An Evaporation Mechanism for Dynamic and Noisy Multimodal Optimization

Jose Luis Fernandez-Marquez and Josep Lluis Arcos
IIIA, Artificial Intelligence Research Institute
CSIC, Spanish National Research Council
Campus UAB, Bellaterra, E-08193 (SPAIN)
{fernandez,arcos}@iiia.csic.es

ABSTRACT
Dealing with imprecise information is a common characteristic in real-world problems. Specifically, when the source of the information are physical sensors, a level of noise in the evaluation has to be assumed. Particle Swarm Optimization is a technique that presented a good behavior when dealing with noisy fitness functions. Nevertheless, the problem is still open. In this paper we propose the use of the evaporation mechanism for managing with dynamic multi-modal optimization problems that are subject to noisy fitness functions. We will show how the evaporation mechanism does not require the detection of environment changes and how can be used for improving the performance of PSO algorithms working in noisy environments.

Categories and Subject Descriptors
I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search—Heuristic methods; I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—Multi-agent systems

General Terms
Algorithms

Keywords
PSO, multimodal dynamic environments, noisy functions

1. INTRODUCTION
In order to solve real-world optimization problems, several uncertainty issues have to be considered. As it was described in [11], uncertainty issues can arise from four different origins: the evaluation of the fitness function may be subject to noise; the design variables may be subject to perturbations; the fitness function may be only approximated; or the optimum of the problem may change over time.

In this paper we are interested in addressing optimization problems when i) the problem may change over time (dynamic environments) and ii) the evaluation of the fitness function is subject to noise.

Particle Swarm Optimization (PSO) has been proved as an efficient mechanism for static functions. Since 2001 when Parsopoulos and Vrahatis [13] showed that PSO could track slowly moving optima without any changes at all, several authors have proposed different variants for improving the performance of PSO in dynamic environments.

Specifically, in order to improve the original PSO for working on dynamic environments, two main problems have to be addressed [2]: the outdated memory (due to the environment dynamism) and the diversity loss (due to particles’ convergence). The outdated memory problem is related to the storage of the best position found. When a change in the environment occurs, the best solution found may become obsolete and may misguide the particle’s search. The diversity loss problem appears when the swarm has converged and an environment change occurs. If the peak where the swarm converged disappear or changes its location, the low velocity of the particles inhibits the tracking and the particles may oscillate around a false attractor.

As we will describe in the next section, several variants of the PSO algorithm have been proposed for dealing with the diversity problem. In the existing approaches, the outdated memory problem is easily solved by re-evaluating periodically the best positions found. This solution has been proved enough in environments when the fitness function is not subject to noise.

Regarding to the evaluation of fitness functions that are subject to noise, the main challenge appears when particles try to detect changes in the environment. The noise associated to the evaluation of the fitness function difficults this detection. [13] also showed some experimental results that PSO is somehow noise tolerant. Nevertheless, some detected changes may be false and some real changes may not be detected.

The goal of this research is to propose a mechanism for improving the performance of PSO algorithms in dynamic and noisy environments. Specifically, we have extended the Multi Quantum Swarm Optimization (mQSO) algorithm [4] by adding an evaporation mechanism that improves the convergence in dynamic multimodal environments when the evaluation of the fitness function is subject to noise. Moreover, the evaporation mechanism avoids the continuous detection of changes in the environment.
The paper is organized as follows. Next section briefly reviews the research related to our problem. Section 3 proposes the evaporation mechanism and describes how it is incorporated into the mQSO algorithm. Section 4 presents some empirical results. Finally in section 5 we draw some conclusions and set paths to future research.

2. BACKGROUND

The initial ideas on particle swarm optimization were essentially aimed at producing computational intelligence by exploiting simple analogues of social interaction, rather than purely individual cognitive abilities. Potential solutions in PSO are represented as particles.

2.1 Particle Swarm Optimization

In PSO, each particle has a position $\vec{p}_i$ and a velocity $\vec{v}_i$. Initially, the set of particles is randomly distributed in the search space with a random initial velocity. The position and velocity of each particle are modified iteratively. The movement of the particles is determined by combining some aspect of its experience, as its best position found $\vec{b}_i$, with social information, as the best position of its neighbors, $\vec{g}$. At each algorithm iteration, particles evaluate the objective function at its current location (fitness value) $f_u(\vec{p}_i)$.

