

# Dynamic Multi-swarm Particle Swarm Optimization with Fractional Global Best Formation

Jenni Pulkkinen  
Tampere University of Technology  
Tampere, Finland  
[jenni.pulkkinen@tut.fi](mailto:jenni.pulkkinen@tut.fi)

Serkan Kiranyaz  
Tampere University of Technology  
Tampere, Finland  
[serkan@cs.tut.fi](mailto:serkan@cs.tut.fi)

Moncef Gabbouj  
Tampere University of Technology  
Tampere, Finland  
[moncef.gabbouj@tut.fi](mailto:moncef.gabbouj@tut.fi)

## Abstract

Particle swarm optimization (PSO) has been initially proposed as an optimization technique for static environments; however, many real problems are dynamic, meaning that the environment and the characteristics of the global optimum can change over time. Thanks to its stochastic and population based nature, PSO can avoid being trapped in local optima and find the global optimum. However, this is never guaranteed and as the complexity of the problem rises, it becomes more probable that the PSO algorithm gets trapped into a local optimum due to premature convergence. In dynamic environments the optimization task is even more difficult, since after an environment change the earlier global optimum might become just a local optimum, and if the swarm is converged to that optimum, it is likely that new real optimum will not be found. For the same reason, local optima cannot be just discarded, because they can be later transformed into global optima. In this paper, we propose novel techniques, which successfully address these problems and exhibit a significant performance over multi-modal and non-stationary environments. In order to address the premature convergence problem and improve the rate of PSO's convergence to global optimum, Fractional Global Best Formation (FGBF) technique is developed. FGBF basically collects all the best dimensional components and fractionally creates an artificial Global Best particle (*aGB*) that has the potential to be a better "guide" than the PSO's native *gbest* particle. In this way the potential diversity that is present among the dimensions of swarm particles can be efficiently used within the *aGB* particle. To establish follow-up of (current) local optima, we then introduce a novel multi-swarm algorithm, which enables each swarm to converge to a different optimum and use FGBF technique distinctively. We investigated the proposed techniques over the Moving Peaks Benchmark (MPB), which is a publicly available test bench for testing optimization algorithms in a multi-modal dynamic environment. An extensive set of experiments show that FGBF technique with multi-swarms exhibits an impressive speed gain and tracks the global maximum peak with the minimum error so far achieved with respect to the other competitive PSO-based methods.

*Index Terms*—Particle Swarm Optimization, Fractional Global Best Formation

## 1 Introduction

Many real-world problems are dynamic and thus require systematic re-optimizations due to system and/or environmental changes. Even though it is possible to handle such dynamic problems as a series of individual processes via restarting the optimization algorithm after each change, this may lead to a significant loss of useful information, especially when the change is not too drastic. Since most of such problems have multi-modal nature, which further complicates the dynamic optimization problems, the need for powerful and efficient optimization techniques is imminent. In the last decade the efforts have been focused on evolutionary algorithms (EAs) [3] such as Genetic Algorithms (GA) [12], Genetic Programming (GP) [14], Evolution

Strategies (ES), [4] and Evolutionary Programming (EP) [11]. The common point of all EAs is that they have population based nature and they can avoid being trapped in a local optimum. Thus they can find the optimum solutions; however, this is never guaranteed.

Conceptually speaking, Particle Swarm Optimization (PSO) [13], which has obvious ties with the EA family, lies somewhere in between GA and EP. PSO is originated from the computer simulation of individuals (particles or living organisms) in a bird flock or fish school [22], which basically show a natural behavior when they search for some target (e.g. food). Their goal is, therefore, to converge to the global optimum of a possibly nonlinear function or system. Similarly, in a PSO process, a swarm of particles (or agents), each of which represents a po-

tential solution to an optimization problem, navigate through the search space. The particles are initially distributed randomly over the search space with a random velocity and the goal is to converge to the global optimum of a function or a system. Each particle keeps track of its position in the search space and its best solution so far achieved. This is the personal best value (the so-called *pbest* in [13]) and the PSO process also keeps track of the global best solution so far achieved by the swarm by remembering the index of the best particle (the so called *gbest* in [13]). During their journey with discrete time iterations, the velocity of each agent in the next iteration is affected by the best position of the swarm (the best position of the particle *gbest* as the social component), the best personal position of the particle (*pbest* as the cognitive component), and its current velocity (the memory term). Both social and cognitive components contribute randomly to the velocity of the agent in the next iteration.

