Adapting Particle Swarm Optimization in Dynamic and Noisy Environments

Jose Luis Fernandez-Marquez and Josep Lluis Arcos

Abstract—The optimisation in dynamic and noisy environments brings closer real-world optimisation. One interesting proposal to adapt the PSO for working in dynamic and noisy environments was the incorporation of an evaporation mechanism. The evaporation mechanism avoids the detection of environment changes, providing a continuous adaptation to the environment changes and reducing the effect when the fitness function is subject to noise. However, its performance decreases when the fitness function is not subjected to noise (with respect to methods that use environment change detection). In this paper we propose a new dynamic evaporation policy to adapt the PSO algorithm to dynamic and noisy environments. Our approach improves the performance when the fitness function is dynamic and not subject to noise. It also keeps a similar performance when the fitness function is subject to noise.

I. INTRODUCTION

Nowadays, Real-World optimisation is still a challenge because it involves uncertainty issues that must be considered [1]. These uncertainty issues may arise from several sources: the evaluation of the fitness function may be subjected to noise; the design variables may be subjected to perturbations; the fitness function may be only approximated; or the optima of the problem may change over time. This paper deals with two of these issues: The optima of the problem change over time and the fitness function is subjected to noise.

The presence of noise in the fitness function involves the problem of guiding the search with uncertainty information, i.e. the decision taken by one entity during the optimisation process could be misguided due to the noise in the fitness value. Evolutionary and Swarm approaches have reached very good results in noisy optimisation because the decision of one entity does not damage the global behavior of the optimisation algorithm (e.g. the movement of a particle, movement of an ant, or the acceptance of one individual in GA selection process). On the other hand when dealing with dynamic environments, the optimization algorithm must ensure the diversity in order to continuously find and converge to new optima positions.

Particle Swarm Optimisation algorithm (PSO) has demonstrated that it is an efficient algorithm for solving static function optimisation. Even when the fitness function is subjected to noise, PSO has demonstrated that the noise is not a handicap for PSO effectiveness in static environments [2]. Different PSO extensions have been proposed to deal with dynamic optimisation problems. Most of these extensions are based on keeping the diversity through the optimisation to improve the adaptability whenever environment changes occur.

PSO extensions dealing with dynamic environments have improved the performance of the PSO approach. However, the dynamic environment optimisation is still a challenging problem. Moreover, when the environment is dynamic and noisy a major problem appears: The noise may be misinterpreted as environment changes or the environment changes may be considered noise, as presented [1]. In [1] they tackled this problem and proposed a mechanism for providing continuous adaptation when an environment change occurs and the fitness function is subjected to noise.

Their mechanism, called the evaporation mechanism, avoids the change detection. Moreover, it was designed to improve the performance in terms of convergence speed and the standard of the optimisation, in the presence of noise. They proposed the multi Quantum Swarm Optimization Evaporation (mQSOE) algorithm as an extension of the multi Quantum Swarm Optimization (mQSO) [3] that introduces two main features: (1) A mechanism to keep the diversity through the run, i.e. the mQSOE algorithm is able to adapt itself when environment changes occur without the resetting of particles positions; and (2) an evaporation mechanism for adapting the particles’ memories to the environment changes without their resetting. mQSOE has demonstrated that it improves performance in noisy environments, but at the cost of reducing the performance when the fitness function is not subjected to noise.

In this paper we propose a new dynamic evaporation approach that improves the performance of mQSOE in dynamic and noisy environments and moreover, reduces the decrement of performance when the fitness function is not subject to noise.

This paper is organised as follows: The following section presents the background: PSO basics and the extension of PSO for improving its performance in dynamic and noisy environments. Next, we present our approach and experimental results. Finally, conclusions and future work are discussed.

II. BACKGROUND

The background description is organised around different issues: firstly the standard PSO is introduced; secondly we present a summary about the different extensions of PSO that are used to deal with dynamic environments; thirdly PSO approaches in noisy functions are introduced; and finally, we
present an existing extension of PSO to solve dynamic and noise optimisation problems.