Since 1995 when James Kennedy and Eberhart proposed the PSO algorithm, some extensions and optimizations of their parameters have been realized [14]. One of them, the constriction coefficients has been well accepted by the community. The constriction coefficients control the convergence of the particle and allow an elegant and well-explained method for preventing explosion, ensuring convergence and eliminating the arbitrary $V_{max}$ parameter.

The movement of each particle follows the next two equations:

$$\vec{v}_i = \chi(\vec{v}_i + \vec{U}(0, \phi_1)(\vec{b}_i - \vec{p}_i) + \vec{U}(0, \phi_2)(\vec{g} - \vec{p}_i))$$  \hspace{1cm} (1)

$$\vec{p}_i = \vec{p}_i + \vec{v}_i$$  \hspace{1cm} (2)

Where:

• $\chi$ is the constant multiplier that ensures the convergence;

• $\vec{p}_i$ is the current position of the particle $i$;

• $\vec{v}_i$ is the velocity of the particle $i$;

• $\vec{b}_i$ the best position found by the particle $i$;

• $\vec{g}$ the global best solution found by the particles; and

• $\vec{U}(0, \phi_i)$ represents a vector of random numbers uniformly distributed in $[0, \phi_i]$.

As we can see in the equation 1, there are two parts: the cognitive part $\vec{U}(0, \phi_1)(\vec{b}_i - \vec{p}_i)$ that drives the particles to its best position found and the social part $\vec{U}(0, \phi_2)(\vec{g} - \vec{p}_i)$ that drives the particles to the best position found by the swarm.

2.2 PSO in Dynamic Environments

Optimization in dynamic environments is a challenging problem for PSO. Different extensions of PSO have been proposed to improve its adaptiveness in dynamic environments. Extensions (see [14] for an overview) propose solutions such as resetting the position of particles frequently or using a multi-swarm model.

Diversity loss has been addressed either by introducing randomization, repulsion, dynamic networks, or multi-populations [2]. One important contribution was the idea of keeping the diversity along the algorithm execution instead of a position resetting of the particles when a change in the environment occurs. That is, not all the particles tend to reach the optimal position, but there are particles that are continuously exploring the search space while other are converging to the peaks. These explorer particles have been implemented in different ways.

In [8] we proposed an exploration mechanism based on heterogeneous swarms that combines attractive and repulsive particles. Repulsive agents keep a formation that allows a continuous exploration of the search space, whereas attractive (PSO) agents collaborate to improve the solution. The mechanism maintains the diversity property allowing the swarm to self-detect the changes of the environment. The exploration mechanism was tested in unimodal dynamic environments.

Charged Particle Swarm Optimization [3] (CPSO) use repulsion between a subset of swarm particles, to avoid the convergence of the whole swarm. Quantum Swarm Optimization [2] (QSO) uses the notion of a cloud of particles, that are randomly positioned around the swarm attractor. Both methods have been tested in a multi-swarm context and QSO has shown a higher performance.

The Collaborative Evolutionary-Swarm Optimization [12] is a hybrid approach that also uses two different sets of particles for preserving the diversity. Specifically, the diversity is maintained using crowding techniques. The performance of CESO is higher than QSO but the mechanism for detecting changes in the environment is the same.

Recently, [6] have proposed MEPSO (Multi-strategy Ensemble Particle Swarm Optimization). MEPSO outperforms other approaches but for unimodal environments where changes have high severity. For detecting changes, MEPSO uses re-evaluation. Moreover, after a change is detected a re-randomization is performed.

The outdated memory problem has been tackled by setting best positions as their current positions or by re-evaluating best positions to detect the changes in the environment (increasing the computation cost) and then resetting the memory of the particles. Most of these existing approaches assume that either the changes are known in advance by the algorithm or that they can be easily detected. These hypotheses are not feasible in many real problems due to the presence of noise and its unpredictable nature.

2.2.1 Multi Quantum Swarm Optimization (mQSO)

Multi Quantum Swarm Optimization (mQSO) is an algorithm proposed for dealing with multi-modal dynamic problems. mQSO divides the swarm in a number of subswarms with the goal of exploiting different promising peaks in parallel. The multiswarm approach, increases the diversity and decreases the probability to finalize the search in a local optimum.
Moreover, each swarm consisted of two different kinds of particles: i) PSO particles that try to reach a better position by following the standard PSO algorithm and ii) quantum particles that orbit around the subswarm attractor within a radius \( r_{cloud} \) in order to keep the diversity along the algorithm execution. Quantum particles address the diversity loss problem.