Similar to the aforementioned EAs, PSO might exhibit some major problems and severe drawbacks such as parameter dependency [17] and loss of diversity [20]. Particularly the latter phenomenon increases the probability of being trapped in local optima and it is the main source of premature convergence problem especially when the search space is in high dimensions and the problem to be optimized is multi-modal [20]. Since PSO was proposed for static problems in general, effects of such drawbacks eventually become more severe for dynamic environments. Various modifications and PSO variants have been proposed in order to address these problems such as [1], [8], [15], [17] and [20]. Such methods usually try to improve the diversity among the particles and the search mechanism either by changing the update equations towards a more diversified versions or adding more randomization to the system (to particle velocities, positions, etc.). However, their performance improvement might be quite limited even in static environments and most of them use additional parameters and/or thresholds to accomplish this whilst making the PSO variant even more parameter dependent. Therefore, they do not set a reliable solution for dynamic environments, which usually have multi-modal nature and high dimensionality.

There are some efforts for simulating dynamic environments in a standard and configurable way. Some early works like [2] and [10] use experimental setup introduced by Angeline in [2]. In this setup the minimum of the three-dimensional parabolic function  $f(x, y, z) = x^2 + y^2 + z^2$  is moved along a linear or circular trajectory or randomly. However, this setup enables testing only in an uni-modal envi-

ronment. Branke in [7] has provided a publicly available Moving Peaks Benchmark (MPB) to enable dynamic optimization algorithms to be tested in a standard way in a multi-modal environment. MPB allows creation of different dynamic fitness functions consisting of a number of peaks with varying location, height and width. The primary measure for performance evaluation is offline error, which is the average difference between the optimum and the best evaluation since the last environment change. Obviously, this value is always a positive number and it is zero only for perfect tracking. Several PSO methods are developed and tested using MPB such as [5], [6], [16], and [18]. Particularly Blackwell and Branke in [5] proposed a successful multi-swarm approach. The idea behind this is that different swarms can converge to different peaks and track them when the environment changes. The swarms interact only by mutual repulsion that keeps two swarms from converging to the same peak.

In this paper, we shall first introduce a novel algorithm that significantly improves the global convergence performance of PSO by forming an artificial Global Best particle (*aGB*) fractionally. This algorithm, the so-called Fractional GB Formation (FGBF), collects the best dimensional components from each swarm particle and fractionally creates the *aGB* particle, which will replace *gbest* as guide for the swarm, if it turns out to be better than the swarm's native *gbest*. We then propose a novel multi-swarm algorithm, which combines multi-swarms with the FGBF technique so that each swarm can apply FGBF distinctively. Via applying the proposed techniques on MPB we shall show that they can find and track the global peak well even in high dimensions and usually in earlier stages. Furthermore, no additional parameter is needed to perform the proposed techniques.

The rest of the paper is organized as follows. Section 2 surveys related work on PSO and MPB. The proposed techniques, multi-swarms and FGBF and their applications over the MPB are presented in detail in Section 3. Section 4 provides the experiments conducted and discusses the results. Finally, Section 5 concludes the paper.

## 2 Related work

### 2.1 The basic PSO algorithm

In the basic PSO method, (*bPSO*), a swarm of particles flies through an  $N$ -dimensional search space where each particle represents a potential solution to the optimization problem. Each particle  $a$  in the swarm,  $\xi = \{x_1, \dots, x_a, \dots, x_S\}$ , is represented

by the following characteristics:

$x_{a,j}(t)$ :  $j^{\text{th}}$  dimensional component of the position of particle  $a$ , at time  $t$

$v_{a,j}(t)$ :  $j^{\text{th}}$  dimensional component of the velocity of particle  $a$ , at time  $t$

$y_{a,j}(t)$ :  $j^{\text{th}}$  dimensional component of the personal best (*pbest*) position of particle  $a$ , at time  $t$

$\hat{y}_j(t)$ :  $j^{\text{th}}$  dimensional component of the global best position of the swarm, at time  $t$

Let  $f$  denote the fitness function to be optimized. Without loss of generality assume that the objective is to find the maximum of  $f$  in an  $N$ -dimensional space. Then the personal best of particle  $a$  can be updated at iteration  $t$  as,

$$y_{a,j}(t) = \begin{cases} y_{a,j}(t-1) & \text{if } f(x_a(t)) < f(y_a(t-1)) \\ x_{a,j}(t) & \text{else} \end{cases} \quad j=1,2,\dots,N \quad (1)$$

