM - r_1), (1-\omega_2)(1-BCR_R - r_2), (1-\omega_3)(1-BCR_F - r_3)\} \quad (7)$$

where α_s is the scaling factor of the quality measurement $DRDM$, BCR_R is the robustness measurement of the robust watermark, BCR_F is the robustness measurement of the fragile watermark, ω_i is the weight of the i^{th} objective with $\omega_i = \frac{1}{3}, \forall_i$, r_i is the reference point of objective i.

4.2 Validation protocol

The BancTec logo (Figure 2a), which has 26×36 pixels will be employed as robust watermark and the Université du Québec logo (Figure 2b), which has 36×26 pixels will be employed as fragile watermark.

(a) (b)

Figure 2: Bi-tonal logos used as watermarks. (a) 26×36 BancTec logo. (b) 36×26 Université du Québec logo.

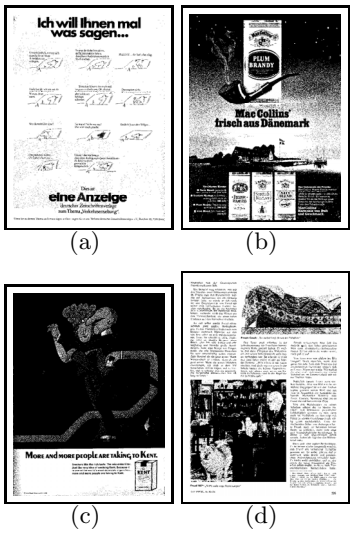
Oulu University's MediaTeam [20] (OULU-1999) database is employed as image stream. This database was scanned at 300 dpi with 24-bit color encoding. Since the baseline watermarking system is bi-tonal, the OULU-1999 database was binarized using the same protocol as in [22]. As some of its images lack the capacity necessary to embed the watermarks described above, a reject rule was applied: all images containing less than 1872 pixels with Structural Neighborhood

Distortion Metric (SNDM) [15] greater than zero were discarded. This rule resulted in the elimination of 15 of the original 512 images. The database was split in two subsets: a smaller one for development purposes, containing 100 images and a larger one, for validation purposes, containing 397 images. Images were assigned to these sets randomly. Table 1 shows the structure of both subsets.

Table 1: OULU-1999 database structure.

Category	Number of Images	
	TRAIN	TEST
Addresslist	0	6
Advertisement	5	19
Article	51	180
Businesscards	1	10
Check	0	3
Color Segmentation	1	7
Correspondence	6	18
Dictionary	1	9
Form	9	14
Line Drawing	0	10
Manual	6	29
Math	4	13
Music	0	4
Newsletter	4	37
Outline	4	13
Phonebook	4	3
Program Listing	2	10
Street Map	0	5
Terrainmap	2	7
Total:	**100**	**397**

As can be seen in Figure 3, the images in this database are considerably heterogeneous.



(a) (b) (c) (d)

Figure 3: Images from OULU-1999-TRAIN. (a) Image 1. (b) Image 2. (c) Image 5. (d) Image 6.

The number of previous updates T employed to compute the adaptive threshold will be set to 10. The initial mean and standard deviation of the minimum distance were set to 361.7 and 172.3 respectively. These values were obtained by running the proposed technique in a "fill-up" mode (forcing re-optimization and LTM insert for every image in the OULU-1999-TRAIN subset). The resulting minimum $C2$ distances of these inserts were employed in order to compute these initial values.

The metrics employed in order to assess the computational performance are the average number of fitness evaluations per image ($AFPI$), the total number of fitness evaluations required to optimize the whole image stream (F_{Evals}) and the decrease in the number of fitness evaluations (DFE), computed as:

$$DFE = 1 - \frac{F_{Evals,M}}{F_{Evals,F}} \tag{8}$$

where $F_{Evals,M}$ is the cumulative number of fitness evaluations for the memory based approach and $F_{Evals,F}$ is the cumulative number of fitness evaluations for full optimization.

