A dual-purpose memory approach for dynamic particle swarm optimization of recurrent problems

Eduardo Vellasques, Robert Sabourin, Eric Granger

evellasques@livia.etsmtl.ca, {robert.sabourin, eric.granger@etsmtl.ca}

École de technologie supérieure

Keywords: Evolutionary Computing; Particle Swarm Optimization; Dynamic Optimization; Gaussian Mixture Models; Estimation of Distribution Algorithms; Surrogate-Based Optimization

Abstract. In a dynamic optimization problem (DOP) the optima can change either in a sequential or in a recurrent manner. In sequential DOPs, the optima change gradually over time while in recurrent DOPs, previous optima reappear over time. The common strategy to tackle recurrent DOPs is to employ an archive of solutions along with information allowing to associate them with their respective problem instances. In this paper, a memory-based Dynamic Particle Swarm Optimization (DPSO) approach which relies on a dual-purpose memory for fast optimization of streams of recurrent problems is proposed. The dual-purpose memory is based on a Gaussian Mixture Model (GMM) of candidate solutions estimated in the optimization space which provides a compact representation of previously-found PSO solutions. This GMM is estimated over time during the optimization phase. Such memory operates in two modes: generative and regression. When operating in generative mode, the memory produces solutions that in many cases allow avoiding costly re-optimizations over time. When operating in regression mode, the memory replaces costly fitness evaluations with Gaussian Mixture Regression (GMR). 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. Results involving the use of the memory of GMMs in regression mode as a mean of replacing fitness evaluations (surrogate-based optimization) indicate that such learned memory also provides means of decreasing computational burden in situations where re-optimization cannot be avoided.

1 Introduction

Evolutionary computing (EC) allows tackling optimization problems where the derivatives are unknown (black box optimization) by evolving populations of candidate solutions through a certain number of generations, driven by one or more fitness functions. Because of its black box nature, EC has been successfully employed in many different practical applications. The drawback is that most of the techniques available in the literature assume that the optima location does not change (static optimization) while most practical applications have a dynamic nature.

Problems where the optima change with time are known in the literature as dynamic optimization problems (DOPs). In a DOP, 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) [1] and be followed or not by a period of stasis [2]. 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 as change occurs in short intervals of time. Spatial severity is usually low as peaks move in a smooth fashion, therefore stasis is small or inexistent.

In DOPs involving low spatial severity (e.g.: moving a robot through a 3D space), the optima usually moves within a region in the fitness landscape already surrounded by one or more candidate solutions. In such case, 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 (whichever is better) [3]. 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 [4, 5]. An alternative is to compute a running average of the fitness function for the best individuals over a certain number of iterations [6].

In recurrent problems [7–9], the spatial severity is high as the optima re-appears abruptly in a previous location but the temporal severity is low. Such type of DOP is usually linked to applications involving optimizing system parameters for streams of data like machine learning [10], digital watermarking [11] and video surveillance [12–14]. It has been demonstrated in the literature that the use of a memory of previously seen solutions can decrease the computational cost of optimization in such recurring environments [8].

One of the main strategies to decrease the computational cost of optimization in such scenario is through the use of a memory of ready-to-use solutions, as demonstrated in [11]. The motivation for the use of ready-to-use solutions is that the optimization of streamed data (which corresponds to optimizing a stream of problem instances) can be seen as 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. Time severity is inexistent as the exact moment a new problem instance arrives is known beforehand. Put differently, stasis can have an indefinite length. Such case of type III change can be considered as a pseudo-type II change. Optimal solutions are interchangeable across problem instances involving pseudo-type II change. Due to its fast convergence, Particle Swarm Optimization (PSO) is preferred in such time-constrained applications.

A novel memory-based Dynamic PSO (DPSO) technique has been proposed for fast optimization of recurring dynamic problems, where a two-level memory of selected solutions and Gaussian Mixture Models (GMM) of their corresponding environments is incrementally built [15]. 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 solution is employed directly if both distributions are considered similar. Otherwise, L-best PSO [10] is employed in order to optimize parameters. Vellasques *et al* [16] present a more detailed description of that technique, including a comprehensive experimental validation in many different scenarios involving homogeneous and heterogeneous problem streams. In the present paper, a study on the use of surrogates as a tool to decrease the computational cost of the L-best PSO is presented. The main research problem addressed in the given work is how to employ a model of the stream of optimization problems in order to at the same time (1) generate ready-to-use solutions which allow avoiding re-optimization and (2) replace costly fitness evaluations with regression when re-optimization cannot be avoided. The research hypothesis is that density estimates of historical solutions found during the optimization phase allow tackling (1) and (2) at the same time.