A. Particle Swarm Optimization

PSO was first proposed in 1995 [6] modeled on Swarm Intelligence. The initial ideas of PSO were essentially aimed at producing computational intelligence by exploiting simple analogues of social interaction, rather than purely individual cognitive abilities [4]. In this population based algorithm, the particles represent solutions that are able to move over the search space. The movements of each of these particles are based on a combination of both cognitive and social parts. The cognitive part drives each particle to its best position at that moment in time. The social part drives each particle to the best position found by particles belonging to its neighborhood. 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 with its best position found \( \vec{b}_i \), with social information and as with the best position of its neighbors, \( \vec{g} \). At each algorithm iteration, particles evaluate the objective function at its current location (fitness value) \( f_e(\vec{p}_i) \).

Since 1995, when James Kennedy and Eberhart first proposed the PSO algorithm, some extensions and optimisations of their parameters have been realised [4]. 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:

\[
\vec{v}_i = \chi (\vec{v}_i + \vec{U}(0, \phi_1)(\vec{b}_i - \vec{p}_i) + \vec{U}(0, \phi_2)(\vec{g} - \vec{p}_i)) \tag{1}
\]

\[
\vec{p}_i = \vec{p}_i + \vec{v}_i \tag{2}
\]

\[
\chi = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}} \tag{3}
\]

Where:

- \( \phi = \phi_1 + \phi_2 > 4 \)
- \( \chi \) is the constant multiplier that ensures the convergence, see equation 3;
- \( \vec{p}_i \) 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
- \( \vec{U}(0, \phi_1) \) represents a vector of random numbers uniformly distributed in \([0, \phi_1]\).

Equation 1 has 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.

B. PSO in Dynamic Environments

Optimisation in dynamic environments is a challenging problem for PSO. In essence, PSO presents two problems when it deals with dynamic environments [5]: (1) the Diversity Loss Problem and (2) the Outdated Memory Problem. The diversity loss problem, common in evolutionary optimisation approaches, occurs when the particles reach the convergence and then it is not possible to find a new good solution, due to the lack of diversity. 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 particles search.

Different extensions of PSO have been proposed to improve its adaptiveness in dynamic environments. Extensions (see [4] for an overview) propose solutions such as resetting the positions of particles frequently or using a multi-swarm model.

The diversity loss problem has been addressed either by introducing randomisation, repulsion, dynamic networks or multi-populations [5]. 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 others are converging to the peaks. These explorer particles have been implemented in different ways.

In [6] they proposed an exploration mechanism based on heterogeneous swarms that combines attractive and repulsive particles. Repulsive particles keep a formation that allows a continuous exploration of the search space, whereas attractive (PSO) particles 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 uni-modal dynamic environments.

Charged Particle Swarm Optimisation [7] (CPSO) uses repulsion between a subset of particles, to avoid the convergence of the whole swarm. Multi Quantum Swarm Optimisation [5] (mQSO) 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 mQSO has shown a higher performance.

The Collaborative Evolutionary-Swarm Optimisation [8] is a hybrid approach that also uses two different sets of particles for preserving the diversity. More specifically, the diversity is maintained using crowding techniques. The performance of CESO is higher than mQSO but the mechanism for detecting changes in the environment is the same.

Recently, [9] have proposed MEPSO (Multi-strategy Ensemble Particle Swarm Optimization). MEPSO outperforms other approaches but only for unimodal environments where changes have high severity. For detecting changes, MEPSO uses re-evaluation. Moreover, after a change is detected a re-randomisation 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.

**Multi Quantum Swarm Optimisation (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 of finalising the search in a local optimum.

Moreover, each swarm consisted of two different kind 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 $r_{\text{cloud}}$ 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:

$$\vec{p}_i \in B_n(r_{\text{cloud}})$$

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

The idea of mQSO is that each swarm reaches one peak and tracks the peak 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 $r_{\text{excld}}$) to each other. The winner is the swarm with the best fitness value at its swarm attractor. The loser swarm is expelled and reinitialised in the search space.

When there are more peaks than swarms, some peaks will not be tracked by swarms. Thus, because of the changes in the environment may be produced, any local maximum may become a new global maximum and 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 max distance found is less than $r_{\text{conv}}$. At that moment, anti-convergence expels the worst swarm from its peak by reinitialising the particles of the swarm. As a result, there is at least one swarm continuously looking for for new peaks.

