

Multi-dimensional particle swarm optimization in dynamic environments

Serkan Kiranyaz*, Jenni Pulkkinen, Moncef Gabbouj¹

Department of Signal Processing, Tampere University of Technology, P.O. Box 553 FIN-33101, Tampere, Finland

ARTICLE INFO

Keywords:

Particle swarm optimization
Multi-dimensional search
Fractional Global Best Formation

ABSTRACT

Particle swarm optimization (PSO) was 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 in time. In this paper, we adapt recent techniques, which successfully address several major problems of PSO and exhibit a significant performance over multi-modal and non-stationary environments. In order to address the pre-mature convergence problem and improve the rate of PSO's convergence to the global optimum, Fractional Global Best Formation (FGBF) technique is used. FGBF basically collects all the best dimensional components and fractionally creates an artificial Global Best particle (α GB) that has the potential to be a better "guide" than the PSO's native gbest particle. To establish follow-up of 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. Finally for the multi-dimensional dynamic environments where the optimum dimension also changes in time, we utilize a recent PSO technique, the multi-dimensional (MD) PSO, which re-forms the native structure of the swarm particles in such a way that they can make inter-dimensional passes with a dedicated dimensional PSO process. Therefore, in a multi-dimensional search space where the optimum dimension is unknown, swarm particles can seek for both positional and dimensional optima. This eventually pushes the frontier of the optimization problems in dynamic environments towards a global search in a multi-dimensional space, where there exists a multi-modal problem possibly in each dimension. We investigated both standalone and mutual applications of the proposed methods over the moving peaks benchmark (MPB), which originally simulates a dynamic environment in a unique (fixed) dimension. MPB is appropriately extended to accomplish the simulation of a multi-dimensional dynamic system, which contains dynamic environments active in several dimensions. An extensive set of experiments show that in traditional MPB application domain, FGBF technique applied with multi-swarms exhibits an impressive speed gain and tracks the global peak with the minimum error so far achieved with respect to the other competitive PSO-based methods. When applied over the extended MPB, MD PSO with FGBF can find optimum dimension and provide the (near-) optimal solution in this dimension.

© 2010 Elsevier Ltd. All rights reserved.

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 a multi-modal nature, which further complicates the dynamic optimization problem, the need for powerful and efficient optimization techniques is imminent.

In the last decade the efforts have been focused on evolutionary algorithms (EAs) (Bäck & Schwefel, 1993) such as Genetic Algorithms (GA) (Goldberg, 1989), Genetic Programming (GP) (Koza, 1992), Evolution Strategies (ES), (Bäck & Kursawe, 1995) and Evolutionary Programming (EP) (Fayyad, Shapire, Smyth, & Uthurusamy, 1996). The common point of all EAs, which have population based nature, is that they may also avoid being trapped in local optima. Thus they can find the optimum solutions; however, this is never guaranteed.

Conceptually speaking, particle swarm optimization (PSO) (Engelbrecht, 2005; Kennedy & Eberhart, 1995; Omran, Salman, & Engelbrecht, 2006), which has obvious ties with the EA family, lies somewhere in between GA and EP. Yet unlike GA, PSO has no complicated evolutionary operators such as *crossover*, *selection* and *mutation* and it is highly dependent on stochastic processes. PSO is originated from the computer simulation of individuals (particles or living organisms) in a bird flock or fish school (Wilson,

* Corresponding author. Tel.: +358 50 4324123; fax: +358 (0)3 3115 4989.

E-mail addresses: serkan.kiranyaz@tut.fi (S. Kiranyaz), jenni.pulkkinen@tut.fi (J. Pulkkinen), moncef.gabbouj@tut.fi (M. Gabbouj).

¹ This work was supported by the Academy of Finland, project No. 213462 (Finnish Centre of Excellence Program (2006–2011).

1975), 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 non-linear function or system. Similarly, in a PSO process, a swarm of particles (or agents), each of which represents a potential 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 Kennedy & Eberhart (1995)) 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 Kennedy & Eberhart (1995)). 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.

There are some efforts for simulating dynamic environments in a standard and configurable way. Some early works such as (Angeline, 1997, 1998; Eberhart & Shi, 2001) use experimental setup introduced by Angeline (1997). 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. Three different update frequencies (200, 1000 and 2000 evaluations) and change severities (0.01, 0.1, 0.5) are used. However, this setup enables testing only in a uni-modal environment. Branke (2008) has provided a publicly available Moving Peaks Benchmark (MPB) to enable different dynamic optimization algorithms to be tested in a standard way in a multi-modal environment. MPB allows the 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 have been developed and tested using MPB such as (Blackwell & Branke, 2004a; Blackwell & Branke, 2004b; Li, Branke, & Blackwell, 2006; Mendes & Mohais, 2005). Particularly Blackwell & Branke (2004a) 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 any two swarms from converging to the same peak.

Similar to the aforementioned EAs, PSO might exhibit some major problems and severe drawbacks such as parameter dependency (Lovberg & Krink, 2002) and loss of diversity (Riget & Vesterstrom, 2002). Particularly the latter phenomenon increases the probability of being trapped in a local optimum and it is the main source of premature convergence especially when the dimensionality of the search space is large (Van den Bergh, 2002) and the problem to be optimized is multi-modal (Esquivel & Coello, 2003; Riget & Vesterstrom, 2002). Another reason for premature convergence is that particles are flown through a single point, which is (randomly) determined by *gbest* and *pbest* positions and this point is not even guaranteed to be a local optimum (Van den Bergh & Engelbrecht, 2002). Since PSO was proposed for static problems in general, effects of such drawbacks eventually become much more severe for dynamic environments. Various modifications and PSO variants have been proposed in order to address these problems such as (Abraham, Das, & Roy, 2007; Chen & Li, 2007; Chen, Peng, & Jian, 2007; Christopher & Seppi, 2004; Clerc, 1999; Eberhart, Simpson, & Dobbins, 1996; Higashi & Iba, 2003; Ince, Kiranyaz, & Gabbouj,

2009; Janson & Middendorf, 2005; Kaewkamnerdpong & Bentley, 2005; Krohling & Coelho, 2006; Liang & Qin, 2006; Li et al., 2006; Lovberg, 2002; Lovberg & Krink, 2002; Mendes, Kennedy, & Neves, 2004; Peng, Reynolds, & Brewster, 2003; Peram, Veeramachaneni, & Mohan, 2003; Ratnaweera, Halgamuge, & Watson, 2003; Ratnaweera, Halgamuge, & Watson, 2002; Riget & Vesterstrom, 2002; Richards & Ventura, 2003; Shi & Eberhart, 1998; Shi & Eberhart, 2001; Van den Bergh & Engelbrecht, 2002; Van den Bergh & Engelbrecht, 2004; Xie, Zhang, & Yang, 2002a; Xie, Zhang, & Yang, 2002b; Xie, Zhang, & Yang, 2002c; Yasuda, Ide, & Iwasaki, 2003; Zhang & Xie, 2003). Such methods usually try to improve the diversity among the particles and the search mechanism either by changing the update equations towards more diversified versions or adding more randomization to the system (to particle velocities, positions, etc.) or simply resetting some or all of them randomly when some conditions are met. However, their performance improvement might be quite limited even in static environments and most of them use *more* parameters and/or thresholds to accomplish this whilst making the PSO variant even more parameter dependent. Therefore, they do not yield set a reliable solution for dynamic environments, which usually have a multi-modal nature and high dimensionality.

Another major drawback of the basic PSO and the aforementioned variants is that they can only be applied to a search (solution) space with a fixed dimensionality. However, in many optimization problems, the optimum dimension is also unknown (e.g. data clustering, object extraction, optimization of the dynamic functions, etc.) and should thus be determined within the PSO process. Take for example the color-based *image* segmentation as a data clustering application, where the optimum dimension of the solution space corresponds to the true number of clusters (segments) in the data (color) space, which cannot be known beforehand. In such a case the PSO process should perform a multi-dimensional search in order to determine both the true (optimum) number of clusters and the optimum centroid location for each cluster. The problem becomes even more challenging when it is applied over a dynamic environment such as a *video* where both the number of clusters (segments) and their centroids (dominant colors) are changing over time. Yet since the change between consecutive frames is not drastic but rather minor, instead of performing a new clustering in the color domain via multi-dimensional search for each frame in the video, for a new (next) frame, the PSO process can establish a follow-up mechanism to track the optimum (number of) clusters (segments) from the previous frame. Therefore, using the past history about global and local optima becomes a crucial information to search for the current optima (both dimension and location).