The position of quantum particles is calculated with the following equation:

\[
\tilde{p}_i \in B_n(r_{cloud})
\]

where \( B_n \) denotes the \( d \)-dimensional ball of the swarm \( n \) centered on the swarm attractor \( \tilde{g}_n \) with radius \( r_{cloud} \).

The idea of mQSO is that each swarm reaches one peak and tracks it along the algorithm execution. To ensure that two swarms are not exploiting the same peak, an exclusion mechanism is proposed as a form of swarm interaction. The exclusion mechanism uses a simple competition rule among swarms that are close (distance less than \( r_{excl} \)) to each other. The winner is the swarm with the best fitness value at its swarm attractor. The loser swarm is expelled and reinitialized in the search space.

When there are more peaks than swarms, not all peaks can be tracked by swarms. Because of the changes in the environment, any local maximum may become a global maximum. Thus, whenever this new optimum is not tracked by any swarm, the performance of the system decreases. To prevent this, an anti-convergence operator is applied whenever all swarms have converged, i.e. when for all swarms the maximum distance found between two particles is less than \( r_{conv} \). At that moment, anti-convergence expels the worst swarm from its peak by reinitializing the particles of the swarm. As a result, there is at least one swarm watching out for new peaks.

mQSO was proved as a good mechanism in multi-peak dynamic environments. The dynamicity is expressed by small changes applied to the peak locations, heighs, and widths.

### 2.3 PSO in Noisy functions

Noisy fitness functions have been addressed by different researches and are a key issue in real-world problems. Different authors have demonstrated that noisy fitness functions are not a handicap for PSO effectiveness in static environments. In 2001, Parsopoulos and Vrahatis studied the behaviour of the PSO when a Gaussian distributed random noise was added to the fitness function. They demonstrated that PSO remained effective in the presence of noise.

In 2005 [15] proposed the Noise-resistant variant, where each particle takes multiple evaluations of the same candidate solution to assess a fitness value. It was demonstrated that noise-resistant PSO showed considerable better performance than the original PSO. The disadvantage of this approach is that in order to improve the confidence of a fitness value, multiple evaluations have to be performed. In [1] the stagnation effect is analyzed for additive and multiplicative noise sources. Bartz-Beielstein et al propose the use of a statistical sequential selection procedure, together with PSO, to improve the accuracy of the function estimation and to reduce the number of evaluations of samples. Moreover, the authors show that the tuning of PSO parameters is not enough to eliminate the influence of noise.

Another proposal for reducing the number of re-evaluations is Partitioned Hierarchical PSO [10]. PH-PSO organizes the neighborhood of the swarm in a dynamic tree hierarchy. This organization allows the reduction of the number of sample evaluations and can be used as a mechanism to detect the changes in noisy and dynamic environments. In PH-PSO, the mechanism used to detect the changes is based on the observation of the changes that occur within the swarm hierarchy.

### 3. PROPOSAL

As we have described in the previous section, there is some evidence that PSO is able to deal with noisy environments but many of the current PSO extensions for dynamic multimodal problems have not been tested in noisy environments.

The main drawback of the current proposals is that they continuously check for changes in the environment. This continuously checking strategy cannot be directly applied when the result of the evaluation of a specific position is subject to noise. The noise provides different values for each evaluation of a given point. These differences are misinterpreted as environment changes causing the continuously resetting of the memory of particles.

A possible solution for dealing with noisy environments is to incorporate a filter that tries to minimize the problem of receiving different fitness values each time a specific position is requested. In addition of the issue of determining the best threshold for the filter for an unknown environment, we will show in the experiments section that the use of noise filters is not the best solution.

An alternative approach is to consider that the trust of a given fitness value degrades with the time. Thus, taking inspiration from ACO algorithms, our proposal is to extend the mQSO algorithm with an evaporation mechanism that will avoid to continuously check up on changes in the environment and, as a second beneficial consequence, will improve the performance of mQSO in noisy fitness evaluation problems.