The application which motivates this research is the optimization of embedding parameters for digital watermarking systems in scenarios involving streams of document images. Digital watermarking allows enforcing authenticity and integrity of such type of image which is a major security concern for many different industries including financial, healthcare and legal. Since the protection provided by digital watermarks is minimally intrusive, it can be easily integrated into legacy document management systems (allowing an extra layer of security with minimum implementation costs).

The research presented in this paper is a continuation of the research presented in [15]. The main distinction, is that in the approach presented in this paper, the GMM memory of solutions serves two main purposes – (1) generating ready-to-use solutions and (2) replacing fitness evaluation with regression. Therefore, the main contribution of this paper is that here it is demonstrated that a previously learned memory of GMMs not only provides means of avoiding costly re-optimization operations but also makes possible a further decrease in computational burden in situations where re-optimization cannot be avoided. The surrogate strategy employed in these experimental simulations is based on the strategy proposed by Parno *et al* [17] with the difference that in the proposed strategy, no surrogate update takes place since it is assumed that a memory of GMMs learned over a training sequence of optimization problems should provide enough knowledge about future similar problems, which would make possible replacing fitness evaluations with regression. Thus, surrogate-based optimization is formulated as an unsupervised learning problem. There are two reasons for using a surrogate in the envisioned application. The first reason is that re-optimization cannot be completely avoided and when it is made necessary, its computational cost is much higher than that of recall. Therefore, there is a considerable amount of computational cost savings to be made for such operation. The second reason is that although a surrogate can lead to such computational cost savings, it also requires costly fitness evaluations. However, in a scenario involving recurring problems, even when re-optimization cannot be avoided, previously learned GMMs can offer a good approximation of the new problem, avoiding part of the costs involved in training surrogates.

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

2 Dynamic PSO (DPSO)

PSO [18] 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 ($\mathbf{p}_g$):

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

where χ is a constriction factor, chosen to ensure convergence [19], 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 $\mathbf{p}_g$ [20]), 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 [20]. 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:

$$\mathbf{x}_i = \mathbf{x}_i + \mathbf{v}_i \tag{2}$$

Canonical PSO is tailored for the optimization of static problems. However, numerous real world problems have a dynamic nature. A DOP is either defined as a sequence of static problems linked up by some dynamic rules or as a problem that has time-dependent parameters in its mathematical formulation [21].

The three main issues that affect the performance of EC algorithms in DOPs are (1) outdated memory, (2) lack of change detection mechanism and (3) diversity loss [22, 4]. It is important to mention that for PSO, outdated memory can be easily tackled by re-evaluating the fitness function for the new problem instance [3, 5].

Change detection mechanisms either assume that changes in the environment are made known to the optimization algorithm or that they need to be detected [21]. In the proposed formulation of a DOP, each problem instance in the stream of optimization problems is static and the moment a transition between any two instances occurs is known. However, the similarity between a new and previously seen instances is unknown and the objective of change detection in this concept is to measure the similarity between new and previously seen problem instances. The most common change detection strategy is based on the use of fixed sentry particles, re-evaluated at each generation [4]. Another strategy relies on measuring algorithmic behavior with the use of a statistical test [21].

Tackling diversity loss requires more elaborate techniques, which can be categorized as [23]: 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 [6]) 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 Parno *et al* [24].

Problems involving one or more states re-appearing over time are better tackled through the use of memory-based schemes [8]. The main reason is that in such type of DOP, the transition between one or more problem instances is not smooth as assumed in [24] and the three strategies described above are of no use when the optimum moves away from the area surveyed by the swarm. In such case, as stated by Nguyen *et al* [21], the optima may return to the regions near their previous locations, thus it might be useful to re-use previously found solutions to save computational time and to bias the search process. As observed by Yang and Yao [8], 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 Models (GMMs) of solutions in the optimization space along with their respective global best solutions. The main motivation for relying on GMM representation of the fitness landscape is that as stated by Nguyen *et al* [21], the general assumption of a DOP is that the problem after a change is somehow related to the problem before a change, and thus an optimization algorithm needs to learn from its previous search experience as much as possible. The use of probabilistic models in EC is not new. Such approach, known as Estimation of Distribution Algorithms (EDA) [25] 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 Proposed approach

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 recurring 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. Optimization relies on a surrogated-based PSO algorithm. Such approach relies on the use of two populations – one based on exact fitness evaluations and another one which replaces exact fitness evaluations with a regression model. Both populations tackle optimization with the use of the L-best PSO (Section 2). The change detection module was adapted to the memory recall mechanism employed in the proposed approach. Although each problem instance is static, this PSO variant allows tackling multi-modal optimization problems and the motivation behind the proposed technique is tackling dynamic optimization of recurring 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[26]. 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).