In this paper, we shall first introduce a recent technique that significantly improves the global convergence performance of PSO by forming an artificial Global Best particle (aGB) fractionally. This algorithm, the so-called Fractional Global Best Formation (FGBF), collects the best dimensional components from each swarm particle and fractionally creates the aGB, which will replace *gbest* as a guide for the swarm, if it turns out to be better than the swarm's native *gbest* particle. 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 conventional MPB we shall show that they can find and track the global peak in an efficient way and usually in earlier stages. For the multi-dimensional dynamic environments where the optimum dimension also changes over time, we shall then introduce a multi-dimensional (MD) PSO technique, which re-forms the native structure of swarm particles in such a way that they can make inter-dimensional passes with a dedicated dimensional PSO process. Therefore, in a multi-dimensional search space where the optimum dimension is unknown, swarm particles can seek for both positional and dimensional

optima. This eventually pushes the frontier of optimization problems in dynamic environments towards a *global* search in a multi-dimensional space, where the problems in each dimension are possibly multi-modal and dependent on each other in a certain manner. Since the conventional MPB is created for a unique (fixed) dimensionality, we shall propose multi-dimensional extension of the benchmark in which there exists a unique optimum dimension where the global (highest) peak is located. Also the optimum dimension can change over time within a dimension range. In order to provide a certain degree of dependency among individual dimensions, peaks in different dimensions share the common coordinates of the peak locations. This is basically accomplished by subtracting a penalty term, whose magnitude depends on a dimensional error, from the landscape height in non-optimal dimensions. As a result, over the extended MPB, MD PSO can seek for both the optimum dimension and the global peak on it simultaneously. FGBF is an add-on to both multi-swarms and MD PSO which can enhance their performance. We shall show when the multi-dimensional search is needed, the best performance is achieved by their mutual operation. In recent works, both MD PSO and FGBF have been successfully applied over *static* problems such as general data clustering (Kiranyaz, Ince, Yildirim, & Gabbouj, 2010) and evolutionary artificial neural networks (Ince et al., 2009), respectively.

The rest of the paper is organized as follows. Section 2 surveys related work on PSO and MPB. The proposed techniques, MD PSO, multi-swarms with FGBF and their applications over the (extended) 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 with index a in the swarm, $\xi = \{x_1, \dots, x_a, \dots, x_S\}$, is represented by the following characteristics:

- $x_{aj}(t)$: j th dimensional component of the position of particle a , at time t .
- $v_{aj}(t)$: j th dimensional component of the velocity of particle a , at time t .
- $y_{aj}(t)$: j th dimensional component of the personal best (*pbest*) position of particle a , at time t .
- $\hat{y}_j(t)$: j 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,

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

Since *gbest* is the index of the GB particle, $\hat{y}(t) = y_{gbest}(t)$. 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:

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

where w is the inertia weight, (Shi & Eberhart, 1998) and c_1, c_2 are the acceleration constants which are usually set to 1.49 or 2. $r_{1j} \sim U(0, 1)$ and $r_{2j} \sim U(0, 1)$ 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 contribution of the 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. Although the use of inertia weight, w , was later added by Shi & Eberhart (1998), 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. $f > \varepsilon_C$). Accordingly the general pseudo-code of the *bPSO* can be given as follows:

bPSO (termination criteria: $\{IterNo, \varepsilon_C, \dots\}, V_{max}$)

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 $\forall t \in [1, IterNo]$ do:
 - 3.1 For $\forall a \in [1, S]$ do:
 - 3.1.1 Compute $y_a(t)$ using (1)
 - 3.1.2 If $(f(y_a(t)) > \max(f(\hat{y}(t-1)), f(y_i(t))))$ then $gbest = a$ and $\hat{y}(t) = y_a(t)$ $1 \leq i < a$
 - 3.2 End For.
 - 3.3 If any *termination criterion* is met, then **Return**.
 - 3.4 For $\forall a \in [1, S]$ do:
 - 3.4.1 For $\forall j \in [1, N]$ do:
 - 3.4.1.1 Compute $v_{aj}(t+1)$ using (2)
 - 3.4.1.2 If $(|v_{aj}(t+1)| > V_{max})$ then clamp it to $|v_{aj}(t+1)| = V_{max}$
 - 3.4.1.3 Compute $x_{aj}(t+1)$ using (2)
 - 3.4.2 End For.
 - 3.5 End For.
 4. End For.
 5. **Return**.

Velocity clamping also called "dampening" with the user-defined maximum range V_{max} (and $-V_{max}$ for the minimum) as in Step 3.4.1.2 is one of the earliest attempts to control or prevent oscillations. Such oscillations are indeed crucial since they broaden the search capability of the swarm; however, they have a potential drawback of oscillating continuously around the optimum point. Therefore, such oscillations should be dampened and the convergence is achieved with the proper use of the velocity clamping and the inertia weight. Furthermore, this is the *bPSO* algorithm where the particle *gbest* is determined within the entire swarm. Another major topological approach, the so-called *lbest*, also exists where the swarm is divided into overlapping neighborhoods of particles and instead of defining *gbest* and $\hat{y}(t) = y_{gbest}(t)$ over the entire swarm, for a particular neighborhood N_i , the (local) best particle is referred as *lbest* with the position $\hat{y}_i(t) = y_{lbest}(t)$. Neighbors can be defined with respect to particle indices (i.e. $i \in [j-l, \dots, j+l]$) or by using some other topological forms (Suganthan, 1999). It is

obvious that *gbest* is a special case of *lbest* scheme where the neighborhood is defined as the entire swarm. The *lbest* approach is one of the earlier attempts, which usually improves the diversity; however, it is slower than the *gbest* approach. PSO variants for dynamic environments will be reviewed in Section 2.3.

2.2. Moving Peaks Benchmark

Conceptually speaking, MPB developed by Branke (2008), is a simulation of a configurable dynamic environment changing over time. The environment consists of a certain number of peaks with varying locations, heights and widths. The dimensionality of the fitness function is fixed in advance and thus is an input parameter of the benchmark. An N -dimensional fitness function with m peaks is expressed as,

$$F(\vec{x}, t) = \max \left(B(\vec{x}), \max_{p=1 \dots m} P(\vec{x}, h_p(t), w_p(t), \vec{c}_p(t)) \right) \quad (3)$$

where $B(\vec{x})$ is a time variant basis landscape, whose utilization is optional, and P is the function defining the height of the p th peak at location \vec{x} , where each of the m peaks can have its own dynamic parameters such as height, $h_p(t)$, width, $w_p(t)$ and location vector of the peak center, $\vec{c}_p(t)$. Each peak parameter can be initialized randomly or set to a certain value and after a time period (number of evaluations), T_e , at time (evaluation) t , a change over a single peak, p , can be defined as follows:

$$\begin{aligned} h_p(t) &= h_p(t - T_e) + r\Delta h \\ w_p(t) &= w_p(t - T_e) + r\Delta w \\ \vec{c}_p(t) &= \vec{c}_p(t - T_e) + \vec{v}_p(t) \end{aligned} \quad (4)$$

where $r \sim U(0, 1)$, Δh and Δw are the heights and width change severities, respectively, and $\vec{v}_p(t)$ is a shift vector, which is a linear combination of a random vector and the previous shift vector, $\vec{v}_p(t - T_e)$. The shift vector $\vec{v}_p(t)$ is always normalized to length $vlength$, which is called change severity. Accordingly, the shift vector $\vec{v}_p(t)$ can be defined as

$$\vec{v}_p(t) = vlength \frac{(1 - \lambda)\vec{r}(t) + \lambda\vec{v}_p(t - T_e)}{\|(1 - \lambda)\vec{r}(t) + \lambda\vec{v}_p(t - T_e)\|}$$

where λ is the correlation factor, which defines the level of location change randomness. The types and number of peaks along with their initial heights and widths, environment (search space) 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, the so-called *Scenarios*, have been defined. *Scenario 2* is the most widely used and it allows a range of values, among them the following are commonly used: number of peaks = 10, change severity $vlength = 1.0$, correlation value $\lambda = 0.0$ and peak change frequency = 5000. In *Scenario 2* no basis landscape is used and the peak type is a simple *cone* with the following expression,

$$\begin{aligned} P(\vec{x}, h_p(t), w_p(t), \vec{c}_p(t)) &= h_p(t) - s_p(t) \|\vec{x} - \vec{c}_p(t)\| \quad \text{where} \\ s_p(t) &= \frac{h_p(t)}{w_p(t)} \quad \text{and} \quad \|\vec{x} - \vec{c}_p(t)\| = \sqrt{\sum_{i=1}^N (x_i - c_{pi})^2} \quad \forall x_i \in \vec{x}, \forall c_{pi} \in \vec{c}_p(t) \end{aligned} \quad (5)$$

where $s_p(t)$ is the slope and $\|\cdot\|$ is the *Euclidean* distance between two N -dimensional vectors, \vec{x} and $\vec{c}_p(t)$. More detailed information on MPB and the rest of the parameters used in this benchmark can be obtained from Branke (2008).

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 Blackwell & Branke (2004a, 2004b) by introducing multi-swarms that 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, some other method can also 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. However, it is then required that the number of peaks is known beforehand. Blackwell (2007) presents self-adapting multi-swarms, which can be created or removed during the PSO process and, therefore, it is not necessary to fix the number of swarms beforehand.

The repulsion between swarms is realized by simply re-initializing the worse of the 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 , $r_{rep} = r_{bas} = X/p^{1/N}$. Detailed information about multi-swarms can be obtained in Blackwell & Branke (2004a, 2004b).