Then at each iteration in a PSO process, positional updates are performed for each dimensional component,  $j \in \{1, N\}$  and for each particle,  $a \in \{1, S\}$ , as follows:

$$\begin{aligned} v_{a,j}(t+1) &= w(t)v_{a,j}(t) + c_1 r_{1,j}(t)(y_{a,j}(t) - x_{a,j}(t)) + c_2 r_{2,j}(t) \\ x_{a,j}(t+1) &= x_{a,j}(t) + v_{a,j}(t+1) \end{aligned} \quad (2)$$

where  $w$  is the inertia weight, [21] and  $c_1, c_2$  are the acceleration constants which are usually set to 1.49 or 2.  $r_{1,j} \sim U(0,1)$  and  $r_{2,j} \sim U(0,1)$  are random variables with uniform distribution. Recall from the

earlier discussion that the first term in the summation is the *memory* term, which represents the role of previous velocity over the current velocity, the second term is the *cognitive* component, which represents the particle's own experience and the third term is the *social* component through which the particle is "guided" by the *gbest* particle towards the GB solution so far obtained. Accordingly the general pseudo-code of the bPSO can be given as in Table 1.

Although the use of inertia weight,  $w$ , was later added by Shi and Eberhart [21], into the velocity update equation, it is widely accepted as the basic form of PSO algorithm. A larger value of  $w$  favors exploration while a small inertia weight favors exploitation. As originally introduced,  $w$  is often linearly decreased from a high value (e.g. 0.9) to a low value (e.g. 0.4) during iterations of a PSO run. Depending on the problem to be optimized, PSO iterations can be repeated until a specified number of iterations, say *IterNo*, is exceeded, velocity updates become zero, or the desired fitness score is achieved (i.e.  $f > \mathcal{E}_C$ ). Velocity clamping to the user-defined maximum velocity range  $V_{\max}$  (and  $-V_{\max}$  for the minimum) is one of the earliest attempts to avoid premature convergence [9].

Table 1: Pseudo-code of *bPSO* algorithm

<b>bPSO</b> ( termination criteria: { <i>IterNo</i> , $\mathcal{E}_C, \dots$ }, $V_{\max}$ )	
1.	For $\forall a \in \{1, S\}$ do:
1.1.	Randomize $x_a(1), v_a(1)$
1.2.	Let $y_a(0) = x_a(1)$
1.3.	Let $\hat{y}(0) = x_a(1)$
2.	End For.
3.	For $\forall t \in \{1, \textit{IterNo}\}$ do:
3.1.	For $\forall a \in \{1, S\}$ do:
3.1.1.	Compute $y_a(t)$ using (1)
3.1.2.	If ( $f(y_a(t)) > \max_{1 \leq i < a} (f(\hat{y}(t-1)), f(y_i(t)))$ ) then $gbest = a$ and $\hat{y}(t) = y_a(t)$
3.2.	End For.
3.3.	If any <i>termination criterion</i> is met, then <b>Return</b> .
3.4.	For $\forall a \in \{1, S\}$ do:
3.4.1.	For $\forall j \in \{1, N\}$ do:
3.4.1.1.	Compute $v_{a,j}(t+1)$ using (2)
3.4.1.2.	If ( $ v_{a,j}(t+1)  > V_{\max}$ ) then clamp it to $ v_{a,j}(t+1)  = V_{\max}$
3.4.1.3.	Compute $x_{a,j}(t+1)$ using (2)
3.4.2.	End For.
3.5.	End For.
4.	End For.

## 2.2 Moving Peaks Benchmark

Conceptually speaking, MPB developed by Branke in [7], is a simulation of a configurable dynamic environment changing over time. The environment consists of a certain number peaks with varying location, height and width. The dimensionality of the fitness function is fixed in advance and thus is an input parameter of the benchmark. Type and number of peaks along with their initial heights and widths, environment dimension and size, change severity, level of change randomness and change frequency can be defined. To facilitate standard comparative evaluations among different algorithms, three standard settings of such MPB parameters, so called “*Scenarios*”, have been defined. *Scenario 2* is the most widely used. Where the scenario allows a range of values, the following are commonly used: number of peaks = 10, change severity  $vlength = 1.0$ , correlation  $lambda = 0.0$  and peak change frequency = 5000. In *Scenario 2* no basis landscape is used and peak type is a simple cone. Due to the page limit more formal description and further details can be obtained from [7].