#### 3.1 Evaporation Mechanism

For solving the outdated memory problem when changes are not known in advance or they cannot be detected due to the noisy fitness function, we hypothesize that providing a mechanism for continuously forgetting is better than a periodical resetting approach and even better than the use of an ad-hoc threshold to filter the noise in the fitness function. Moreover, a continuous mechanism avoids the assumption that changes can be predicted or detected in some way.

We propose an evaporation mechanism for reducing the fitness value of the best position found by each particle along time. This mechanism will penalize optima that were visited a long time ago. Thus, evaporation provides an automatic dissipation mechanism over the information taking into account the acquisition time. The idea of evaporation is not new. ACO systems use evaporation in pheromone trails as a mechanism to achieve a signal degradation [9] and self-adapt to environment changes. Moreover in [7], some preliminary results were shown using an evaporation factor in unimodal dynamic environments.

Different approaches can be used as evaporation factors. The main approaches are the use of either a subtractive or a multiplicative factor. A multiplicative factor will decrease, at each particle iteration, the fitness value in a constant factor.
\( \nu \) following the equation:

\[
s_{ni} = s_{ni} - \nu
\]  

(4)

where given a particle \( i \) belonging to the swarm \( n \), the fitness value of the best position found \( \vec{p}_{ni} \) is stored at \( s_{ni} \) (best solution).

A multiplicative factor will decrease the fitness value by multiplying it with a constant \( \alpha \) following the next equation:

\[
s_{ni} = s_{ni} \times \alpha
\]  

(5)

where \( \alpha \) is an evaporation factor such that \( \alpha \in (0, 1) \) and \( \times \) is the multiplier operator.

We will see in the experiments that with the subtractive factor \( \text{mQSO} \) achieves lower errors. Nevertheless, multiplicative factors can be used achieving also good performance results. Then, we incorporated a subtractive factor \( \nu \). Specifically, in each algorithm step, each particle evaluates the fitness of its current position and updates \( s_{ni} \) to the current fitness if the current fitness is higher than \( s_{ni} \), that is:

\[
\text{if } (f_u(\vec{p}_{ni}) > s_{ni}) \text{ then } \begin{align*}
    & s_{ni} = f_u(\vec{p}_{ni}); \\
    & \vec{b}_{ni} = \vec{p}_{ni}; \\
\end{align*}
\]

\[
\text{else } \begin{align*}
    & s_{ni} = s_{ni} - \nu
\end{align*}
\]

(6)

where \( f_u \) is de fitness function and \( \vec{p}_{ni} \) is the current position of the particle \( i \) of swarm \( n \).

### 3.2 mQSOE

mQSO has been extended with the evaporation mechanism. The main advantages of this extension (mQSOE) are twofold: i) we avoid the false change detection produced by the noisy fitness function; and ii) if we don’t have to check for environment changes, we are saving evaluations of the fitness function. The savings can be used to improve the algorithm performance.

The modification of the mQSO algorithm is simple: the test for change step was eliminated and the evaporation equation (4) was added when updating the particle and swarm attractors.

In Figure 1 we can see the mQSO algorithm extended with the evaporation mechanism. At the beginning, the particles of all swarms are randomly initialized. Given a particle \( i \) of a swarm \( n \), \( \vec{v}_{ni} \) is the velocity of the particle, \( \vec{p}_{ni} \) is the position of the particle, \( \vec{b}_{ni} \) is the position of the best fitness found by the particle, and \( s_{ni} \) is the fitness value of the best position \( \vec{b}_{ni} \). Once, the particles are initialized, the best positions \( \vec{g}_n \) and fitness values \( s_n \) of the swarms are calculated.

In the main loop, the first step is to apply the anti-convergence operator when all swarms have converged. A swarm has converged when the maximum distance found between its particles is lower than the \( r_{conv} \) parameter. Anti-convergence operator marks the worst swarm to be reseted after. As we said before, this operator ensures that at least one swarm is ever exploring the search space to detect new peaks or low peaks that could become important.

Next to the anti-convergence operator, the exclusion operator is applied. This operator detects when two swarms...
have converged to a close position and then marks the worst of them to be reseted. Two swarms are too close when the distance between their attractors is lower than $r_{exc}$.

After applying all operators, the particles are moved. The particles that belong to a swarm marked to be reseted will be reinitialized with random positions an velocities. The rest of the particles will be updated according to the equations defined for their type: PSO particles will be updated by following the standard PSO equations (1) and (2) while QSO particles will be randomized according (3).