3. The proposed optimization technique for dynamic environments

In this section, we first introduce the multi-swarms with FGBF technique and its application to MPB. The multi-dimensional extension of PSO, the so-called MD PSO, will be detailed next. Finally we show their mutual application over the (extended) MPB.

3.1. FGBF technique

For those problems where dimensional fitness evaluation is possible, Fractional Global Best Formation (FGBF) can be used to avoid the pre-mature 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). Some problems such as data clustering, on the other hand, can exhibit some kind of non-trivial interdependency among the dimensional components of the search space. For these cases, a proper *approximation* for the fractional fitness evaluation, if possible, should be designed to perform FGBF. In either case, FGBF fractionally creates an artificial GB particle, called aGB, at each iteration in a PSO process by selecting the best particle (dimensional) components from the entire swarm. Therefore, especially during the initial steps, aGB 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 the 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 g_{best} 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 g_{best} , the personal best location of the g_{best} particle is replaced by the location of the aGB particle and, since $\hat{y}(t) = y_{g_{best}}(t)$, 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 the native g_{best} and 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 g_{best} (which can be also a former aGB now).

Suppose that for a swarm ξ , FGBF is performed in a PSO process in 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 individual 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 th component of the position of particle a and $f(g_{best}, j)$ be the dimensional fitness score of the j th component of the personal best position of the g_{best} particle. Suppose that all dimensional fitness scores $(f(a, j), \forall a \in [1, S])$ and $(f(g_{best}, j))$ 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 follows:

FGBF in $bPSO$ ($\xi, f(a, j)$)

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

Note that Step 1 along with the computation of $f(a, j)$ depends entirely on the optimization problem. It keeps track of partial fitness contributions from each individual dimensional component of each particle's position (the potential solution). Take for instance the function minimization problem as illustrated in Fig. 1 where 2D space is used for illustration purposes. In the figure, three particles in a swarm are ranked as the 1st (or the g_{best}), the 3rd and the 8th with respect to their proximity to the target position (or the global solution) of some function. Although g_{best} 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 solution via FGBF yields a better (closer) particle than the swarm's native g_{best} . Particularly in Kiranyaz et al. (2010), the usage and merits of FGBF for the optimization of several multi-modal, non-linear (benchmark) functions in high dimensions have been shown where all MD PSO runs with FGBF found the global minimum at the target dimension for all runs, over all functions, regardless of the dimension, swarm size and modality, and without any exception. This is the case where the function

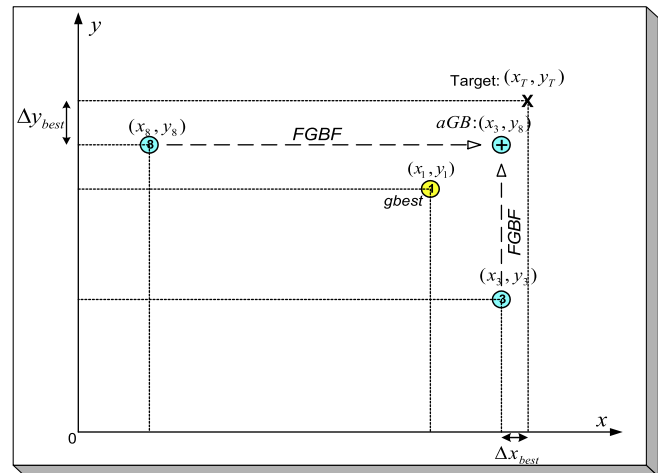


Fig. 1. A sample FGBF operation in 2D space.

to be optimized is known *a priori* and FGBF is easier to use. For those problems where the knowledge of how each fractional dimension contributes to the overall fitness is no longer available (i.e. black-box problems), FGBF can still be adapted with respect to the problem, e.g. in Kiranyaz et al. (2010), see the usage of FGBF over general data clustering where the optimum dimension of the search space (number of clusters) is also unknown and hence MD PSO with the proper application of FGBF is used to find the optimum (number of) clusters in a data space.

3.2. FGBF application for MPB

The previous section introduced the principles of FGBF theory within a $bPSO$ process in a single dimension and referred to some of its applications in other domains, each of which is 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 solution. This is the basic reason of utilizing the multi-swarms along with the FGBF operation. As described in Section 2.3, the mutual repulsion between swarms is performed and for computing the distance between two swarms, we use the distance between the swarms' global best locations. Instead of charged or quantum swarms, FGBF is the mechanism alternatively used to provide necessary diversity and thus to enable peak tracking if peaks' location are changed. We also re-initialize the particle velocities after each environment change to further contribute to the diversity.

A particle with index a in a swarm ξ , represents a potential solution and therefore, the j th component of an N -dimensional point $(x_j, j \in [1, N])$ is stored in its positional component, $x_{a,j}(t)$, at a time t . The aim of the PSO process is to search for the global optimum point, which maximizes $P(\bar{x}, h_p(t), w_p(t), \bar{c}_p(t))$, in other words, finding the global (highest) peak in MPB environment. Recall that in Scenario 2 of MPB the peaks used are cone shaped, as given in (5). Since in (5), $h_p(t)$ and $s_p(t)$ are both set by MPB, finding the highest peak is equivalent to minimizing the $\|\bar{x} - \bar{c}_p(t)\|$ term, yielding $f(a, j) = -(x_j - c_{pj})^2$. Step 3.1 in $bPSO$'s pseudo-code computes the (dimensional) fitness scores $(f(a, j), f(g_{best}, j))$ of the j th components $(x_{a,j}, y_{g_{best},j})$ and in Step 2 of the FGBF process, the dimensional component yielding the 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 g_{best} particle, if they yield 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.

3.3. MD PSO algorithm

Instead of operating in a fixed dimension N , the MD PSO algorithm is designed to seek both positional and dimensional optima within a dimension range, ($D_{\min} \leq d \leq D_{\max}$). In MD PSO each particle has two sets of components, each of which is subjected to one of two independent and consecutive processes. The first one is the regular positional PSO, i.e. the traditional velocity updates and due positional shifts in a d -dimensional search space. The second one is the dimensional PSO, which allows the particle to navigate through dimensions. Accordingly, each particle keeps track of its last position, velocity and personal best position (*pbest*) in a particular dimension so that when it re-visits the same dimension at a later time, it can perform its regular “positional” fly using this information. The dimensional PSO process of each particle may then move the particle to another dimension where it will remember its earlier positional status and keep “flying” within the positional PSO process in this dimension, and so on. The swarm, on the other hand, keeps track of the *gbest* particles in all dimensions, each of which respectively indicates the best (global) position so far achieved and can thus be used in the regular velocity update equation for that dimension. Similarly the dimensional PSO process of each particle uses its personal best dimension in which the personal best fitness score has so far been achieved. Finally, the swarm keeps track of the global best dimension, *dbest*, among all the personal best dimensions. The *gbest* particle in *dbest* dimension represents the optimum solution and dimension, respectively.

The MD PSO process at time (iteration), is represented by the following characteristics:

- $xx_{aj}^{xd_a(t)}(t)$: j th component (dimension) of the position of particle a , in dimension $xd_a(t)$.
- $vx_{aj}^{xd_a(t)}(t)$: j th component (dimension) of the velocity of particle a , in dimension $xd_a(t)$.
- $xy_{aj}^{xd_a(t)}(t)$: j th component (dimension) of the personal best (*pbest*) position of particle a , in dimension $xd_a(t)$.
- $gbest(d)$: Global Best particle index in dimension d .
- $xy_j^d(t)$: j th component (dimension) of the global best position of swarm, in dimension d .
- $xd_a(t)$: current dimension of particle a .
- $vd_a(t)$: dimensional velocity of particle a .
- $\tilde{x}d_a(t)$: personal best dimension of particle a .

Fig. 2 shows sample MD PSO and *b*PSO particles with indices a . *b*PSO particle that is at a (fixed) dimension, $N = 5$, contains only positional components whereas MD PSO particle contains both positional and dimensional components, respectively. In the figure the dimension range for the MD PSO is between 2 and 9, therefore the particle contains eight sets of positional components. In this example the current dimension where the particle with index a resides is 2 ($xd_a(t) = 2$) whereas its personal best dimension is 3 ($\tilde{x}d_a(t) = 3$). Therefore, at time t , a positional PSO update is first performed over the positional elements, $xx_{aj}^{xd_a(t)}(t)$ and then the particle may move to another dimension with respect to the dimensional PSO. Recall that each positional element, $xx_{aj}^{xd_a(t)}(t), j \in \{1, 2\}$, represents a potential solution in the data space of the problem.