## 2.3 Multi-swarm PSO

The main problem of using the basic PSO algorithm in a dynamic environment is that eventually the swarm will converge to a single peak – whether global or local. When another peak becomes the global maximum as a result of an environmental change, it is likely that the particles keep circulating close to the peak to which the swarm has converged and thus they cannot find the new global maximum. Blackwell and Branke have addressed this problem in [5] and [6] by introducing multi-swarms. Multi-swarms are actually separate PSO processes. Each particle is now a member of one of the swarms only and it is unaware of other swarms. The main idea is that each swarm can converge to a separate peak. Swarms interact only by mutual repulsion that keeps them from converging to the same peak. For a single swarm it is essential to maintain enough diversity so that the swarm can track small location changes of the peak to which it is converging. For this purpose Blackwell and Branke introduced charged and quantum swarms, which are analogues to an atom having a nucleus and charged particles randomly orbiting it. The particles in the nucleus take care of the fine tuning of the result while the charged particles are responsible of detecting the position changes. However, it is clear that, instead of charged or quantum swarms, any method can be used to ensure sufficient diversity among particles of a single swarm so that the peak can be tracked despite of small location

changes. As one might expect, the best results are achieved when the number of swarms is set equal to the number of peaks.

The repulsion between swarms is realized by simply re-initializing worse of two swarms if they move within a certain range from each other. Using physical repulsion could lead to equilibrium, where swarm repulsion prevents both swarms from getting close to a peak. A proper limit closer to which the swarms are not allowed to move,  $r_{rep}$  is attained by using the average radius of the peak basin,  $r_{bas}$ . If  $p$  peaks are evenly distributed in  $X^N$ ,  $r_{rep} = r_{bas} = X / p^{1/N}$ .

## 3 The Proposed Techniques for Dynamic Environments

### 3.1 FGBF Technique

Fractional Global Best Formation (FGBF) is designed to avoid the premature convergence by providing a significant diversity obtained from a proper fusion of the swarm’s best components (the individual dimension(s) of the current position of each particle in the swarm). At each iteration in a PSO process, an artificial GB particle (*aGB*) is (fractionally) formed by selecting best particle (dimensional) components from the entire swarm. Therefore, especially during the initial steps, the FGBF can be and, most of the time, is a better alternative than the native *gbest* particle since it has the advantage of assessing each dimension of every particle in the swarm individually, and forming the *aGB* particle fractionally by using the best components among them. This process naturally uses the available diversity among individual dimensional components and thus it can prevent swarm from being trapped in local optima due to its ongoing and ever-varying particle creations. At each iteration FGBF is performed after the assignment of the swarm’s *gbest* particle (i.e. performed between steps 3.2 and 3.3 in the pseudo-code of *bPSO*) and, if *aGB* turns out to be better than *gbest*, the personal best location of the *gbest* particle is replaced by the location of the *aGB* particle and, since  $\hat{y}(t) = y_{gbest}(t)$ , the artificially created particle is thus used to guide the swarm through the *social* component in (2). In other words, the swarm will be guided only by the best (winner) between native *gbest* and the *aGB* particle at any time. In the next iteration, a new *aGB* particle is created and it will again compete against the personal best of *gbest* (which can be also a former *aGB* now).

Suppose that for a swarm  $\xi$ , FGBF is per-

formed in a PSO process in a dimension  $N$ . Recall from the earlier discussion that in a particular iteration,  $t$ , each PSO particle,  $a$ , has the following components: position  $(x_{a,j}(t))$ , velocity  $(v_{a,j}(t))$  and the personal best position  $(y_{a,j}(t))$ ,  $j \in \{1, N\}$ . As the  $aGB$  particle is fractionally (re-) created from the dimensions of some swarm particles at each iteration, it does not need the velocity term and, therefore, it does not have to remember its personal best location.

Let  $f(a, j)$  be the dimensional fitness score of the  $j^{\text{th}}$  component of the position of particle  $a$  and  $f(gbest, j)$  be the dimensional fitness score of the  $j^{\text{th}}$  component of the personal best position of the  $gbest$  particle. Suppose that all dimensional fitness scores  $(f(a, j), \forall a \in \{1, S\} \text{ and } f(gbest, j))$  can be computed in step 3.1 and FGBF can then be plugged in between steps 3.2 and 3.3 of  $bPSO$ 's pseudo-code. Accordingly, the pseudo-code for FGBF can be expressed as given in Table 2.