The reference points for the Chebyshev Weighted Aggregation were obtained through sensitivity analysis of the OULU-1999-TRAIN subset and were set to $r_1 = r_2 = r_3 = 0.01$. The scaling factor of the DRDM (α_r) was also obtained through sensitivity analysis using the training subset and was to 0.53.

During recall, 19 solutions are re-sampled from each probe, which are re-evaluated along with the global best solution, resulting in 20 sentry particles. The confidence level (α) of the KS statistic was set to the same value employed in [22], which is 0.95 and corresponds to a critical value (D_α) of 0.43. The LTM size is limited to 20 probes. The PSO parameters employed on full optimization are the same defined in [22] (20 particles, neighborhood size of 3, optimization stops if the global best has not improved for 20 generations). Cropping of 1% of the image surface was employed during the optimization since such attack can effectively remove a non-optimized watermark. The proposed approach is compared with a previous approach (case-based), which relies on a memory of selected solutions [22] to demonstrate the main advantages of employing a memory of mixture models.

4.3 Simulation results

The GMM-based approach resulted in a significant decrease in computational burden when compared to full optimization and even compared to the case-based approach (Table 2).

Despite the decrease in computational burden, the watermarking performance of the GMM-based approach is practically the same of the other two approaches (Table 3).

These results demonstrate that the memory of mixture models can cope better with the variations in the stream of optimization problems. Since the case-based probes are more tuned to the problems that generated them, they are more sensitive to small variations in a given recurrent landscape caused by noise. A clear sign of this is that the smaller number of fitness evaluations was obtained even though for two of the watermarking performance metrics there was even an improvement when compared to the case-based approach. Moreover, for both cases, the watermarking performance is similar to that of full optimization, which illustrates the applicability of the proposed technique. However it is important to notice that the basic assumption is that the application can be formulated as the problem of optimizing a stream of optimization problems with some of the problems re-appearing over time, subject to noise.

Table 2: Computational cost of the proposed technique compared to case-based approach and full optimization. $AFPI$ is the average number of fitness evaluations per image where the mean μ and standard deviation σ are presented as $\mu(\sigma)$. F_{Evals} is the cumulative number of fitness evaluations required to optimize the whole stream and DFE is the decrease in the number of fitness evaluations compared to full optimization.

Subset	Full PSO		Case-based			GMM-based		
	$AFPI$	F_{Evals}	$AFPI$	F_{Evals}	DFE	$AFPI$	F_{Evals}	DFE
TRAIN	887 (340)	88740	351 (455)	35100	60.5%	274 (447)	27420	69.1%
TEST	860 (310)	341520	177 (351)	70300	79.4%	83 (213)	32920	90.4%

Table 3: Watermarking performance of the proposed technique compared to case-based approach and full optimization. Here, † is the $DRDM$, ‡ is the BCR robust, § is the BCR fragile. For all values, the mean μ and standard deviation σ per image are presented in the following form: $\mu(\sigma)$. $DRDM$ is presented with two decimal points and BCR is presented in percentage (%) with one decimal point.

Subset	Full PSO			Case-based			GMM-based		
	†	‡	§	†	‡	§	†	‡	§
TRAIN	0.03 (0.03)	98.4 (2.1)	99.7 (0.6)	0.03 (0.03)	97.9 (2.6)	99.6 (1.0)	0.03 (0.03)	97.5 (2.8)	99.4 (1.0)
TEST	0.03 (0.04)	98.4 (2.2)	99.6 (0.6)	0.03 (0.03)	97.2 (3.6)	99 (1.6)	0.03 (0.03)	96.7 (4.0)	99.1 (1.5)

4.4 Discussion

The proposed technique was assessed in a proof of concept intelligent watermarking problem. These simulation results demonstrate that the proposed technique allows decreasing computational burden of dynamic optimization with little impact on precision for applications involving the optimization of a stream of recurrent problems. For example, in Figure 4, which shows the best fitness value for each generation for both, full optimization and the proposed approach, it is possible to observe that although the proposed technique resulted in less generations, the best recalled solutions (square and triangle marks) are very close to those obtained by full optimization.