Let f denote the fitness function that is to be optimized within a certain dimension range, $[D_{\min}, D_{\max}]$. In accordance with the scope of the current work and without loss of generality assume that the objective is to find the maximum (position) of f in the optimum dimension within a multi-dimensional search space. Assume that the particle a visits (back) the same dimension after T iterations (i.e. $xd_a(t) = xd_a(t+T)$), then the personal best position can be updated in iteration $t+T$ as follows,

$$xy_{aj}^{xd_a(t+T)}(t+T) = \begin{cases} xy_{aj}^{xd_a(t)}(t) \\ \text{if } f(xx_{aj}^{xd_a(t+T)}(t+T)) < f(xy_{aj}^{xd_a(t)}(t)) \\ xx_{aj}^{xd_a(t+T)}(t+T) \\ \text{else} \end{cases} \quad j = 1, 2, \dots, xd_a(t+T) \quad (6)$$

To overcome the necessity to remember the iteration numbers, when a particle has visited a certain dimension last time, the following updates are performed for each dimensional component except $xd_a(t)$, i.e. $j \in [1, d]$ where $d \in [D_{\min}, D_{\max}] - \{xd_a(t)\}$ and for each particle, $a \in [1, S]$,

$$xy_{aj}^d(t) = xy_{aj}^d(t-1), \quad xy_j^d(t) = xy_j^d(t-1), \quad \text{for } \forall d \in [D_{\min}, D_{\max}] - \{xd_a(t)\} \quad (7)$$

Furthermore, the personal best dimension of particle a can be updated in iteration $t+1$ as follows,

$$\tilde{x}d_a(t) = \begin{cases} \tilde{x}d_a(t-1) & \text{if } f(xx_{a\tilde{x}d_a(t)}^{xd_a(t)}(t)) < f(xy_{a\tilde{x}d_a(t-1)}^{xd_a(t-1)}(t-1)) \\ xd_a(t) & \text{else} \end{cases} \quad (8)$$

Recall that $gbest(d)$ is the index of the Global Best particle in dimension d and so $xy_j^d(t) = xy_{gbest(d)}^d(t)$. For a particular iteration t , and for a particle with index $a \in [1, S]$, first the positional components are updated in its current dimension, $xd_a(t)$ and then the dimensional

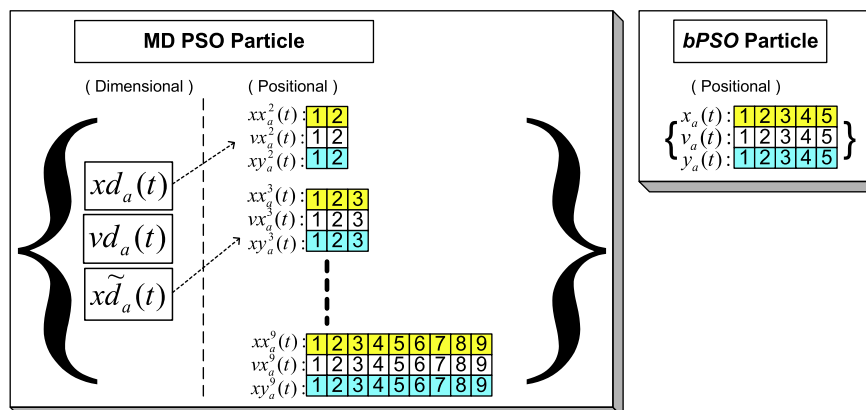


Fig. 2. Sample MD PSO (right) vs. *b*PSO (left) particle structures. For MD PSO $\{D_{\min} = 2, D_{\max} = 9\}$ and at the current time t , $xd_a(t) = 2$ and $\tilde{x}d_a(t) = 3$. For *b*PSO $N = 5$.

update is performed to determine its next dimension, $xd_a(t+1)$. The positional update is performed for each dimension component, $j \in [1, xd_a(t)]$, as follows:

$$\begin{aligned} vx_{aj}^{xd_a(t)}(t+1) &= w(t)vx_{aj}^{xd_a(t)}(t) + c_1r_{1,j}(t)(xy_{aj}^{xd_a(t)}(t) - vx_{aj}^{xd_a(t)}(t)) \\ &\quad + c_2r_{2,j}(t)(\dot{xy}_{aj}^{xd_a(t)}(t) - vx_{aj}^{xd_a(t)}(t)) \\ xx_{aj}^{xd_a(t)}(t+1) &= vx_{aj}^{xd_a(t)}(t) + C_{vx}[vx_{aj}^{xd_a(t)}(t+1), \{V_{\min}, V_{\max}\}] \\ xx_{aj}^{xd_a(t)}(t+1) &\leftarrow C_{xx}[xx_{aj}^{xd_a(t)}(t+1), \{X_{\min}, X_{\max}\}] \end{aligned} \quad (9)$$

where $C_{xx}[\dots] \equiv C_{vx}[\dots]$ are the clamping operators applied over each positional component, $xx_{aj}^{xd_a(t)}$ and $vx_{aj}^{xd_a(t)}$. $C_{xx}[\dots]$ may or may not be applied depending on the optimization problem but $C_{vx}[\dots]$ is basically needed to avoid exploding. Each operator can be applied in two different ways, such as,

$$C_{xx}[xx_{aj}^d(t), \{X_{\min}, X_{\max}\}] = \begin{cases} xx_{aj}^d(t) & \text{if } X_{\min} \leq xx_{aj}^d(t) \leq X_{\max} \\ X_{\min} & \text{if } xx_{aj}^d(t) < X_{\min} \\ X_{\max} & \text{if } xx_{aj}^d(t) > X_{\max} \end{cases} \quad (10.a)$$

$$C_{xx}[xx_{aj}^d(t), \{X_{\min}, X_{\max}\}] = \begin{cases} xx_{aj}^d(t) & \text{if } X_{\min} \leq xx_{aj}^d(t) \leq X_{\max} \\ U(X_{\min}, X_{\max}) & \text{else} \end{cases} \quad (10.b)$$

where option (a) is a simple thresholding to the range limits and (b) re-initializes randomly the j th positional component ($j \leq d$).

Note that the particle's new position, $xx_a^{xd_a(t)}(t+1)$, will still be in the same dimension, $xd_a(t)$; however, the particle may jump to another dimension afterwards. Therefore, for all the other dimensions in the dimensional range except $xd_a(t)$, i.e. $\forall d \in \{D_{\min}, \dots, D_{\max}\} - \{xd_a(t)\}$, the following updates are necessary for each dimensional component, $j \in [1, \dots, d]$ and for each particle, $a \in [1, S]$:

$$\begin{aligned} vx_{aj}^d(t+1) &= vx_{aj}^d(t), xx_{aj}^d(t+1) = xx_{aj}^d(t) \\ \text{for } \forall d \in [D_{\min}, D_{\max}] - \{xd_a(t)\} \end{aligned} \quad (11)$$

The dimensional updates are computed with the following equations:

$$\begin{aligned} vd_a(t+1) &= \lfloor vd_a(t) + c_1r_1(t)(\tilde{xd}_a(t) - xd_a(t)) \\ &\quad + c_2r_2(t)(d_{best} - xd_a(t)) \rfloor \\ xd_a(t+1) &= xd_a(t) + C_{vd}[vd_a(t+1), \{VD_{\min}, VD_{\max}\}] \\ xd_a(t+1) &\leftarrow C_{xd}[xd_a(t+1), \{D_{\min}, D_{\max}\}] \end{aligned} \quad (12)$$

where $\lfloor \cdot \rfloor$ is the floor operator, $C_{xd}[\dots]$ and $C_{vd}[\dots]$ are the clamping operators applied over dimensional components, $xd_a(t)$ and $vd_a(t)$, respectively. As in (2), we employ the inertia weight for positional velocity update; however, we have witnessed no benefit of using it for dimensional PSO, and hence we left it out of (12) for the sake of simplicity. $C_{vd}[\dots]$ is similar to $C_{vx}[\dots]$, which is basically applied to avoid exploding and we use the basic thresholding for this, expressed as follows:

$$C_{vd}[vd_a(t), \{VD_{\min}, VD_{\max}\}] = \begin{cases} vd_a(t) & \text{if } VD_{\min} \leq vd_a(t) \leq VD_{\max} \\ VD_{\min} & \text{if } vd_a(t) < VD_{\min} \\ VD_{\max} & \text{if } vd_a(t) > VD_{\max} \end{cases} \quad (13)$$

$C_{xd}[\dots]$, on the other hand, is a mandatory clamping operator, which keeps the dimensional jumps within the dimension range of the problem, $[D_{\min}, D_{\max}]$. Furthermore within $C_{xd}[\dots]$, an optional in-flow buffering mechanism can also be implemented. This can be a desired property, which allows only sufficient number of particles in a certain dimension and thus avoids the redundancy. Particularly d_{best} and dimensions within the close proximity have a natural

attraction and without such buffering mechanism, the majority of swarm particles may be hosted within this local neighborhood and hence other dimensions might encounter a severe depletion. To prevent this, the buffering mechanism should control the in-flow of the particles (by the dimensional velocity updates) to a particular dimension. On some early *bPSO* implementations over problems with low (and fixed) dimensions, 15–20 particles were usually sufficient for a successful operation. However, in high dimensions this may not be so since more particles are usually needed as the dimension increases. Therefore, we empirically set the limit to be proportional to the solution space dimension and not less than 15. At time t , let $P_d(t)$ be the number of particles in dimension d . $C_{xd}[\dots]$ can then be expressed with the (optional) buffering mechanism as follows:

$$C_{xd}[xd_a(t), \{D_{\min}, D_{\max}\}] = \begin{cases} xd_a(t-1) & \text{if } P_{xd_a(t)}(t) \geq \max(15, xd_a(t)) \\ xd_a(t-1) & \text{if } xd_a(t) < D_{\min} \\ xd_a(t-1) & \text{if } xd_a(t) > D_{\max} \\ xd_a(t) & \text{else} \end{cases} \quad (14)$$