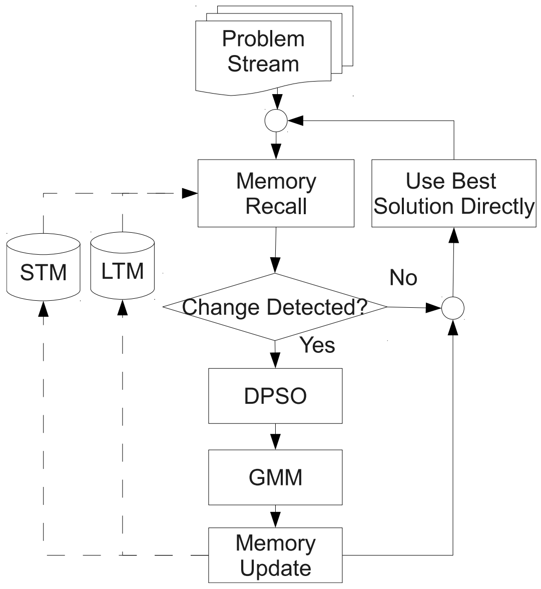

Fig. 1: Hybrid GMM/DPSO framework.

Thus, the proposed approach relies on the use of an associative memory [8]. In an associative memory approach, selected solutions are stored in a separate archive along with a density estimate that allows associating these solutions with recurring environments. However, an important distinction has to be made here. In the associative memory approaches found in the literature, the density estimates are trained using only position data while in the proposed approach, the density is trained using fitness and position data. The main motivation is that in the proposed approach, a density provides not only means of associating solutions with recurring problems, but it is also an important part of the change detection mechanism (sampled fitness values are compared with re-evaluated fitness values in order to measure the similarity between the new and previously seen problems). Therefore, instead of providing a hint of locations where good solutions are likely to be found, a memory element in the proposed approach provides a topographic map of one or more optimization problems while the change detection mechanism provides means of probing the new optimization problem in order to compare how similar both landscapes are. This GMM representation of the fitness landscape is also employed as a regression model for surrogate-based optimization.

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, the GMM approach of Figueiredo and Jain [26] 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

$$p(\boldsymbol{x}|\Theta) = \sum_{k=1}^{K} \alpha_k p(\boldsymbol{x}|\theta_j) \tag{3}$$

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.

In the GMM approach employed in this framework, the mixture is initialized with a large number of components, where each component is centered at a randomly picked data point. Training is based on the use of Expectation Maximization (EM) where the E-step and M-step are applied iteratively. In the E-step, the posterior probability given the data is computed:

$$w_{i,j}^{(t)} = \frac{\alpha_j p(\boldsymbol{x}_i|\theta_j)}{\sum_{k=1}^{K} \alpha_k p(\boldsymbol{x}_i|\theta_k)} \tag{4}$$

Then, in the M-step, the model parameters are updated. Authors propose a slightly modified variant of mixing weights update, which discounts the number of parameters in each Gaussian (N):

$$N = d + \frac{d(d+1)}{2} \tag{5}$$

$$\alpha_j^{(t+1)} = \frac{max\{0, (\sum_{i=1}^{n} w_{i,j}) - N/2\}}{\sum_{k=1}^{K} max\{0, (\sum_{i=1}^{n} w_{i,k}) - N/2\}} \tag{6}$$

where d is the number of dimensions of $\boldsymbol{x}$ and n is the number of data points. The remaining parameters are updated as:

$$\boldsymbol{\mu}_j^{(t+1)} = \frac{\sum_{i=1}^{n} w_{i,j}^{(t)} \boldsymbol{x}_i}{w_{i,j}} \tag{7}$$

$$\boldsymbol{\Sigma}_j^{(t+1)} = \frac{\sum_{i=1}^{n} w_{i,j}^{(t)} (\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(t+1)})(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{(t+1)})^T}{w_{i,j}} \tag{8}$$

However, during learning as the model is updated (1) components lacking enough data points to estimate their covariance matrices have their corresponding mixing weights set to zero (component annihilation) and (2) the number of components is gradually decreased until a lower boundary is achieved and then, the number that resulted in the best performance is chosen.