The improvement compared with a previous (case-based) version of the proposed technique is also worth of notice and justifies the storage of GMM of solutions rather than selected solutions. Such improvement is mainly due to the fact that selected solutions tend to be over-fit to the optimization problems that generated them. For this reason, these sentries cover a limited region of the fitness landscape and tend to be more sensitive to small variations in the fitness values of the optimum caused by noise (as explained in our pseudo-type II formulation of such scenario).

Another advantage of the proposed approach is that it results in a more adaptive model than that created by the case-based approach and is thus more appropriate for applications where the stream of optimization problems to be optimized is heterogeneous (that is, the recurrent problems might be subject to noise). This is mainly due to the proposed adaptive memory management technique, which is novel and can be considered as one of the main contributions of this paper since it allows the memory to incrementally adapt to new trends on the stream of optimization problems.

5. CONCLUSION

A hybrid GMM/PSO dynamic optimization technique was proposed in this paper. The main objective of the proposed approach is to tackle optimization of streams of recurrent optimization problems. Such formulation of dynamic optimization is applicable to many practical applications, mainly those related to optimizing heuristic parameters of systems that process streamed data such as batch processing of images, video processing, incremental machine learning.

In the proposed technique, a two-level adaptive memory containing GMMs of solutions in the optimization space and global best solutions is incrementally built. For each new problem instance, solutions are re-sampled from this memory and employed as sentries in order to (1) measure the similarity between the new problem instance and previous instances that had already resulted in optimization; (2) provide ready-to-use solutions for recurrent problems, avoiding an unnecessary re-optimization.

It is worth noticing that the proposed technique resulted in a decrease of 90.4% in computational burden (compared to full optimization) with minimum impact on accuracy in an application involving a heterogeneous stream of document images. Although these results are still preliminary, they are comparable to results obtained in a previous version of the proposed approach already published, with the main difference that the experiments reported in this paper involved a much more challenging database which results in more varied (noisy) optimization problems.

As a future work we propose an evaluation in a larger stream of document images and also in synthetic benchmark functions which could allow a better understanding of the mechanisms behind the proposed approach.

6. ACKNOWLEDGMENTS

This work has been supported by National Sciences and Engineering Research Council of Canada and BancTec Canada Inc.

7. REFERENCES

[1] G. J. Barlow and S. F. Smith. Using memory models to improve adaptive efficiency in dynamic problems. In *IEEE Symposium on Computational Intelligence in Scheduling*, Nashville, March 2009.

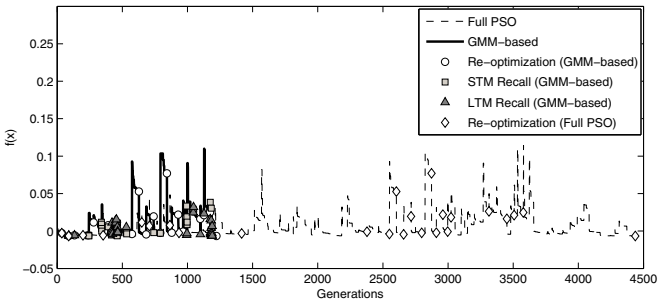


Figure 4: Tracking of best fitness for each generation using dataset OULU-1999-TRAIN.

[2] M. Blackwell. Particle swarms and population diversity. *Soft Comput.*, 9(11):793–802, 2005.

[3] T. Blackwell. *Evolutionary Computation in Dynamic Environments*, chapter Particle swarm optimization in dynamic environments. Springer, 2007.