In short, the clamping and buffering operator, $C_{xd}[\dots]$, allows a dimensional jump only if the target dimension is within dimensional range and has space for a newcomer. Accordingly, the general pseudo-code of the MD PSO method can be expressed as follows:

MD PSO (termination criteria: $\{IterNo, \varepsilon_C, \dots\}$)

1. For $\forall a \in [1, S]$ do:
 - 1.1. Randomize $xd_a(1)$, $vd_a(1)$
 - 1.2. Initialize $xd_a(0) = xd_a(1)$
 - 1.3. For $\forall d \in \{D_{\min}, D_{\max}\}$ do:
 - 1.3.1. Randomize $xx_a^d(1)$, $xv_a^d(1)$
 - 1.3.2. Initialize $xy_a^d(0) = xx_a^d(1)$
 - 1.3.3. Initialize $x\dot{y}_a^d(0) = xv_a^d(1)$
 - 1.4. End For.
 2. End For.
 3. For $\forall t \in [1, IterNo]$ do:
 - 3.1. For $\forall a \in [1, S]$ do:
 - 3.1.1. If $(f(xx_a^{xd_a(t)}(t)) > f(xy_a^{xd_a(t)}(t-1)))$ then do:
 - 3.1.1.1. $xy_a^{xd_a(t)}(t) = xx_a^{xd_a(t)}(t)$
 - 3.1.1.2. Update $\tilde{xd}_a(t)$ according to (8)
 - 3.1.2. Else $xy_a^{xd_a(t)}(t) = xy_a^{xd_a(t)}(t-1)$
 - 3.1.3. End If
 - 3.1.4. If $(f(xx_a^{xd_a(t)}(t)) > \max(f(x\dot{y}_a^{xd_a(t)}(t-1)), \max_{1 \leq p < a} (f(xx_p^{xd_a(t)}(t))))$ then do:
 - 3.1.4.1. $g_{best}(xd_a(t)) = a$
 - 3.1.4.2. If $(f(xx_a^{xd_a(t)}(t)) > f(x\dot{y}_a^{d_{best}}(t-1)))$ then $d_{best} = xd_a(t)$
 - 3.1.5. End If
 - 3.1.6. Do updates in other dimensions according to (7)
 - 3.2. End For.
 - 3.3. If the termination criteria are met, then **Stop**.
 - 3.4. For $\forall a \in [1, S]$ do:
 - 3.4.1. For $\forall j \in [1, xd_a(t)]$ do:
 - 3.4.1.1. Compute $vx_{aj}^{xd_a(t)}(t+1)$ and $xx_{aj}^{xd_a(t)}(t+1)$ using (9)
 - 3.4.1.2. Update velocities and locations in other dimensions using (11)
 - 3.4.2. End For.
 - 3.4.3. Compute $vd_a(t+1)$ and $xd_a(t+1)$ using (12)
 - 3.5. End For.
4. End For.

Once the MD PSO process terminates, the optimum solution will be x_j^{dbest} in the optimum dimension, $dbest$, achieved by the particle with the index $gbest$ ($dbest$) and finally the best (fitness) score achieved will naturally be $f(x_j^{dbest})$. Note that MD PSO is only a natural extension or a generic form of the basic PSO, as it searches for both optimum dimension and solution (in the optimum dimension). Note that this is not a 'superiority' in terms of convergence, rather the ability of searching the optimum solution space dimension while searching for the positional optimum, in a simultaneous way. In other words, the basic PSO is a MD PSO process in a fixed dimension, which basically means that $bPSO$ is identical to MD PSO for a particular (fixed) dimension. However, contrary to $bPSO$, due to its ability to simultaneously search for both optimum dimension and solution, MD PSO can conveniently be used in many applications where the optimum dimension is unknown. For instance in Kiranyaz et al. (2010) MD PSO (with FGBF) has been used to find the true number of clusters in a complex data space. This is obviously a tremendous advantage since with a traditional approach such as K -means, the number of clusters, K , has to be given *a priori* and this may not be possible for many complex problems. In another work (Ince et al., 2009), MD PSO has been used for the automatic design of Artificial Neural Networks (ANNs) by evolving to the optimal network configuration(s) for a particular problem. Thus MD PSO can seek for the positional optimum in the error space and dimensional optimum in the architecture space. The optimum dimension converged at the end of a MD PSO process corresponds to the best ANN configuration (or the most appropriate ANN for that problem) where the trained network parameters (connections, weights and biases) can then be resolved from the positional optimum reached in that dimension. This presents a significant advantage over many traditional training methods such as the well-known back-propagation algorithm, which can only be used to train a single ANN with a fixed configuration.

3.4. The proposed techniques over multi-dimensional MPB

For testing the proposed multi-dimensional optimization technique, we extended Branke's MPB into a multi-dimensional version, in which there exists many search space dimensions within a dimensional range $[D_{min}, D_{max}]$, and the optimal dimension changes over time in addition to the dynamic nature of the conventional MPB. Peak locations in different dimensions share the common peak center coordinates and thus such an extension further allows exploitation of the information gathered in other search space dimensions.

The multi-dimensional extension of the MPB is simple. The initialization and changes of peak locations must now be done in the highest possible search space dimension, D_{max} . Locations in the other dimensions can be obtained simply by leaving out the redundant coordinates (non-existing dimensions). The optimal dimension is chosen randomly every time the environment is changed. Therefore, the fitness function with m peaks in multi-dimensional environment can be expressed as,

$$F(\vec{x}^d, t) = \max \left(B(\vec{x}^d), \max_{p=1..m} P(\vec{x}^d, d, h_p(t), w_p(t), \vec{c}_p^d(t)) \right) \quad (15)$$

where $d \in [D_{min}, D_{max}]$ is the dimension of position \vec{x}^d and $\vec{c}_p^d(t)$ refers to the first d coordinates (dimensions) of the peak center location. A cone peak is now expressed as follows:

$$P(\vec{x}^d, h_p(t), w_p(t), \vec{c}_p^d(t)) = h_p(t) - s_p(t) * \left\| \vec{x}^d - \vec{c}_p^d(t) \right\| / d - (D_{opt} - d)^2 \quad \text{where} \\ s_p(t) = \frac{h_p(t)}{w_p(t)} \quad \text{and} \quad \left\| \vec{x}^d - \vec{c}_p^d(t) \right\| \\ = \sqrt{\sum_{i=1}^d (x_i^d - c_{pi}^d)^2} \quad \forall x_i^d \in \vec{x}^d, \forall c_{pi}^d \in \vec{c}_p^d(t) \quad (16)$$

where D_{opt} is the current optimal dimension. If compared with expression (5), now for all non-optimal dimensions a penalty term $(D_{opt} - d)^2$ is subtracted from the whole environment. In addition to that the peak slopes are scaled by the term $1/d$. The purpose of this scaling is to prevent the benchmark from favoring the lower dimensions. Otherwise a solution, whose every coordinate differs from the optimum by 1.0 would be a lot better solution in a lower dimension as the Euclidian distance is used.

Similar to the uni-dimensional (PSO) case, each positional component $xx_a^d(t)$ of MD PSO particle represents a potential solution in dimension d . The only difference is that now the dimensionality of the optimal solution is not known beforehand, but it can vary within the defined range. Even a single particle can provide potential solutions in different dimensions as it makes inter-dimensional passes as a result of MD PSO process. Our dynamic multi-dimensional optimization algorithm combines multi-swarms and FGBF with MD PSO. As in the different dimensions the common coordinates of the peak locations are the same, it does not seem purposeful for two swarms to converge to the same peak in different dimensions. Therefore, the mutual repulsion between swarms is extended to affect swarms that are in different dimensions. Obviously, only the common coordinates are considered when the swarm distance is computed.

FGBF naturally exploits information gathered in other dimensions. When the aGB particle is created, FGBF algorithm is not limited to use dimensional components from only those particles which are in a certain dimension, but it can combine dimensional coordinates of particles in different dimensions. Note that as we still use the dimensional fitness score, $f(a, j) = -(x_j - c_{pj})^2$, the common coordinates of the positional components of the aGB particle created in different search space dimensions, $d \in [D_{min}, D_{max}]$, shall be the same. In other words, it is not necessary to create the positional components of the aGB particle from scratch in every search space dimension $d \in [D_{min}, D_{max}]$, instead in dimensions higher than D_{min} , only one (new) coordinate (dimension) to the aGB particle is created and added. Note also that it is still possible that in some search space dimensions aGB beats the native $gbest$ particle, while in other dimensions it does not. In the multi-dimensional version also the dimension and dimensional velocity of each particle are re-initialized after an environmental change in addition to the particle velocities in each dimension.

4. Experimental results

An extensive set of experiments was conducted over both conventional (uni-dimensional) MPB and the extended (multi-dimensional) MPB and the results will be presented in the following subsections.