Finally, after updating the particle’s position, the best position found by the particle $b_{ni}$ and the best fitness found $s_{ni}$ is also updated. The evaporation mechanism is applied to this step according to (6).

## 4. EXPERIMENTS

The goal of the experiments is to analyze the effect of the evaporation mechanism by comparing with the standard mQSO algorithm. Previously to compare both algorithms, we have performed experiments for determining the best value for the subtraction factor $\nu$; we have compared the performance of using a subtractive or a multiplicative factor; and we have analyzed the independence of the evaporation factor with respect to the peak heights and the peak shifts (severity).

To evaluate the performance of the evaporation mechanism and to compare mQSOE with mQSO, we have used the Moving Peaks Benchmark (MPB) [5]. MPB is a benchmark developed to compare dynamic optimization algorithms by modeling problems less complex than the real world but more complex than a simple simulation. MPB allows the design of search spaces that change over time (in the height, width and location of peaks).

### 4.1 Experimental framework

The settings for the MPB parameters correspond to MPB’s scenario 2 with 10 peaks. Specifically, we used a 5-dimensional search space with dimension ranges from 0 to 100. The peak function used is a cone and the maximum distance that one peak moves is 1. Table 1 summarizes all the MPB settings. A run consisted of 100 peak changes in a random direction (correlation coefficient = 0). All peaks changed every 5000 steps (function evaluations) their height (height $\in [30, 70]$), width (width $\in [1, 12]$), and position (vLenght = 1). Results are based on averages over 50 runs with uncorrelated peak changes.

The performance of the algorithms has been assessed by using the offline error measure. Offline error is the average over, at every point in time, the error of the best solution found since the last change of the environment. This measure is always greater or equal to zero and would be zero for perfect tracking.

In order to aggregate noise to the fitness function, we modified MPB such as the fitness function incorporated a noise factor $\gamma$ in the following way:

$$fitness(\vec{p}) = MPBfitness(\vec{p}) + \frac{2 \ast \theta - 1}{2} \ast \gamma$$

where $\theta$ generates a uniform random number between [0..1] and $\gamma$ varies between 0 and 30 depending on the experiment.

The parameters of mQSO and mQSOE are the same. Except the evaporation factor $\nu$ that only exists in mQSOE.

We used 100 particles grouped in 10 different swarms. Each swarm has 10 particles, 5 PSO and 5 QSO. The parameters $r_{exc}$ and $r_{conv}$ were set to 31.5 following the author’s recommendations [4].

The standard PSO parameters have been settled following [2], $\chi = 0.729843788$, $\phi_1 = 2.05$ and $\phi_2 = 2.05$.

### 4.2 Determining the evaporation factor

The first experiments conducted aimed at determining the best evaporation factor $\nu$ and to analyze the dependency of this factor with respect to the noise level $\gamma$. We performed experiments by varying $\gamma$ from 0 to 30 and varying $\nu$ from 0 to 30.

The best results were achieved for low values of $\nu$. Thus, we will concentrate our analysis for values of $\nu$ from 0 to 4. Figure 2 details the offline error when varying the noise level and the evaporation factor. The vertical axis represents the offline error, and the horizontal axis the noise level.

First, as it was predictable, the error increases when the noise increases. Second, the best performance is achieved with low evaporation values (lower than 1) but with a minimum evaporation (values like 0.1 behave also bad).

Our decision was to choose $\nu = 0.5$. This decision was motivated by two main reasons: it presents a good performance (best performances are achieved for $\nu \in [0.4, 0.9]$) and it presents the smoothest behavior.

Moreover, the experiments show that all values of $\nu$ follow the same behavior when the noise level increases. Thus, the
best evaporation value is not changing with the noise level, i.e. the evaporation factor is independent of the noise level.

4.3 Subtraction versus multiplication as evaporation operator

The goal of these experiments was to compare the performance of a subtraction operator with the performance of multiplier operator. Specifically, we compared the best result achieved in the previous experiments ($\nu = 0.5$) with mQSOE where the evaporation mechanism was substituted by the equation (5). We performed experiments by varying $\alpha$ from 0 to 1.

Figure 3 compares the subtractive behavior with the best multiplicative factors. To achieve errors similar to the subtractive operator, the values of $\alpha$ must be close to 1. The best performance of a multiplicative factor is achieved with $\alpha = 0.989$. Specifically, the results are similar to the best subtractive operator but the subtractive has a smoother behavior and achieves lower errors.