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 [27] (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 $C2$ distance between two mixtures Θ and Θ' is defined as:

$$\boldsymbol{\phi}_{i,j} = (\boldsymbol{\Sigma}_i^{-1} + \boldsymbol{\Sigma}_j'^{-1})^{-1} \tag{9}$$

$$\nu_{i,j} = \boldsymbol{\mu}_i^T \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j') + \boldsymbol{\mu}_j^T \boldsymbol{\Sigma}_j'^{-1}(\boldsymbol{\mu}_j' - \boldsymbol{\mu}_i') \tag{10}$$

$$C2(\Theta,\Theta') = -log\left[\frac{2\sum_{i,j}\alpha_i\alpha_j'\sqrt{|\phi_{i,j}|/(e^{\nu_{i,j}}|\boldsymbol{\Sigma}_i||\boldsymbol{\Sigma}_j'|)}}{\sum_{i,j}\alpha_i\alpha_j\sqrt{|\phi_{i,j}|/(e^{\nu_{i,j}}|\boldsymbol{\Sigma}_i||\boldsymbol{\Sigma}_j|)} + \sum_{i,j}\alpha_i'\alpha_j'\sqrt{|\phi_{i,j}|/(e^{\nu_{i,j}}|\boldsymbol{\Sigma}_i'||\boldsymbol{\Sigma}_j'|)}}\right] \tag{11}$$

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 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 [28]. The Bhattacharyya distance is defined as:

$$\bar{\boldsymbol{\Sigma}} = \frac{1}{2}(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2) \tag{12}$$

$$\begin{aligned} d_B(\Theta_1,\Theta_2) &= (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \bar{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ &+ \frac{1}{2}\log\left(\frac{|\frac{1}{2}(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)|}{\sqrt{|\boldsymbol{\Sigma}_1||\boldsymbol{\Sigma}_2|}}\right) \end{aligned} \tag{13}$$

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, un-merged components from the older mixture are deleted (the old un-merged 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 ($\mu^t_{\boldsymbol{\delta}}$ and $\sigma^t_{\boldsymbol{\delta}}$) are initialized based on pre-defined values ($\mu^0_{\boldsymbol{\delta}}$ and $\sigma^0_{\boldsymbol{\delta}}$, steps 4 and 5). Otherwise, $\mu^t_{\boldsymbol{\delta}}$ and $\sigma^t_{\boldsymbol{\delta}}$ 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 $\mu^t_{\boldsymbol{\delta}}$ is greater than the standard deviation ($\sigma^t_{\boldsymbol{\delta}}$, steps 12 to 16). It is important to notice that before the insert, the LTM probe with the smallest number of recalls is deleted if the 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).

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.

```
1: Estimate $\Theta_N$ using $\mathfrak{D}$ [26].
2: Add $\Theta_N$ and $\boldsymbol{p}_g$ to $\mathfrak{M}_S$.
3: if $|\boldsymbol{\delta}| < T$ then
4:     $\mu^t_{\boldsymbol{\delta}} \leftarrow \mu^0_{\boldsymbol{\delta}}$
5:     $\sigma^t_{\boldsymbol{\delta}} \leftarrow \sigma^0_{\boldsymbol{\delta}}$
6: else
7:     $\mu^t_{\boldsymbol{\delta}} \leftarrow \frac{1}{|\boldsymbol{\delta}|}\sum_{i=1}^{|\boldsymbol{\delta}|}\delta_i$
8:     $\sigma^t_{\boldsymbol{\delta}} \leftarrow \sqrt{\frac{\sum_{i=1}^{n}(\delta_i - \mu^t_{\boldsymbol{\delta}})^2}{|\boldsymbol{\delta}|}}$
9: end if
10: $i^* \leftarrow argmin_i\{C2(\Theta_N, \Theta_i)\}, \ \forall_{\Theta_i \in \mathfrak{M}}$
11: $\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} \cup C2(\Theta_N, \Theta_{i^*})$
12: if $C2(\Theta_N, \Theta_{i^*}) - \mu^t_{\boldsymbol{\delta}} > \sigma^t_{\boldsymbol{\delta}}$ then
13:     if $|\mathfrak{M}| = L_{\mathfrak{M}}$ then
14:         Remove LTM probe with smallest number of successful recalls.
15:     end if
16:     Add $\Theta_N$ and $\boldsymbol{p}_g$ to $\mathfrak{M}$
17: else
18:     $Merge(\Theta_{i^*}, \Theta_N)$ (section 3.2)
19:     Purge merged mixture in case number of elements exceed $k_{max}$.
20: end if
21: if $|\boldsymbol{\delta}| > T$ then
22:     Remove $\delta_1$.
23: end if
```

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:

$$\boldsymbol{X}_s = \boldsymbol{\mu}_j + \boldsymbol{\Sigma}_j \boldsymbol{R}_s \tag{14}$$

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

3.4 Surrogated-based particle swarm optimization

Since the proposed method relies on several GMMs learned with the use of a training sequence, a promising strategy to further decrease the computational cost associated in such recurring optimization problems is to employ the GMMs in regression mode whenever re-optimization is triggered. Such approach is known in the literature as surrogate-based optimization [29]. In surrogate-based optimization, the fitness landscape is sampled with the use of a sampling plan (design of experiments). Then, during optimization a portion of the exact fitness evaluations is replaced with approximate fitness values obtained through regression and the surrogate is updated with newly sampled points as necessary.