4.1. Results on conventional MPB

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 peaks and the swarm peaks. To investigate how FGBF and multi-swarms individually contribute to the results, we also made experiments without using either of them.

Fig. 3 presents the current error plot, which shows the difference between the global maximum and the current best result during the first 80,000 function evaluations, when 10 swarms each with four 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 evaluations, it causes the

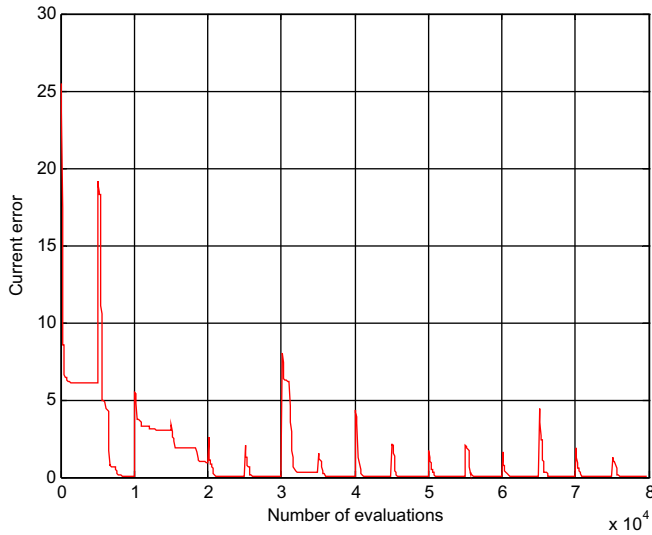


Fig. 3. Current error at the beginning of a run.

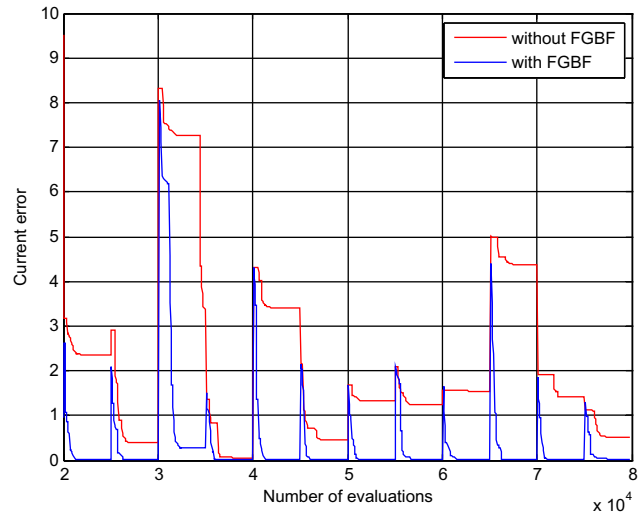


Fig. 5. Effect of FGBF on results.

results to temporarily deteriorate. However, it is clear that after environment changes the results are better than at 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 the algorithms behavior. First of all, after the first few environment changes the algorithm has not yet been behaving as well as later. This is because the swarms have not 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.

In Figs. 4 and 5 the contributions of multi-swarms with FGBF are demonstrated. The algorithm is run on MPB applying the same environment changes, first with both using multi-swarms and FGBF, then without multi-swarms and finally without FGBF. The 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.

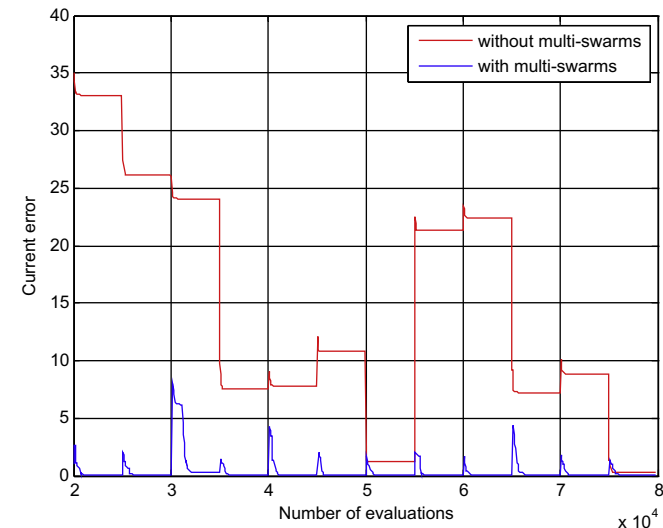


Fig. 4. Effect of multi-swarms on results.

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.

For comparative evaluations, we selected five of the state-of-the-art methods, which use the same benchmark system, the MPB with the same settings. The best MPB results published so far by these competing methods are listed in Table 1.

The overall best results have been achieved by the Extremal Optimization algorithm (Moser & Hendtlass, 2007); however, this algorithm is specially and only 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 technique in terms of the offline error are listed in Table 2. Each result given is the average of 50 runs, where each run consists of 500,000 function evaluations. As expected the best results are achieved when 10 swarms are used. Four particles in a swarm turned out to be the best setting. Between the two FGBF modes, better results are obtained when the swarm peak is used instead of the peak closest to each particle.

4.2. Results on multi-dimensional MPB

On the extended MPB we conducted experiments with settings similar to those used in the fixed dimension except that the change

Table 1 Best results on MPB up to date.

Source	Base algorithm	Offline error
Blackwell and Branke (2004a)	PSO	2.16 ± 0.06
Li et al. (2006)	PSO	1.93 ± 0.06
Mendes and Mohais (2005)	Differential evolution	1.75 ± 0.03
Blackwell and Branke (2004b)	PSO	1.75 ± 0.06
Moser and Hendtlass (2007)	Extremal optimization	0.66 ± 0.02

Table 2
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	1.03 ± 0.35	1.37 ± 0.50
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

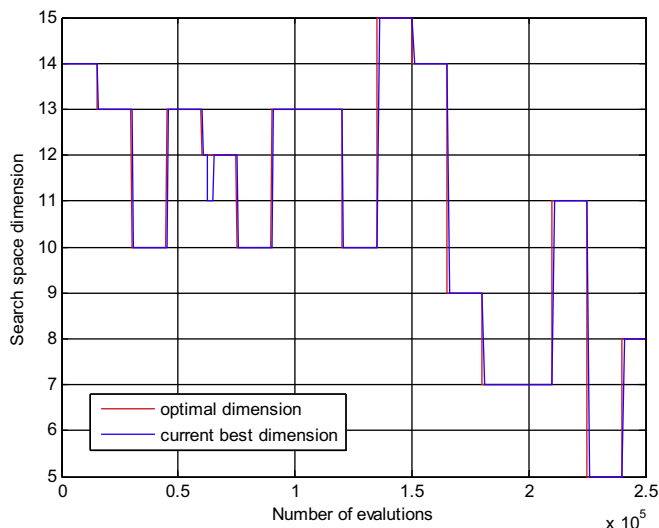


Fig. 6. Optimum dimension tracking in a MD PSO run.

frequency used is set to 15,000. The search space dimension range used is $d \in [5, 15]$. Fig. 6 shows how the global optimal dimension changes over time and how MD PSO is tracking these changes. Current best dimension represents the dimension, where the best solution is achieved among all swarms' *dbest* dimensions. Ten multi-swarms are used with seven particles in each. FGBF is used with the swarm peak mode. It can be seen that the algorithm always finds the optimal dimension, even though the difference in peak heights between the optimal dimension and its neighbor dimensions is quite insignificant (≈ 1) compared to the peak heights (30–70). Fig. 7 shows how the current error behaves during the first 250,000 evaluations, when the same settings are used. It can be seen that the algorithm behavior is similar to the uni-dimensional case, but now the initial converging phase, when the algorithm is not yet behaving at its best is longer. Similarly it takes a longer time to regain the optimal behavior if follow-up of some peaks is lost for some reason (it is, for example, possible that higher peaks hide other lower peaks under them).

Figs. 8 and 9 illustrate the effect of using multi-swarms on the results. Without multi-swarms the number of particles is set to 70. Fig. 8 shows that a single swarm can also find the optimal dimension easily; however, as in the uni-dimensional case, without use of multi-swarms, the optimal peak can be found only if it happens to be the peak to which the swarm is converging. This can be seen in Fig. 9. During the initial converging phase of the multi-swarm algorithm results with and without multi-swarms are similar. This indicates that both algorithms initially converge to the same peak (highest) and as a result of the first few environ-

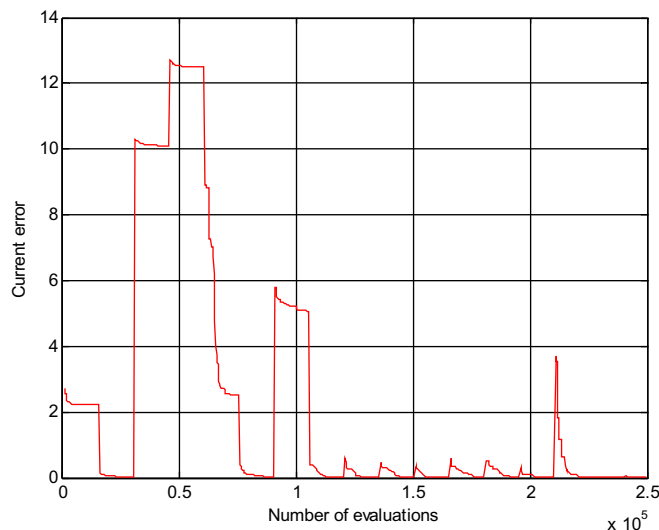


Fig. 7. Current error at the beginning of a MD PSO run.