**C. PSO in Noisy Environments**

The study of noisy environments is a key issue because of the imprecision associated to any measurement in real-world problems. Noisy environments can be modeled by adding a noise factor to the fitness function. Different authors have demonstrated that noisy fitness functions are not a handicap for PSO effectiveness in static environments. Parsopoulos and Vrahatis [2] specifically studied the behavior 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 [10] a Noise-resistant variant is proposed, where each particle takes multiple evaluations of the same candidate solution to assess a fitness value. It was demonstrated that noise-resistant PSO showed considerably 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 [11] the stagnation effect is analysed 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 [12]. PH-PSO organises the neighborhood of the swarm in a dynamic tree hierarchy. This organisation 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.

**D. PSO in Dynamic and Noisy Environments**

Nowadays, dynamic and noisy environments is one of the most important challenges for the optimisation algorithms, due to its importance in real-world optimisation problems. The optimisation in dynamic and noisy environments had not been tackled by any PSO extension until [1]. In these optimisation problems, the environment change detection can not be used for the particle’s memory update, because the noise is misinterpreted as environment changes. Neither can it be solved with the use of threshold values, to filter the noise and avoid this mistake [1].

In mQSOE algorithm, the evaporation mechanism is proposed as a way to reduce the fitness value of the best position found by each particle along time, i.e. to penalise optima that were visited a long time ago. Evaporation was applied in two ways: (1) by a subtractive term or, (2) by a multiplicative factor. The subtractive term decreases at each particle iteration, the fitness value in a constant term $\nu$ following the equation:

$$s_{ni} = s_{ni} - \nu$$

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 decreases the fitness value by multiplying it with a constant $\alpha$ following the next equation:

$$s_{ni} = s_{ni} \times \alpha$$

$$s_{ni} = s_{ni} - \nu$$

$$s_{ni} = s_{ni} \times \alpha$$
where $\alpha$ is an evaporation factor such that $\alpha \in (0, 1)$ and $\times$ is the multiplier operator.

On one hand, the evaporation term produces a very slow adaptation when the fitness values are too high and, on the other hand, both approaches provide a constant evaporation for all the particles without taking into account the environment or particle state. Both approaches have demonstrated in getting a better performance when the fitness function is subjected to noise and the environment is dynamic. However, when noise is not applied to the fitness function the evaporation mechanism decreases the performance of PSO (comparing with using environment change detection).

III. OUR APPROACH

Our goal is to propose a new evaporation mechanism to improve mQSOE in both scenarios (in noise free scenarios and noisy scenarios). Studying the behavior of mQSOE, we observed that a high evaporation factor produces a fast adaptation (fast response after environment changes occur), but the particles can not reach a good solution in the optimisation process (bad convergence), because the particles may not use their cognitive part. That is, the particles forget the best position found very fast and are not capable of reaching good solutions.

Contrarily, a low evaporation factor achieves a fast and effective convergence, but the adaptation is very low. Figure 1 exposes the slow adaptation observed in the mQSOE-mult and mQSOE-sub compared with mQSO using environment change detection. We observe that, even when both approaches assess the same performance, the delay introduced by evaporation produces negative consequences when the frequency of changes increases.

To solve this problem, we propose a dynamic evaporation factor using only the particle’s local information, i.e. its velocity and the difference between the fitness of its best position found and the fitness of its current position (i.e a gradient-based policy). The particle’s velocity presents the following characteristics: (1) the velocity module is very high before converging, and (2) the velocity module decreases when the particles are close to the convergence point. On the other hand, when particles have converged and an environment change occurs, the gradient becomes high. Thus, the intuition is that the evaporation factor must be high when the velocity module is slow (indication of convergence) and the gradient is high. Moreover, the evaporation factor must be low when the particle’s velocity is high, helping the swarm convergence. Specifically, dynamic evaporation is implemented using the following equations:

\[
s_{ni} = s_{ni} \times \alpha_{dyn}
\]

\[
\alpha_{dyn} = 1 - \frac{(s_{ni} - \text{fitness}(\vec{p}_{ni})) + \left|V_{ni}\right|}{2\left|V_{max}\right|}
\]

where $\alpha_{dyn}$ is the dynamic evaporation factor, $s_{ni}$ the fitness value of the best position found, $s_{max}$ the maximum fitness reachable, $\left|V_{ni}\right|$ is the velocity module of the particle, $\left|V_{max}\right|$ the maximum velocity module of the particles. Both maxima are used to normalise the evaporation factor. Notice that when a new best position is found, the evaporation is not applied until next iteration, (see algorithm 2).