The surrogate-based strategy described in [17] is employed in the proposed framework. This surrogate-based approach employs two populations ($\boldsymbol{X}_A$ and $\boldsymbol{X}_B$) in parallel, one based on surrogate fitness evaluations and another one based on exact fitness evaluations. During initialization, solutions sampled from the GMM memory are injected into $\boldsymbol{X}_B$. Optimization is first performed in $\boldsymbol{X}_A$. Then, the best solution found in the surrogate fitness optimization ($\boldsymbol{p}_{g,s2}$) is re-evaluated in the exact fitness. If it improves the neighborhood best of population $\boldsymbol{X}_B$, that neighborhood best is replaced with $\boldsymbol{p}_{g,s2}$. After that, an iteration of optimization is performed using population $\boldsymbol{X}_B$ on the exact fitness. This process is repeated until a stop criterion has been reached.

However, differently than the approach proposed in [17], no surrogate update is employed. The motivation is that a memory of previously learned GMMs already provides a valuable knowledge about new optimization problems.

In terms of regression, the proposed strategy relies on the Gaussian Mixture Regression (GMR) approach described in [30]. The advantage of Sung's approach is that it requires no modification to the proposed GMM learning. Moreover, it provides a distribution of the predicted value with $\hat{\boldsymbol{f}}(\boldsymbol{x})$ as the mean and $\boldsymbol{\varepsilon}^2(\boldsymbol{x})$ as the covariance matrix.

Here $\boldsymbol{\varepsilon}^2(\boldsymbol{x})$ can be seen as the amount of uncertainty about the predicted value. Since it is important to allow for exploration, this quantity will be discounted from the predicted value which should direct search towards unexplored regions of the fitness landscape (higher uncertainty). More formally, costly calls to the exact fitness $f(\boldsymbol{x})$ are partially replaced by a predicted fitness $f_P(\boldsymbol{x}, \Theta)$ using the strategy proposed by Torczon and Trosset [31]:

$$f_P(\boldsymbol{x}, \Theta) = \hat{f}(\boldsymbol{x}, \Theta) - \rho_c \varepsilon(\boldsymbol{x}, \Theta) \tag{15}$$

where $\hat{f}(\boldsymbol{x}, \Theta)$ is an approximation to $f(\boldsymbol{x})$ based on model Θ, ρ_c is a constant that dictates how much emphasis will be put in exploring unknown regions of the model and $\varepsilon(\boldsymbol{x})$ is the prediction error.

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 [11]. 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 $Q_R = Q_F + \varDelta_Q$ where $\varDelta_Q$ is a parameter to be optimized. More details can be found in [11]. 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 [32]; 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 ($\varDelta_Q$), 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) [33], which measures the quality of the watermarked image (more details can be found in [11]). The three fitness values are aggregated using Chebyshev approach [34]:

$$\begin{aligned} F(\mathbf{x}) = max_{i=1,..,3}\{&(1-\omega_1)(\alpha_s DRDM - r_1), \\ &(1-\omega_2)(1 - BCR_R - r_2), \\ &(1-\omega_3)(1 - BCR_F - r_3)\} \end{aligned} \tag{16}$$

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.

Each image correponds to an optimization problem. In situations involving streams of images with similar structure (like document images) it is very likely that some of the images in the stream will have similar embdedding capacity. In such case, a stream of document images can be seen as a stream of recurrent optimization problems.

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.

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

Oulu University's MediaTeam [35] (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 [11]. 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 SNDM 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	**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.

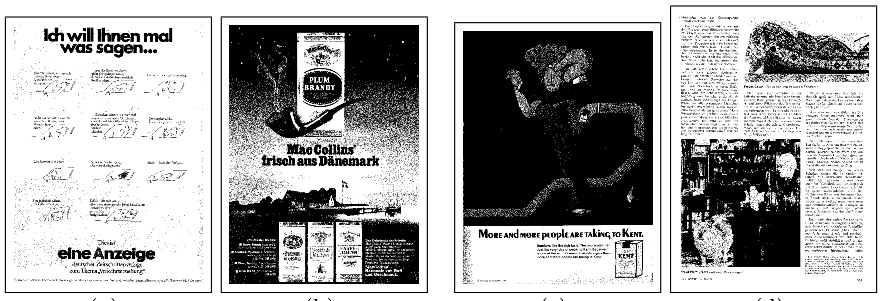

Fig. 3: Images from OULU-1999-TRAIN. (a) Image 1. (b) Image 2. (c) Image 5. (d) Image 6.