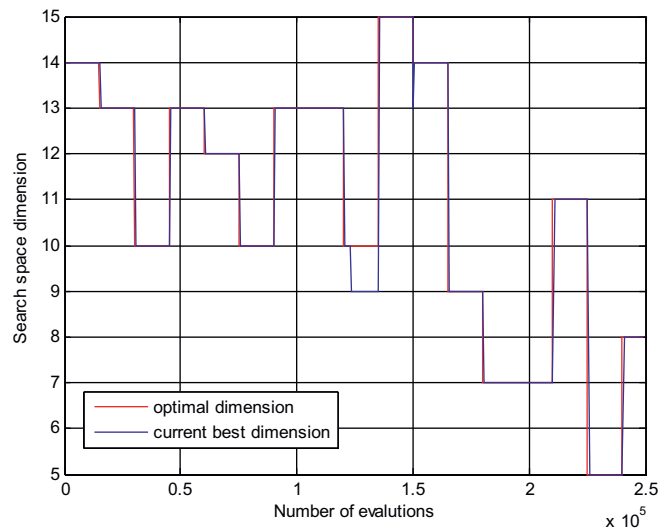


Fig. 8. Optimum dimension tracking without multi-swarms in a MD PSO run.

mental changes some peaks that are not yet discovered by multi-swarms become the highest.

Figs. 10 and 11 illustrate similarly the effect of FGBF on the results. In Fig. 10 it can be seen that without FGBF the algorithm has severe problems in tracking the optimal dimension. In this case, it loses the benefit of exploiting the natural diversity among the dimensional components and also it is not able to exploit information gathered in other dimensions. Therefore, even if some particles visit the optimal dimension, they cannot track the global peak fast enough that they would hence surpass the best results in other dimensions. Therefore, the algorithm gets trapped in some sub-optimum dimension where it happens to find the best results in an early phase. Such reasons also cause the current error to be generally higher without FGBF, as can be seen in Fig. 11.

The numerical results in terms of offline error are given in Table 3. Each result given is the average of 50 runs, where each run consists of 500,000 function evaluations. As in the uni-modal case, best results are achieved when the number of swarms is equal to the number of peaks, which is 10. Interestingly when the swarm peak mode is used the optimal number of particles becomes seven

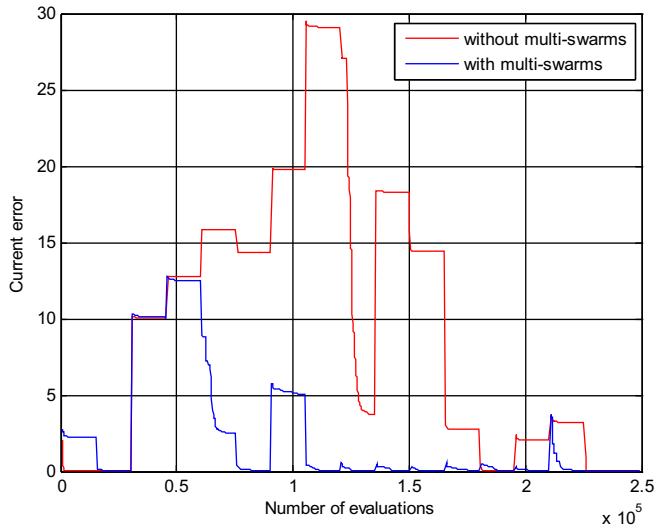


Fig. 9. Effect of multi-swarms on the performance.

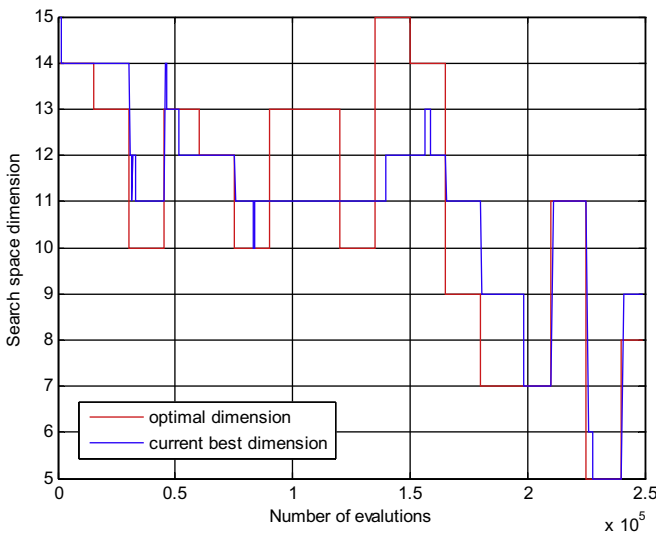


Fig. 10. Optimum dimension tracking without FGBF in a MD PSO run.

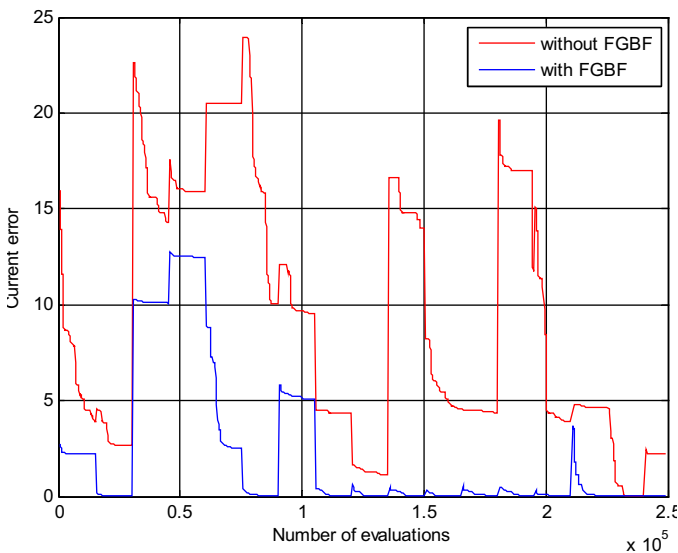


Fig. 11. Effect of FGBF on the performance.

Table 3
Offline error on extended MPB.

No. of swarms	No. of particles	Swarm peak	Current peak
10	4	2.01 ± 0.98	3.29 ± 1.44
10	5	1.77 ± 0.83	3.41 ± 1.69
10	6	1.79 ± 0.98	3.64 ± 1.60
10	7	1.69 ± 0.75	3.71 ± 1.74
10	8	1.84 ± 0.97	4.21 ± 1.83
10	10	1.96 ± 0.94	4.20 ± 2.03
8	7	1.79 ± 0.91	3.72 ± 1.86
9	7	1.83 ± 0.84	4.30 ± 2.15
11	7	1.75 ± 0.91	3.52 ± 1.40
12	7	2.03 ± 0.97	4.01 ± 1.97

while with the current peak mode it is still four. Note that these results cannot be directly compared with the results on the conventional MPB since the objective function of the multi-dimensional MPB is somewhat different.

5. Conclusions

In this paper, we presented two PSO techniques, namely, FGBF with multi-swarms and MD PSO for an efficient and robust optimization over the dynamic systems. Both MD PSO and FGBF have been successfully used in static optimization problems particularly as a cure to common drawbacks of the family of PSO methods such as *a priori* fixation of the search space dimension and pre-mature convergence to local optima. MD PSO efficiently addresses the former drawback by defining a new particle structure and embedding the ability of dimensional navigation into the core of the process. It basically allows particles to make inter-dimensional ‘passes’ with a dedicated PSO process whilst performing regular positional updates in every dimension they visit. Although the ability of determining the optimum dimension where the global solution exists is gained with MD PSO, its convergence performance is still limited to the same level as *bPSO*, which suffers from the lack of diversity among particles. This leads to a pre-mature convergence to local optima especially when multi-modal problems are optimized in high dimensions. 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 adapted in this paper addresses this problem by collecting the best components and fractionally creating an *aGB* particle that has the potential to be a better “guide” than the swarm’s native *gbest* particle. On MPB we do not expect 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 quite 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 conventional MPB with common settings used (i.e. *Scenario 2*) clearly indicate the superiority of the proposed technique over other PSO-based methods. To make the benchmark more generic for real-world applications where the optimum dimension can be unknown too, MPB is extended to a multi-dimensional system in which there is a certain amount of dependency among dimensions. Note that without such dependency embedded, the benchmark would be just a bunch of independent MPBs in different dimensions and thus a distinct and independent optimization process would be sufficient for each dimension. Recall that the convergence behavior of both *bPSO* and MD PSO is the same since MD PSO is only an extension of PSO for the multi-dimensional search. The performance of

both methods degrades with the increasing modality and dimensionality due to the reasons mentioned earlier. When performed with FGBF and multi-swarms, MD PSO exhibits both global convergence ability and an impressive speed gain so that their mutual performance surpasses *bPSO* by several magnitudes. The experiments conducted over the extended MPB approve that the proposed MD PSO technique with multi-swarms and FGBF always finds and tracks the optimum dimension where the global peak resides. On both (conventional and extended) MPBs, the proposed techniques generally find and track the global peak, yet they can occasionally converge to a near-optimum peak, particularly if the height difference happens to be insignificant.