In the same way than mQSOE, our approach extends the original mQSO adding the evaporation mechanism. The modification of the mQSO algorithm is simple: the test for change was eliminated and the evaporation equation (5) was added when updating the particle’s memory.

Figure 2 summarises the mQSO algorithm extended with the evaporation mechanism. At the beginning, the particles of all swarms are randomly initialised. 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 initialised, the best positions $\vec{g}_n$ and fitness values $s_n$ of the swarms are calculated.

In the main loop, the first step applies the anti-convergence operator when all swarms have converged. A swarm converges when the maximum distance found between its particles is lower than the $r_{conv}$ parameter. Anti-convergence operator marks the worst swarm to be reset. As explained above, 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 resetted. Two swarms are too close when the distance between their attractors is lower than $r_{excl}$.

After applying all operators, the particles are moved. The particles belonging to a swarm marked to be reset are reinitialised with random positions and velocities. The rest of the particles are updated according to the equations defined by their type: PSO particles are updated using the standard PSO equations (1) and (2) while quantum particles are randomised.

Finally, after updating the particle’s position, the best position found by the particle ($\vec{b}_{ni}$) and the best fitness found ($s_{ni}$) are updated. The evaporation mechanism is applied at this step according to (7) and (8).
// Initialization
foreach particle ni do
  Randomly initialize \( \vec{v}_{ni}, \vec{p}_{ni} \);
  \( \vec{b}_{ni} = \vec{p}_{ni} \);
  \( s_{ni} = f_u(\vec{p}_{ni}) \);
end
foreach swarm n do
  \( \vec{g}_n = \arg\max\{f_u(\vec{p}_{ni})\} \)
  \( s_n = \max\{s_{ni}\} \)
  // Marker for randomization
  init[n] = false
end
repeat
  // Anti-convergence
  if all swarm have converged then
    // Remember to randomize worst swarm
    init[worst swarm] = true;
  end
  // Exclusion
  foreach pair of swams n, m do
    if swarm attractor \( \vec{g}_n \) is within \( r_{extr} \) of \( \vec{g}_m \) then
      if \( s_n \leq s_m \) then
        init[n] = true;
      else
        init[m] = true;
      end
    end
  end
  foreach swarm n do
    foreach particle i of swarm n do
      if init[n] then
        Randomize particle i
      else
        // Update particle
        Apply equations (1) (4) depending on particle type.
        if \( f_u(\vec{p}_{ni}) > s_{ni} \) then
          \( s_{ni} = f_u(\vec{p}_{ni}) \)
          \( \vec{b}_{ni} = \vec{p}_{ni} \)
        else
          // Apply evaporation factor
          Equations (7), (8)
        end
        // Update attractor
        \( \vec{g}_n = \vec{b}_{ni} \) such as \( i \) has max\( \{s_{ni}\} \)
      end
    end
  end
until number of function evaluations

Fig. 2. mQSODE algorithm (mQSO extended with the evaporation mechanism)

TABLE I

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Small Step</td>
</tr>
<tr>
<td>T2</td>
<td>Large Step</td>
</tr>
<tr>
<td>T3</td>
<td>Random</td>
</tr>
<tr>
<td>T4</td>
<td>Chaotic</td>
</tr>
<tr>
<td>T5</td>
<td>Recurrent</td>
</tr>
<tr>
<td>T6</td>
<td>Recurrent with noise</td>
</tr>
</tbody>
</table>

TABLE II

<table>
<thead>
<tr>
<th>Parameter</th>
<th>values</th>
<th>Parameter</th>
<th>values</th>
</tr>
</thead>
<tbody>
<tr>
<td>movrandom</td>
<td>random</td>
<td>num. of peak</td>
<td>10</td>
</tr>
<tr>
<td>num. of dimensions</td>
<td>10</td>
<td>minheight</td>
<td>10</td>
</tr>
<tr>
<td>maxheight</td>
<td>100</td>
<td>stdheight</td>
<td>50</td>
</tr>
<tr>
<td>minwidth</td>
<td>1</td>
<td>maxwidth</td>
<td>10</td>
</tr>
<tr>
<td>eval per change</td>
<td>10000</td>
<td>mincoodinate</td>
<td>0</td>
</tr>
<tr>
<td>maxcoordinate</td>
<td>100</td>
<td>Optimization type</td>
<td>MAX</td>
</tr>
<tr>
<td>alpha</td>
<td>0.04</td>
<td>max alpha</td>
<td>0.1</td>
</tr>
</tbody>
</table>

IV. EXPERIMENTS

Mainly, in the experimental section we have two goals: to compare the performance of our approach versus the two evaporation methods proposed in [1] and to compare our approach versus the original mQSO in presence of noise and without noise.