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:

$$DFE = 1 - \frac{F_{Evals,M}}{F_{Evals,F}} \tag{17}$$

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. Full optimization means applying the PSO agorithm described in Section 2 without resorting to neither memory recall nor to surrogates.

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 [11], 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 [11] (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 [11] to demonstrate the main advantages of employing a memory of mixture models. In the case-based approach, the STM and LTM contain the local bests of each particle, obtained at the end of the optimization process.

In the simulations involving surrogates, re-optimization is forced for each image, as a mean of generating enough data to validate the contribution of surrogates in decreasing the computational cost of re-optimization. Recall is performed only as a mean of assigning the best model for each new problem (the one with smallest KS is chosen as the most appropriate). Therefore, the comparison to be made is between the surrogate-based approach and full optimization since the main objective here is to decrease the cost of full optimization in situations where re-optimization cannot be avoided. After each transition, 70% of the swarm is randomized and the remaining 30% solutions are replaced with solutions sampled from the STM. The memory is created by applying the GMM-based approach to the training stream, with full optimization activated (when re-optimization is triggered) and with the merge operator de-activated. However, since the underlying assumption of the proposed strategy is that previously learned surrogates provide valuable knowledge about new problems, it is important to define a matching strategy between previously learned surrogates and new problems. The strategy proposed here is to simply choose the GMM that results in the smallest KS value during recall.

The simulations are conducted in the heterogeneous OULU database described before. In order to understand the behavior of surrogates in more stable scenarios, simulations are also conducted using the homogeneous database of scientific documents from Computer Vision and Image Understanding (CVIU) journal. The training database (TITI-61) contains 61 pages from issues 113(1) and 113(2), split in two categories – 30 pages of text and 31 pages of images – while the test database (CVIU-113-3-4) contains 342 pages of 29 complete papers from CVIU 113(3) and 113(4). More details about both databases can be found in [11].

4.3 Simulation results

Avoidance of re-optimization In the first set of experiments, the surrogates are de-activated and the proposed recall mechanism works as described in Section 3. For each image, as soon as a case of KS value between the re-sampled and re-evaluated smaller than the critical value is found, the best recalled solution is employed directly and re-optimization is avoided. Otherwise, if no such case is found, full optimization takes place. 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).

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.

Variant	**Subset**	**†**	**‡**	**§**
Full PSO	TRAIN	0.03(0.03)	98.4(2.1)	99.7(0.6)
	TEST	0.03(0.04)	98.4(2.2)	99.6(0.6)
Case-based	TRAIN	0.03(0.03)	97.9(2.6)	99.6(1)
	TEST	0.03(0.03)	97.2(3.6)	99(1.6)
GMM-based	TRAIN	0.03(0.03)	97.5(2.8)	99.4(1.0)
	TEST	0.03(0.03)	96.7(4.0)	99.1(1.5)

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 recurring 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.

Surrogate-based optimization performance Table 4 shows the computational cost and watermarking performance of the simulations involving surrogates. It is possible to observe that in both cases (heterogeneous and homogeneous image streams), the use of a surrogate allowed a slight decrease in computational burden with a watermarking performance identical to that of full optimization.

Table 4: Computational cost and watermarking performance of the surrogate-based strategy. $AFPI$ 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. † is the $DRDM$, ‡ is the BCR robust, § is the BCR fragile.

Subset	$AFPI$	F_{Evals}	DFE	†	‡	§
OULU-1999-TEST	831(334)	330021	3.4%	0.03(0.03)	98.4(2.2)	99.6(0.7)
CVIU-113-3-4	787(322)	269168	12.3%	0.02(0.04)	98.8(2.2)	99.5(0.4)

The computational cost performance for the homogeneous stream is slightly better than that of the heterogeneous stream with a decrease of 12.3% in the number of fitness evaluations when compared to full optimization. The reason is that the memory learned with the use of a training sequence represents better the images found in the test database for that specific stream.

These results depict the main advantage of using a surrogate. Although the gains are not substantial as in the case where re-optimization is avoided, they are more robust in terms of watermarking performance and are a better alternative compared to performing full optimization.

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 recurring 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.