Step 2 along with the computation of  $f(a, j)$  depends entirely on the optimization problem. It keeps track of partial fitness contributions from each individual dimension from each particle's position (the potential solution). Take for instance the function minimization problem as illustrated in Figure 1 where 2D space is used for illustration purposes. In the figure, three particles in a swarm are ranked as the 1st (or the  $gbest$ ), the 3rd and the 8th with respect to their proximity to the target position (or the global solution) of some function. Although  $gbest$  particle (i.e. 1st rank particle) is the closest in the overall sense, the particles ranked 3rd and 8th provide the best x and y dimensions (closest to the target's respective dimensions) in the entire swarm and hence the  $aGB$  particle via FGBF yields a better (closer) particle than the swarm's native  $gbest$ .

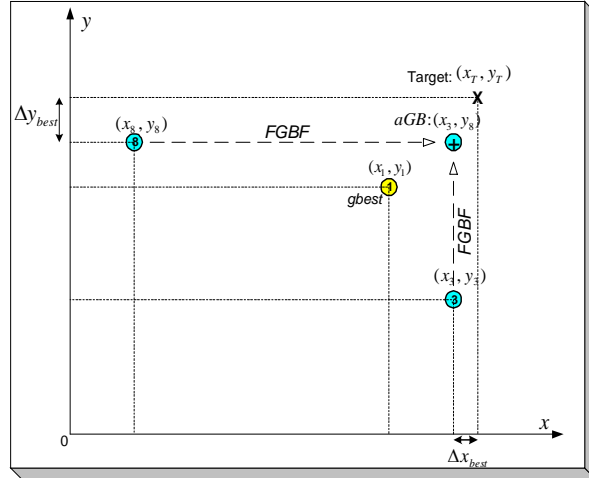


Figure 1: A sample FGBF operation in 2D space.

### 3.2 FGBF Application for MPB

The previous section introduced the principles of FGBF within a  $bPSO$  process in a static environment. However, in dynamic environments this approach eventually leads the swarm to converge to a single peak (whether global or local) and therefore, it may lose its ability to track other peaks. As any of the peaks can become the optimum peak as a result of environmental changes, it is likely to lead to a suboptimal convergence. This is the basic reason of utilizing the multi-swarms with the FGBF operation within each of them. The mutual repulsion between swarms is implemented as described in Section 2.3. For computing the distance between two swarms we use distance of the global best locations of the swarms. Instead of charged or quantum swarms, FGBF is the entire mechanism to provide enough diversity and thus to enable peak tracking if peaks' location are slightly changed. We also re-initialize the particle velocities after each environment change to further contribute to the diversity.

Table 2: Pseudo-code of FGBF

<b>FGBF in <math>bPSO</math> (<math>\xi, f(a, j)</math>)</b>	
1.	Let $a[j] = \arg \max_{a \in \xi, j \in \{1, N\}} (f(a, j))$ be the index of particle yielding the maximum $f(a, j)$ for the $j^{\text{th}}$ dimensional component.
2.	$x_{aGB,j}(t) = x_{a[j],j}(t)$ for $\forall j \in \{1, N\}$
3.	If $f(gbest, j) > f(a[j], j)$ then $x_{aGB,j}(t) = y_{gbest,j}(t)$
4.	If $(f(x_{aGB}(t)) > f(y_{gbest}(t)))$ then $y_{gbest}(t) = x_{aGB}(t)$ and $\hat{y}(t) = x_{aGB}(t)$
5.	<b>Return.</b>

Each particle  $a$  in a swarm  $\xi$ , represents a potential solution and therefore, the  $j^{\text{th}}$  component of an  $N$ -dimensional point  $(x_j, j \in \{1, N\})$  is stored in its positional component,  $x_{a,j}(t)$  at time  $t$ . The aim of the PSO process is to search for the center point of the global maximum peak. Recall that in *Scenario 2* of MPB the peaks used are all in cone shape and finding the highest peak is, therefore, equivalent to minimizing the  $\|\bar{x} - \bar{c}_p(t)\|$  term, where  $\bar{x}$  is a position found by the algorithm,  $\bar{c}_p(t)$  is the center point of the highest cone and  $\|\cdot\|$  is the *Euclidean* distance between them. This yields  $f(a, j) = -(x_j - c_{pj})^2$ . Step 3.1 in *bPSO*'s pseudo-code computes the (dimensional) fitness scores ( $f(a, j), f(gbest, j)$ ) of the  $j^{\text{th}}$  components ( $x_{a,j}, y_{gbest,j}$ ) and in step 1 of the FGBF process, the dimensional component yielding maximum  $f(a, j)$  is then placed in *aGB*. In step 3 these dimensional components are replaced by dimensional components of the personal best position of the *gbest* particle, if they yield even higher dimensional fitness scores. We do not expect that dimensional fitness scores can be evaluated with respect to the optimum peak since this requires the a priori knowledge of the global optimum, instead we use either the current peak where the particle resides on or the peak to which the swarm is converging (swarm peak). We shall thus consider and evaluate both modes separately.