[4] T. Blackwell and J. Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Trans. Evolutionary Computation*, 10(4):459–472, 2006.

[5] A. Carlisle and G. Dozier. Tracking changing extrema with adaptive particle swarm optimizer. *Proceedings of the 5th Biannual World Automation Congress, 2002.*, 13:265–270, 2002.

[6] Y. Collette and P. Siarry. On the sensitivity of aggregative multiobjective optimization methods. *CIT*, 16(1):1–13, 2008.

[7] M. Farina, K. Deb, and P. Amato. Dynamic multiobjective optimization problems: test cases, approximations, and applications. *IEEE Trans. Evolutionary Computation*, 8(5):425–442, 2004.

[8] M. A. T. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:381–396, 2000.

[9] C. Hennig. Methods for merging gaussian mixture components. *Advanced Data Analysis and Classification*, 4:3–34, 2010.

[10] X. Hu and R. Eberhart. Adaptive particle swarm optimization: detection and response to dynamic systems. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC '02.*, 2:1666–1670, 2002.

[11] M. N. Kapp, R. Sabourin, and P. Maupin. A dynamic model selection strategy for support vector machine classifiers. *Applied Soft Computing (accepted for publication)*, March 2011.

[12] J. Kennedy. Some issues and practices for particle swarm. In *Proc. IEEE Swarm Intelligence Symposium*, 2007.

[13] X. Li and K. H. Dam. Comparing particle swarms for tracking extrema in dynamic environments. In *Proceedings of the 2003 Congress on Evolutionary Computation. CEC '03.*, volume 3, pages 1772–1779, Dec. 2003.

[14] H. Lu, A. C. Kot, and Y. Q. Shi. Distance-reciprocal distortion measure for binary document images. *Signal Processing Letters, IEEE*, 11(2):228 – 231, February 2004.

[15] E. Muharemagic. *Adaptive Two-Level Watermarking for Binary Document Images*. PhD thesis, Florida Atlantic University, December 2004.

[16] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh. DNPSO: A dynamic niching particle swarm optimizer for multi-modal optimization. In *IEEE World Congress on Computational Intelligence*, pages 26–32, June 2008.

[17] D. Parrott and X. Li. A particle swarm model for tracking multiple peaks in a dynamic environment using speciation. In *Proceedings of the 2004 Congress on Evolutionary Computation. CEC '04.*, volume 1, pages 98–103, June 2004.

[18] M. Pelikan, D. E. Goldberg, and F. G. Lobo. A survey of optimization by building and using probabilistic models. *Comput. Optim. Appl.*, 21(1):5–20, 2002.

[19] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimisation: an overview. *Swarm Intelligence Journal*, 1(1):33–57, June 2007.

[20] J. Sauvola and H. Kauniskangas. MediaTeam Document Database II, a CD-ROM collection of document images, University of Oulu, Finland, 1999.

[21] G. Sfikas, C. Constantinopoulos, A. Likas, and N. P. Galatsanos. An analytic distance metric for gaussian mixture models with application in image retrieval. In *Proceedings of the 15th international conference on Artificial neural networks: formal models and their applications - Volume Part II*, ICANN'05, pages 835–840, Berlin, Heidelberg, 2005. Springer-Verlag.

[22] E. Vellasques, R. Sabourin, and E. Granger. A high throughput system for intelligent watermarking of bi-tonal images. *Applied Soft Computing*, 11(8):5215–5229, December 2011.

[23] H. Wang, D. Wang, and S. Yang. Triggered memory-based swarm optimization in dynamic environments. In *EvoWorkshops*, pages 637–646, 2007.

[24] S. Yang. Population-based incremental learning with memory scheme for changing environments. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 711–718, New York, NY, USA, 2005. ACM.

[25] S. Yang and X. Yao. Population-based incremental learning with associative memory for dynamic environments. *IEEE Transactions on Evolutionary Computation*, 12(5):542–561, Oct. 2008.