Simulations involving the use of previously learned GMMs as surrogates demonstrate that such memory of GMMs allows not only avoiding re-optimization in situations involving recurring problems but also provides means of directing the search process for not so similar problems. Such strategy allows formulating surrogate-based optimization as a

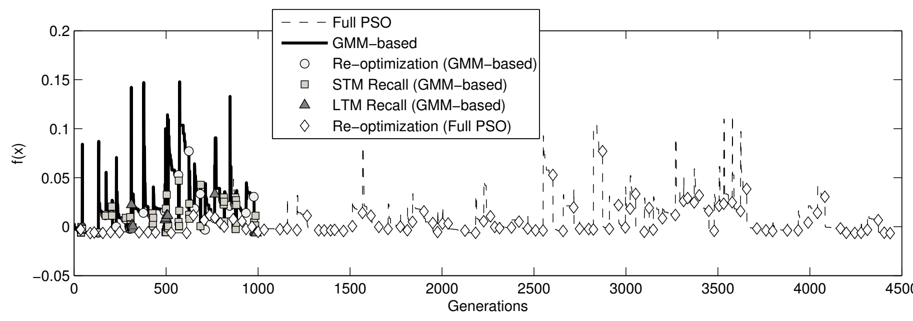

Fig. 4: Tracking of best fitness for each generation using dataset OULU-1999-TRAIN.

machine learning problem. One of the major concerns in surrogate-based optimization is the cost associated with probing the new environment during optimization. However, these simulation results demonstrate that it is possible to learn such a memory of surrogates in a controlled (training) environment and then, deploy this memory of surrogates to a production environment where the constraints on performance are higher. It is interesting to observe in the simulations involving the use of surrogates that the decrease in the number of fitness evaluations was higher for the homogeneous stream than for the heterogeneous stream (12.3% versus 3.4%). Considering that in both cases re-optimization has been forcibly triggered for each image, the only possible explanation is that the train stream represents better the test stream for the homogeneous stream than for the heterogeneous stream. These results suggest that the surrogate model must be updated as optimization of new optimization problems takes place.

The trade-off between precision and computational burden is driven by (1) the number of times optimization is triggered and (2) the speed up in convergence provided by the surrogates. Therefore it is possible to improve precision by either employing a more restrictive decision threshold in the Kolmogorov-Smirnov test employed on change detection or by relying less on surrogate optimization (applying one round of surrogate optimization for every two iterations of exact optimization, for example).

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 recurring 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 recurring 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.

Simulation results involving the use of the memory of GMMs as surrogates demonstrate that it is possible to use the knowledge of previously seen problems as a mean of decreasing the computational cost of re-optimization in situations where re-optimization cannot be avoided. Even though model update was not employed, the use of surrogates resulted in a decrease of 3.4% in the number of fitness evaluations for a heterogeneous stream and 12.3% for an homogeneous stream. To the best of our knowledge, such finding advances the state-of-the-art in the surrogate-based optimization literature by separating surrogate learning (or modeling) which can be performed in a controlled environment from prediction in a more constrained production environment. The superior performance for homogeneous database indicate that model adaptation is an important issue for heterogeneous streams of optimization problems. This means that in the same manner an adaptive memory is required to tackle recall of heterogeneous streams an adaptive surrogate is required to tackle optimization in such scenario. 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. We also propose validating the proposed technique in other applications where such stream of recurring optimization problems is applicable such as incremental learning, video processing.