## 4 Experimental results

We conducted an exhaustive set of experiments over the MPB *Scenario 2* using the settings given in Section 2.2. In order to investigate the effect of multi-swarm settings, we used different numbers of swarms and numbers of particles in a swarm. We applied both FGBF modes using the current and swarm peaks and to investigate how FGBF and multi-swarms individually contribute to the results, we also made experiments without using one of them.

Figure 2 presents the current error plot, which shows the difference between the global maximum and the current best result during the first 80000 function evaluation, when 10 swarms each with 4 particles are used and the swarm peak mode is applied for the FGBF operation. It can be seen from the figure that as the environment changes after every 5000 evaluation, it causes results to temporar-

ily deteriorate. However, it is clear that after environment changes the results are better than the very beginning, which shows the benefit of tracking the peaks instead of randomizing the swarm when a change occurs. The figure also reveals other typical features of algorithm behavior. First of all, after the first few environmental changes the algorithm is not yet behaving as well as later. This is because not the swarms have yet converged to a peak. Generally, it is more difficult to initially converge to a narrow or low peak than to keep tracking a peak that becomes narrow and/or low. It can also be seen that typically the algorithm gets close to the optimal solution before the environment is changed again. In few cases, where the optimal solution is not found, the algorithm has for some reason been unable to keep a swarm tracking that peak, which is too narrow.

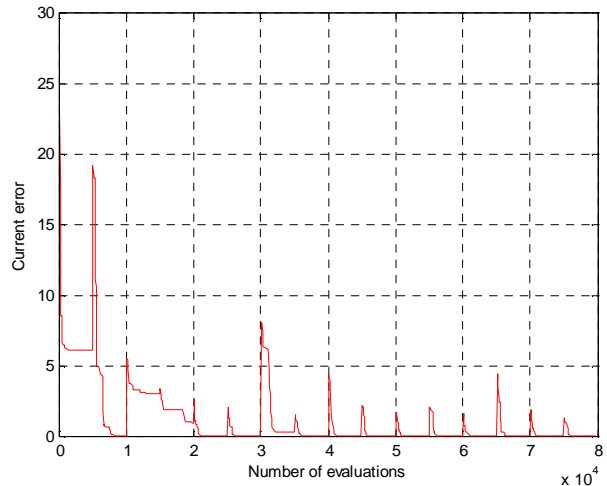


Figure 2: Current error at the beginning of a run

In Figure 3 and Figure 4 the contributions of multi-swarms with FGBF are demonstrated. The algorithm is run on MPB using same random number seed (same environment changes) first with both multi-swarms and FGBF, then without multi-swarms and finally without FGBF. Same settings are used as before. Without multi-swarms the number of particles is set to 40 to keep the total number of particles unchanged.

As expected, the results without multi-swarms are significantly deteriorated due to the aforementioned reasoning. When the environment is changed, the highest point of the peak to which the swarm is converging can be found quickly, but that can provide good results only when that peak happens to be the global optimum. When multi-swarms are used, but without using the FGBF, it is clear that the algorithm can still establish some kind of follow-up of peaks as the results immediately after environment

changes are only slightly worse than with FGBF. However, if FGBF is not used, the algorithm can seldom find the global optimum. Either there is no swarm converging to the highest peak or the peak center just cannot be found fast enough.