Nevertheless, as a conclusion we have to say that both operators can be used as evaporation mechanisms because the performance difference is not significant.

4.4 Independence of peak heights and peak shifts

The goal of these experiments is to analyze the dependency of the evaporation factor with respect to the heights of the peaks and the distance of the peak movements (severity). All the previous experiments were conducted with peak heights from 30 to 70 and with a severity of 1.

We analyzed the performance of the evaporation factor with lower peak heights. Specifically we repeated the experiments with $\text{height} \in [10..30]$. Figure 4 shows the results obtained. Notice that the offline error is lower because the worst error is now also lower. Also, the offline error difference among the different factors has decreased. Nevertheless, $\nu = 0.5$ remains as the best value indicating that the evaporation factor is not dependent of the peak heights.

To compare the performance of mQSOE with reported results of mQSO, we set the severity parameter to 1. Nevertheless, we conducted experiments varying the severity parameter to analyze the performance of mQSOE.

Since we have previously analyzed the performance of the evaporation factor with different noise levels, in these experiments the noise level was set to zero. Figure 5 shows that the error is increasing when the severity increases but also that it affects all with the same proportion. Thus, $\nu = 0.5$ presents also a competitive behavior and the experiments corroborate the independency of the evaporation factor with respect of the severity.

4.5 mQSO versus mQSOE

Finally, we are ready to compare the offline error of mQSO versus our mQSOE. The goal of the experiment is to study the behaviour of both algorithms regarding the increment of noise in the evaluation of the fitness function. As in the previous experiments, $\gamma$ varied from 0 (no noise) to 30 (approx. 23% of noise).

Table 2 shows that when the noise level increases, the performance of both algorithms decreases (as expected). More-

Figure 3: Multiplicative versus subtractive evaporation

Figure 4: Evaporation performance when decreasing the peak height

Figure 5: mQSOE performance when changing severity
Table 2: Average of offline error (± Std. Dev.) achieved by mQSO and mQSOE introducing different noise levels

<table>
<thead>
<tr>
<th>γ</th>
<th>mQSO</th>
<th>mQSOE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.206 ± 0.529</td>
<td>2.362 ± 0.620</td>
</tr>
<tr>
<td>3</td>
<td>3.397 ± 0.727</td>
<td>2.763 ± 0.600</td>
</tr>
<tr>
<td>6</td>
<td>3.581 ± 0.640</td>
<td>2.993 ± 0.562</td>
</tr>
<tr>
<td>9</td>
<td>4.107 ± 0.526</td>
<td>3.156 ± 0.554</td>
</tr>
<tr>
<td>12</td>
<td>4.468 ± 0.475</td>
<td>3.125 ± 0.472</td>
</tr>
<tr>
<td>15</td>
<td>5.094 ± 0.765</td>
<td>3.326 ± 0.661</td>
</tr>
<tr>
<td>18</td>
<td>5.397 ± 0.652</td>
<td>3.427 ± 0.534</td>
</tr>
<tr>
<td>21</td>
<td>6.090 ± 0.670</td>
<td>3.568 ± 0.514</td>
</tr>
<tr>
<td>24</td>
<td>6.603 ± 0.822</td>
<td>3.683 ± 0.668</td>
</tr>
<tr>
<td>27</td>
<td>6.985 ± 0.618</td>
<td>3.713 ± 0.564</td>
</tr>
<tr>
<td>30</td>
<td>7.609 ± 1.015</td>
<td>3.923 ± 0.419</td>
</tr>
</tbody>
</table>

over, in presence of noise mQSOE demonstrates better performance than the original mQSO. However, when in the absence of noise mQSO reaches a better performance than using evaporation. The reason is that environment changes are instantly detected by mQSO because it is constantly re-evaluating the swarm attractors. However the evaporation mechanism is a little bit slower but more robust in presence of noise.

The bad performance of mQSO with noisy fitness functions is produced because the algorithm confuses the noise with changes in the environment and, consequently, it is continuously re-initializing the particle’s memory. That is, at each algorithm iteration the best position found ($\vec{b}_n$) is changed to the current position degrading the search mechanism.