6 Acknowledgments

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

References

1. A. Nickabadi, M. M. Ebadzadeh, R. Safabakhsh, DNPSO: A dynamic niching particle swarm optimizer for multi-modal optimization, in: IEEE World Congress on Computational Intelligence, 2008, pp. 26–32.
2. M. Farina, K. Deb, P. Amato, Dynamic multiobjective optimization problems: test cases, approximations, and applications, IEEE Transactions on Evolutionary Computation 8 (5) (2004) 425–442.
3. T. Blackwell, J. Branke, Multiswarms, exclusion, and anti-convergence in dynamic environments, IEEE Trans. Evolutionary Computation 10 (4) (2006) 459–472.
4. A. Carlisle, G. Dozier, Tracking changing extrema with adaptive particle swarm optimizer, Proceedings of the 5th Biannual World Automation Congress, 2002. 13 (2002) 265–270.
5. X. Hu, R. Eberhart, Adaptive particle swarm optimization: detection and response to dynamic systems, Proceedings of the 2002 Congress on Evolutionary Computation. CEC '02. 2 (2002) 1666–1670.
6. H. Wang, D. Wang, S. Yang, Triggered memory-based swarm optimization in dynamic environments, in: EvoWorkshops, 2007, pp. 637–646.
7. G. J. Barlow, S. F. Smith, Using memory models to improve adaptive efficiency in dynamic problems, in: Computational Intelligence in Scheduling, 2009. CI-Sched '09. IEEE Symposium on, 2009, pp. 7–14.
8. S. Yang, X. Yao, Population-based incremental learning with associative memory for dynamic environments, IEEE Transactions on Evolutionary Computation 12 (5) (2008) 542–561.
9. X. Li, K. H. Dam, Comparing particle swarms for tracking extrema in dynamic environments, in: Proceedings of the 2003 Congress on Evolutionary Computation. CEC '03., Vol. 3, 2003, pp. 1772–1779.
10. M. N. Kapp, R. Sabourin, P. Maupin, A dynamic model selection strategy for support vector machine classifiers, Applied Soft Computing 12 (8) (2012) 2550–2565.
11. E. Vellasques, R. Sabourin, E. Granger, A high throughput system for intelligent watermarking of bi-tonal images, Applied Soft Computing 11 (8) (2011) 5215–5229.
12. J. F. Connolly, E. Granger, R. Sabourin, Evolution of heterogeneous ensembles through dynamic particle swarm optimization for video-based, Pattern Recognition 45 (7) (2012) 2460–2477.
13. J. F. Connolly, E. Granger, R. Sabourin, An adaptive classification system for video-based face recognition, Information Sciences 192 (2012) 50–70.
14. J. F. Connolly, E. Granger, R. Sabourin, Dynamic multi-objective evolution of classifier ensembles for video-based face recognition, Applied Soft Computing 13 (6) (2013) 3149–3166.
15. E. Vellasques, R. Sabourin, E. Granger, Gaussian mixture modeling for dynamic particle swarm optimization of recurrent problems, in: GECCO '12: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, 2012, pp. 73–80.
16. E. Vellasques, R. Sabourin, E. Granger, Fast intelligent watermarking of heterogeneous image streams through mixture modeling of PSO populations, Applied Soft Computing 13 (6) (2013) 3130–3148.
17. M. D. Parno, T. Hemker, K. R. Fowler, Applicability of surrogates to improve efficiency of particle swarm optimization for simulation-based problems, Engineering Optimization 44 (5) (2012) 521 – 535.
18. J. Kennedy, Some issues and practices for particle swarms, in: Swarm Intelligence Symposium, 2007. SIS 2007. IEEE, 2007, pp. 162–169.
19. M. Blackwell, Particle swarms and population diversity, Soft Comput. 9 (11) (2005) 793–802.
20. R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimisation: an overview, Swarm Intelligence Journal 1 (1) (June 2007) 33–57.

21. T. T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: A survey of the state of the art, Swarm and Evolutionary Computation 6 (0) (2012) 1 – 24.
22. T. Blackwell, Evolutionary Computation in Dynamic Environments, Springer, 2007, Ch. Particle swarm optimization in dynamic environments, pp. 29–49.
23. 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, ACM, New York, NY, USA, 2005, pp. 711–718.
24. D. Parrott, 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., Vol. 1, 2004, pp. 98–103.
25. M. Pelikan, D. E. Goldberg, F. G. Lobo, A survey of optimization by building and using probabilistic models, Computational Optimization and Applications 21 (1) (2002) 5–20.
26. M. A. T. Figueiredo, A. K. Jain, Unsupervised learning of finite mixture models, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2000) 381–396.
27. G. Sfikas, C. Constantinopoulos, A. Likas, 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, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 835–840.
28. C. Hennig, Methods for merging gaussian mixture components, Advanced Data Analysis and Classification 4 (2010) 3–34.
29. N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, P. K. Tucker, Surrogate-based analysis and optimization, Progress in Aerospace Sciences 41 (1) (2005) 1 – 28.
30. H. G. Sung, Gaussian mixture regression and classification, Ph.D. thesis, Rice University (2004).
31. V. Torczon, M. W. Trosset, Using approximations to accelerate engineering design optimization, in: Proceedings of the 7th AIAA/USAF/NASA/ISSMO Multidisciplinary Analysis and Optimization Symposium, Saint Louis, USA, 1998, pp. 738–748.
32. E. Muharemagic, Adaptive two-level watermarking for binary document images, Ph.D. thesis, Florida Atlantic University (December 2004).
33. H. Lu, A. C. Kot, Y. Q. Shi, Distance-reciprocal distortion measure for binary document images, IEEE Signal Processing Letters 11 (2) (2004) 228–231.
34. Y. Collette, P. Siarry, On the sensitivity of aggregative multiobjective optimization methods, CIT 16 (1) (2008) 1–13.
35. J. Sauvola, H. Kauniskangas, MediaTeam Document Database II, a CD-ROM collection of document images, University of Oulu, Finland (1999).