A. Experimental Framework

In order to simulate the optimisation problem in dynamic and noisy environments, we used the Generalized Dynamic Benchmark Generator (GDBG) proposed in [13]. GDBG provides different functions that change along time in different ways. The types of changes are: small steps, large steps, random, chaotic, recurrent, and recurrent with noise. Table I shows the different types of changes used in the experiments.

From the functions provided by GDBG, we chose the rotation peak function in order to simulate a similar environment that was used in [1]. However, some constraints proposed in GDBG benchmark are assumed: a higher number of dimensions and a higher number of evaluations between changes. Table II summarises parameter settings for the GDBG benchmark.

All results are averaged over 60 runs. For the mQSO-mult we have used the multiplicative factor \( \alpha = 0.989 \) and for the mQSOE-sub the subtractive factor \( \nu = 0.5 \). Both of them are proposed in [1].

Based on the metrics defined in GDBG, the performance is calculated using the following criteria:

\[
\text{avg}_{\text{max}} = \sum_{i=1}^{\text{runs}} \frac{\sum_{j=1}^{n_c} f_{\text{last}}^{e_{i,j}} T}{\text{runs}} \quad (9)
\]

\[
\text{avg}_{\text{mean}} = \sum_{i=1}^{\text{runs}} \frac{\sum_{j=1}^{n_c} f_{\text{last}}^{e_{i,j}} T}{\text{runs}} \quad (10)
\]

\[
\text{avg}_{\text{min}} = \sum_{i=1}^{\text{runs}} \frac{\sum_{j=1}^{n_c} f_{\text{last}}^{e_{i,j}} T}{\text{runs}} \quad (11)
\]
\[\text{std} = \sqrt{\frac{1}{\text{runs} \times n_c - 1} \sum_{i=1}^{\text{runs}} \sum_{j=1}^{n_c} (E_{i,j}^{\text{last}}(t) - \text{avg}_{\text{mean}})^2}\]  

where \(E_{i,j}^{\text{last}}(t) = |f(x_{\text{best}}^i(t)) - f(x^*(t))|\) is the absolute error after reaching the maximum functions evaluation, i.e. just before an environment change, \(f(x^*(t))\) is the global optima at time \(t\), and \(n_c\) the number of changes.

These metrics provide information of how the tested algorithm behaves in the worst, normal, and best cases. Specifically, Avg\_min (11) measures the average of the best convergences reached, avg\_max (9) measures the average of the worst convergences and avg\_mean (10) measures the average of all convergences.

We used 100 particles grouped in 10 different swarms. Each swarm has 10 particles, 5 PSO and 5 QSO. The parameters \(r_{\text{ excl}}\) and \(r_{\text{ conv}}\) were set to 31.5 following the author's recommendations [3]. The standard PSO parameters have been settled following [5], \(\chi = 0.729843788\), \(\phi_1 = 2.05\), and \(\phi_2 = 2.05\).

B. Performance when the fitness function is noise-free

In these first experiments, the goal was to demonstrate the higher performance of proposed mQSODE when the fitness function is not subjected to noise. As it was presented, the main problem of using evaporation is the decrease of performance in this situation.

Figure 3 shows the comparison between existing evaporation mechanisms, the original mQSO and our new approach (mQSODE). The results are calculated with the small step change type. The horizontal axis represents the simulation steps and the vertical axis represents the relative error averaged over 60 runs. Constant-based evaporation methods (mQSOE-mult and mQSOE-sub) present a delay after the environment changes, due to the slow evaporation process, as we presented in the previous section. The dynamic evaporation mechanism is able to act after a change with similar speed than mQSO (that is detecting environment changes). The faster reaction of our approach when environment changes occur allows a better convergence. Notice in Figure 3 that if the system increases the frequency of changes, mQSOE-sub and mQSOE-mult cannot reach the convergence due to the slow reaction when an environment change occurs. However, mQSODE and mQSO may converge and present a better performance when the frequency of changes increases.