### 4.6 Filtering noise in mQSO

In order to avoid the continuously resetting in mQSO, we experimented with the incorporation of a noise filter into the original mQSO. Specifically, we used a threshold value $\beta$ to determine whether a change observed in the environment is effectively considered a real change. Then, a change in the environment is only considered when the difference between the stored fitness value and the current evaluation is higher than this threshold. Thus, we add the condition (8) in the function that checks for a change.

```plaintext
if $\beta < |s_{ni} - f_u(\vec{p}_{ni})|$ then
  | changeProduced = true
end
else
  | changeProduced = false
end
```

where $\beta$ is the ad-hoc threshold that controls the tolerated noise.

In this way, the goal of this experiment is to analyze the performance of mQSO modified with (8) to avoid the continuously resetting. Two different values for the threshold were analyzed: $\beta = 15$ and $\beta = 30$. The first value is a threshold that corresponds to the middle noise level. The second one is the highest noise level.

Figure 6 shows how the use of a noise threshold ($\beta > 0$) improves the performance of the original mQSO in presence of noise. However, noise filters produce a lower performance when the fitness function is not subjected to noise. This decrement is caused because the algorithm confuses small environment changes with noise.

It can be observed that mQSO with $\beta = 15$ presents the best performance when $\gamma = 15$. Nevertheless, this behavior is not robust and decays for other values of $\gamma$. This result is not surprising because when $\beta$ and $\gamma$ share the same value, the noise threshold is correlated with the error in the fitness evaluation. Then, when $\beta$ and $\gamma$ are 15, we ensure that the noise is never confused with a change in the environment. Nevertheless, as Figure 6 shows, for $\gamma$ values different than $\beta$ the offline error grows. The reason is because when $\gamma$ is lower than $\beta$ the noise is confused with environment changes and produces a memory resetting when it is not necessary. On the other hand, when $\gamma$ is higher than $\beta$, it increases the probability of not detecting the changes in the environment, preventing the peak tracking.

For $\beta = 30$, environment changes are treated as noise. Thus, most of the changes in the environment are not detected and, consequently, the memory of the particles is not re-initialized. As a consequence, swarms are not able to track the peaks.

Comparing the performance of mQSO with noise filters and mQSOE, mQSOE presents a more robust behavior. mQSOE is not achieving the best performance at all the different noise levels: without noise the standard mQSO presents the lowest error; and mQSO extended with a noise filter presents the best performance when the noise level in the fitness function is the same that the noise threshold at mQSO. In these specific cases, the performance of mQSOE is ever close to the best error.

Moreover, the noise threshold cannot be fixed because in real problems the noise level is never static.

Summarizing, mQSOE is demonstrated as a good solution for dealing with dynamic environments where the evaluation of the fitness function is subject to noise. Moreover, the evaporation mechanism does not require an explicit checking of changes.
5. CONCLUSIONS

In this paper we have proposed an evaporation mechanism for dealing with optimization problems when the problems change over time (dynamic environments) and the evaluation of the fitness function is subject to noise.

We have incorporated the evaporation mechanism as an extension of mQSO, a well known algorithm for dynamic multimodal problems. Nevertheless, the solution proposed in this research is general and can be incorporated to other existing algorithms that detect the changes in the environment by re-evaluating swarm attractors.

We have shown that the evaporation mechanism is robust to different simulation conditions such as the peak heights, the severity, and the level of noise. Furthermore, we reported experiments that introduce the possibility of using either subtractive or multiplicative evaporation factors.

Experiments have shown that the performance of the proposed mQSOE extension outperforms the standard mQSO when the fitness evaluation is noisy. Moreover, we have shown that the detection of changes when noise is present is not a trivial issue. Specifically, we have shown how the introduction of noise filters is not able to outperform mQSOE.

Moreover, the evaporation mechanism has an additional benefit: the effort avoided in re-evaluating swarm attractors can be used to achieve a better performance by spending them on the swarm’s convergence.

As future work, we plan to analyze the performance of the evaporation mechanism to other existing PSO benchmarks and to a real peak detection problem where a collection of sensors are continuously scanning the changes in a specific marine area.

Acknowledgments

This work was funded by IEA (TIN2006-15662-C02-01), Agreement Technologies (CONSOLIDER CSD2007-0022, INGENIO 2010), and ANERIS (CSIC-PIF08-15-2) projects. The first author holds a FPI scholarship from the Spanish Government.

6. REFERENCES