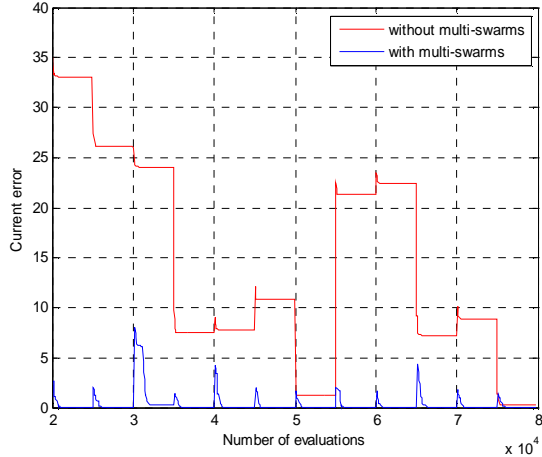


Figure 3: Effect of multi-swarms on results

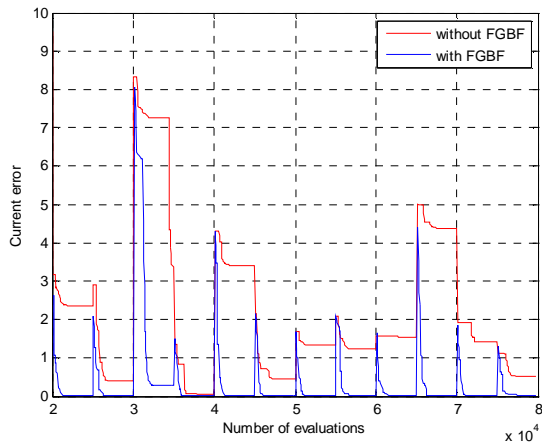


Figure 4: Effect of FGBF on results

For comparative evaluations, we selected 5 of the state-of-the-art methods, which use the same benchmark system, the MPB. The best MPB results published so far by these competing methods are listed in Table 3.

Table 3: Best results on MPB up to date

Source	Algorithm	Offline error
Blackwell and Branke [5]	PSO	2.16±0.06
Li et. al[16]	PSO	1.93±0.06

Mendes and Mohais [18]	Differential Evolution	1.75±0.03
Blackwell and Branke [6]	PSO	1.75±0.06
Moser and Hendtlass [19]	Extremal Optimization	0.66±0.02

The overall best results have been achieved by the Extremal Optimization algorithm [19]; however, this algorithm is specially designed for MPB and its applicability for other practical dynamic problems is not clear. The best results by a PSO-based algorithm have been achieved by Blackwell and Branke’s multi-swarm algorithm described in Section 2.3. The numerical results of the proposed methods in terms of the offline error are listed in Table 4. Each result given is the average of 50 runs, where each run consists of 500000 function evaluations.

Table 4: Offline error using *Scenario 2*

No. of swarms	No. of particles	Swarm peak	Current peak
10	2	1.81±0.50	2.58±0.55
10	3	1.22±0.43	1.64±0.53
10	4	<b>1.03±0.35</b>	<b>1.37±0.50</b>
10	5	1.19±0.32	1.52±0.44
10	6	1.27±0.41	1.59±0.57
10	8	1.31±0.43	1.61±0.45
10	10	1.40±0.39	1.70±0.55
8	4	1.50±0.41	1.78±0.57
9	4	1.31±0.54	1.66±0.54
11	4	1.09±0.35	1.41±0.42
12	4	1.11±0.30	1.46±0.43

As expected the best results are achieved when 10 swarms are used. 4 particles in a swarm turned out to be the best setting. Between the two FGBF modes, better results are obtained when the swarm peak mode is used.

## 4 Conclusion

In this paper, we proposed a novel PSO technique, namely, FGBF with the multi-swarms for an efficient and robust optimization over the dynamic systems. The technique can also be used over the static optimization problems particularly as a cure to common drawback of the family of PSO methods, pre-mature convergence to local optima. Realizing that the main problem lies in fact at the inability of using the available diversity among the dimensional components of swarm particles, the FGBF technique proposed in this paper collects the best components

and fractionally creates an *aGB* particle that has the potential to be a better “guide” than the swarm’s native *gbest* particle. On MPB we do not except to receive fractional scores with respect to the global highest peak, but instead we use either the peak, on which the particle is currently located (current peak) or the peak to which the swarm is converging (swarm peak). Especially swarm peak mode makes it possible to find and track the global highest peak successfully in a dynamic environment.

In order to make comparative evaluations with the current state-of-the-art, FGBF with multi-swarms is then applied over a benchmark system, the MPB. The results over the MPB with common settings used (i.e. *Scenario 2*) clearly indicate the superiority of the proposed technique over other PSO-based methods.

Overall, the proposed technique fundamentally upgrades the swarm guidance, which accomplishes substantial improvements in terms of speed and accuracy. The FGBF technique is modular and independent, i.e. it can be conveniently performed also with other PSO methods/variants.