Figure 4 summarises average maxima (the average of worst convergences) for each algorithm when different change types are applied (types of changes are described in Table I). mQSODE presents a better performance than the constant-based evaporation mechanisms and the standard mQSO.

But mQSODE is not only presenting the best performance in the worst cases, it is also the best according to average means, i.e. the average of all convergences realised after changes, for the majority of change types. Figure 5 presents the results mQSODE for average means. The performance of mQSODE is achieved because its faster reaction to environment changes by adjusting the evaporation factor depending of the local state of the particle.

Figure 6 represents the average of the best cases. Regarding this measure, the original mQSO presents a better performance than our approach, i.e., even though mQSODE presents a better performance than mQSO in average, mQSO is able to reach some best results. However, the behaviour of mQSO is not as robust as the mQSODE. Figure 7 exposes how the mQSO algorithm presents a higher standard deviation than mQSODE.

As a conclusion, in noise free environments the dynamic evaporation mechanism improves the existing algorithms (mQSOE-sub and mQSOE-mult). Moreover, the problem reported for those algorithms, i.e. the decreasing of performance when the fitness function is not subjected to noise, has been solved for the new approach. mQSODE presents similar results than the standard mQSO, that does not use the evaporation, for two reasons: a close reaction time when an environment change occurs, and the fact that the mQSO uses fitness function evaluations that the mQSODE can use in the optimisation process.
C. Performance in presence of noise

The existing approaches that use the evaporation, mQSOE-sub and mQSOE-mult, have demonstrated that they achieve a very good performance when the fitness function is subjected to noise [1]. The goal of these experiments is to demonstrate that mQSODE achieves similar results in presence of noise. In order to aggregate noise to the fitness reads, we modified GDBG such as the fitness function incorporates a noise factor $\gamma$ in the following way:

$$\text{fitness}(\vec{p}) = \text{GDBG,fitness}(\vec{p}) + (2 \times \theta - 1) \times \gamma$$  \hspace{1cm} (13)

where $\theta$ generates a uniform random number between $[0..1]$. In these experiments we assume $\gamma = 4.5$. This value represents the 10% of the fitness value.

Figure 8 shows the average of relative best solutions found by the algorithms along the simulation. We can observe how the mQSO algorithm is not able to reach the convergence because the noise is misinterpreted as environment changes and produces a continuous resetting of the particles memory. This problem was exposed in [1] where they proposed a solution using a threshold value to filter the noise. Figure 8 shows how mQSODE solves the problem of slow adaptation after a change and moreover, achieves a similar performance than the mQSOE-mult and mQSOE-sub algorithms.

mQSODE presented a higher performance than the other (see Figure 11). Moreover, mQSODE achieves a clear higher performance than the others in the average worst and lower standard deviation. We consider it a big success to achieve a low standard deviation because it highlights a robust behavior in the optimisation process. Our proposal presents slightly lower results than mQSOE-sub and mQSOE-mult regarding the average best. Nevertheless, this difference is not statistically significant.

V. Conclusions

It has been demonstrated that the evaporation is an interesting mechanism for adapting PSO extensions in noisy and dynamic environments. Moreover, We consider the evaporation an important mechanism for adapting the standard PSO and its extensions in order to deal with dynamic environments, even in absence of noise. The two main motivations are: (1) The environment change detection by re-evaluating the best position found is not feasible in real environments (it is only possible in the existing benchmarks that assume that all the peaks change at the same time, i.e. the environment changes are easy to detect). Because of that, it is not possible to detect environment changes applying only a re-evaluation of the best position found. (2) The presence of noise is a challenge that environment change mechanisms must face, because it
is not easy to detect those changes when the fitness function is subjected to noise.

In this paper a dynamic evaporation mechanism (extending the mQSOE algorithm) for dynamic and noisy environments is proposed. Dynamic evaporation improves the performance of existing evaporation approaches. Moreover, our proposal achieved a lower standard deviation, i.e. mQSODE presents more stable results.

ACKNOWLEDGMENTS

This work was funded by EVE (TIN2009-14702-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.

REFERENCES