## References

- [1] A. Abraham, S. Das and S. Roy, “Swarm Intelligence Algorithms for Data Clustering”, in *Soft Computing for Knowledge Discovery and Data Mining* book, Part IV, pp. 279-313, October 25, 2007.
- [2] P.J. Angeline, “Tracking extrema in dynamic environments”, In *Proc. Of the 6<sup>th</sup> Conference on Evolutionary Programming*, pp.335-345, Springer Verlag, 1997
- [3] T. Bäck and H.P. Schwefel, “An overview of evolutionary algorithms for parameter optimization”, *Evolution. Comput.* 1, pp. 1–23, 1993.
- [4] T. Bäck and F. Kursawe, “Evolutionary algorithms for fuzzy logic: a brief overview”, In *Fuzzy Logic and Soft Computing*, World Scientific, pp. 3–10, Singapore, 1995.
- [5] T.M. Blackwell and J. Branke, “Multi-Swarm Optimization in Dynamic Environments”, *Applications of Evolutionary Computation*, vol. 3005, pp. 489-500, Springer, 2004.
- [6] T.M. Blackwell and J. Branke, “Multiswarms, Exclusion, and Anti-Convergence in Dynamic Environments”, *IEEE Transactions on Evolutionary Computation*, vol. 10/4, pp. 51-58, 2004.
- [7] J. Branke, “Moving Peaks Benchmark”, <http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/>, viewed 26/06/08
- [8] Y.-P. Chen, W.-C. Peng; M.-C. Jian, “Particle Swarm Optimization With Recombination and Dynamic Linkage Discovery”, in *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, Vol. 37, Issue 6, pp. 1460 – 1470, Dec. 2007.
- [9] R. Eberhart, P. Simpson, and R. Dobbins, *Computational Intelligence. PC Tools*, Academic Press, Inc., Boston, MA, USA, 1996.
- [10] R. Eberhart and Y. Shi, “Tracking and Optimizing Dynamic Systems with Particle Swarms”, in *Proc. of Computational Evolution Conference (CEC 2001)*, NJ, US, pp. 94-100, 2001.
- [11] U.M. Fayyad, G.P. Shapire, P. Smyth and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*, MIT Press, Cambridge, MA, 1996.
- [12] D. Goldberg, “Genetic Algorithms in Search, Optimization and Machine Learning”, Addison-Wesley, Reading, pp. 1-25. MA, 1989.
- [13] J. Kennedy, R Eberhart., “Particle swarm optimization”, in *Proc. of IEEE Int. Conf. On Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, 1995.
- [14] J. Koza, *Genetic Programming: On the Programming of Computers by means of Natural Selection*, MIT Press, Cambridge, Massachusetts, 1992.
- [15] R. A. Krohling, L S. Coelho, “Coevolutionary Particle Swarm Optimization Using Gaussian Distribution for Solving Constrained Optimization Problems”, *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, Vol. 36, Issue 6, pp. 1407 – 1416, Dec. 2006.
- [16] X. Li, J. Branke and T. Blackwell, “Particle Swarm with Speciation and Adaptation in a Dynamic Environment”, *Proc. of Genetic and Evolutionary Computation Conference*, pp. 51-58, Seattle Washington, 2006.
- [17] M. Lovberg and T. Krink, “Extending Particle Swarm Optimisers with Self-Organized Criticality”, In *Proc. of the IEEE Congress on Evolutionary Computation*, vol. 2, pp.1588-1593, 2002.
- [18] R. Mendes and A. Mohais, “DynDE: a Differential Evolution for Dynamic Optimization Problems”, *IEEE Congress on Evolutionary Computation*, pp. 2808-2815, 2005.
- [19] I. Moser and T. Hendtlass, “A Simple and Efficient Multi-Component Algorithm for Solving Dynamic Function Optimisation Problems”, *IEEE Congress on Evolutionary Computation*, pp. 252-259, 2007.
- [20] J. Riget and J. S. Vesterstrom, “A Diversity-Guided Particle Swarm Optimizer - The ARPSO”, Technical report, Department of Computer Science, University of Aarhus, 2002.
- [21] Y. Shi and R.C. Eberhart, “A Modified Particle Swarm Optimizer”, In *Proc. of the IEEE Congress on Evolutionary Computation*, pp. 69-73, 1998.
- [22] E.O. Wilson, *Sociobiology: The new synthesis*, Cambridge, MA: Belknap Press, 1975.