Overall, the proposed techniques fundamentally upgrade the particle structure and the swarm guidance, both of which accomplish substantial improvements in terms of speed and accuracy. Both techniques are modular and independent from each other, i.e. one can be performed without the other whilst other PSO methods/variants can also be performed conveniently with (either of) them.

References

- Abraham, A., Das, S., & Roy, S. (2007). Swarm intelligence algorithms for data clustering. In *Soft computing for knowledge discovery and data mining book, Part IV* (pp. 279–313).
- Angeline, P. J. (1997). Tracking extrema in dynamic environments. In *Proceedings of the 6th conference on evolutionary programming* (pp. 335–345). Springer-Verlag.
- Bäck, T. (1998). On the behaviour of evolutionary algorithms in dynamic environments. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 446–451).
- Bäck, T., & Kursawe, F. (1995). Evolutionary algorithms for fuzzy logic: A brief overview. *Fuzzy logic and soft computing* (pp. 3–10). Singapore: World Scientific.
- Bäck, T., & Schwefel, H. P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolution on Computers, 1*, 1–23.
- Blackwell, T. M. (2007). Particle swarm optimization in dynamic environments. *Evolutionary Computation in Dynamic and Uncertain Environments, Studies in Computational Intelligence, 51*, 29–49.
- Blackwell, T. M., & Branke, J. (2004a). Multi-swarm optimization in dynamic environments. *Applications of evolutionary computation* (Vol. 3005, pp. 489–500). Springer.
- Blackwell, T. M., & Branke, J. (2004b). Multi-swarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation, 10*(4), 51–58.
- Branke, J. (2008). Moving Peaks Benchmark. <http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/>, viewed 26/06/08.
- Chen, X., & Li, Y. (2007). A modified PSO structure resulting in high exploration ability with convergence guaranteed. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 37*(5), 1271–1289.
- Chen, Y.-P., Peng, W.-C., & Jian, M.-C. (2007). Particle swarm optimization with recombination and dynamic linkage discovery. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 37*(6), 1460–1470.
- Christopher, K. M., & Seppi, K. D. (2004). The Kalman swarm. A new approach to particle motion in swarm optimization. In *Proceedings of the genetic and evolutionary computation conference* (pp. 140–150). GECCO.
- Clerc, M. (1999). The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In *Proceedings of the IEEE congress on evolutionary computation* (Vol. 3, pp. 1951–1957).
- Eberhart, R., & Shi, Y. (2001). Tracking and optimizing dynamic systems with particle swarms. In *Proceedings of computational evolution conference (CEC 2001)* (pp. 94–100). NJ, US.
- Eberhart, R., Simpson, P., & Dobbins, R. (1996). *Computational intelligence. PC tools*. Boston, MA, USA: Academic Press, Inc..
- Engelbrecht, A. P. (2005). *Fundamentals of computational swarm intelligence*. John Wiley & Sons.
- Esquivel, S. C., & Coello, C. A. (2003). On the use of particle swarm optimization with multimodal functions. *IEEE Transactions on Evolutionary Computation, 2*, 1130–1136.
- Fayyad, U. M., Shapire, G. P., Smyth, P., & Uthurusamy, R. (1996). *Advances in knowledge discovery and data mining*. Cambridge, MA: MIT Press.
- Goldberg, D. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.
- Higashi, H., & Iba, H. (2003). Particle swarm optimization with gaussian mutation. In *Proceedings of the IEEE swarm intelligence symposium* (pp. 72–79).
- Ince, T., Kiranyaz, S., & Gabbouj, M. (2009). A generic and robust system for automated patient-specific classification of electrocardiogram signals. *IEEE Transactions on Biomedical Engineering, 56*(5), 1415–1426.
- Janson, S., & Middendorf, M. (2005). A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 35*(6), 1272–1282.
- Kaewkamnerdpong, B., & Bentley, P. J. (2005). Perceptive particle swarm optimization: An investigation. In *Proceedings of the IEEE swarm intelligence symposium* (pp. 169–176). California.
- Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks* (Vol. 4, pp. 1942–1948). Perth, Australia.
- Kiranyaz, S., Ince, T., Yildirim, A., & Gabbouj, M. (2010). Fractional particle swarm optimization in multi-dimensional search space. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 40*(2), 298–319.
- Koza, J. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, Massachusetts: MIT Press.
- Krohling, R. A., & Coelho, L. S. (2006). Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 36*(6), 1407–1416.
- Liang, J. J., & Qin, A. K. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions On Evolutionary Computation, 10*(3), 281–295.
- Li, X., Branke, J., & Blackwell, T. (2006). Particle swarm with speciation and adaptation in a dynamic environment. In *Proceedings of genetic and evolutionary computation conference* (pp. 51–58). Seattle Washington.
- Lovberg, M. (2002). *Improving particle swarm optimization by hybridization of stochastic search heuristics and self-organized criticality*. MSc thesis. Department of Computer Science, University of Aarhus, Denmark.
- Lovberg, M., & Krink, T. (2002). Extending particle swarm optimisers with self-organized criticality. *Proceedings of the IEEE Congress on Evolutionary Computation, 2*, 1588–1593.
- Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation, 8*(3), 204–210.
- Mendes, R., & Mohais, A. (2005). DynDE: A differential evolution for dynamic optimization problems. *IEEE congress on evolutionary computation*, 2808–2815.
- Moser, I., & Hendtlass, T. (2007). A simple and efficient multi-component algorithm for solving dynamic function optimisation problems. *IEEE Congress on Evolutionary Computation*, 252–259.
- Omran, M. G., Salman, A., & Engelbrecht, A. P. (2006). *Particle swarm optimization for pattern recognition and image processing*. Berlin: Springer.
- Peng, B., Reynolds, R. G., & Brewster, J. (2003). Cultural swarms. *Proceedings of the IEEE Congress on Evolutionary Computation*, 3, 1965–1971.
- Peram, T., Veeramachaneni, K., & Mohan, C. K. (2003). Fitness-distance-ratio based particle swarm optimization. In *Proceedings of the IEEE Swarm Intelligence Symposium* (pp. 174–181). IEEE Press.
- Ratnaweera, A. C., Halgamuge, S. K., & Watson, H. C. (2002). Particle swarm optimiser with time varying acceleration coefficients. In *Proceedings of the international conference on soft computing and intelligent systems* (pp. 240–255).
- Ratnaweera, A. C., Halgamuge, S. K., & Watson, H. C. (2003). Particle swarm optimization with self-adaptive acceleration coefficients. In *Proceedings of the first international conference on fuzzy systems and knowledge discovery* (pp. 264–268).
- Richards, M., & Ventura, D. (2003). Dynamic sociometry in particle swarm optimization. In *Proceedings of the sixth international conference on computational intelligence and natural computing* (pp. 1557–1560). North Carolina.
- Riget, J., & Vesterstrom, J. S. (2002). *A diversity-guided particle swarm optimizer – the ARPSO, Technical report*. Department of Computer Science, University of Aarhus.
- Shi, Y., & Eberhart, R. C. (1998). A modified particle swarm optimizer. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 69–73).
- Shi, Y., & Eberhart, R. C. (2001). Fuzzy adaptive particle swarm optimization. *Proceedings of the IEEE congress on evolutionary computation* (Vol. 1, pp. 101–106). IEEE Press.
- Suganthan, P. N. (1999). Particle swarm optimiser with neighborhood operator. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 1958–1962). IEEE Press.
- Van den Bergh, F. (2002). *An analysis of particle swarm optimizers, PhD thesis*. Department of Computer Science, University of Pretoria, Pretoria, South Africa.
- Van den Bergh, F., & Engelbrecht, A. P. (2002). A new locally convergent particle swarm optimizer. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 96–101.
- Van den Bergh, F., & Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation, 3*, 225–239.
- Wilson, E. O. (1975). *Sociobiology: The new synthesis*. Cambridge, MA: Belknap Press.
- Xie, X., Zhang, W., & Yang, Z. (2002). Adaptive particle swarm optimization on individual level. In *Proceedings of the sixth international conference on signal processing* (Vol. 2, pp. 1215–1218).
- Xie, X., Zhang, W., & Yang, Z. (2002a). A dissipative particle swarm optimization. *Proceedings of the IEEE Congress on Evolutionary Computation, 2*, 1456–1461.
- Xie, X., Zhang, W., & Yang, Z. (2002c). Hybrid particle swarm optimizer with mass extinction. *Proceedings of the International Conference on Communication, Circuits and Systems, 2*, 1170–1173.
- Yasuda, K., Ide, A., & Iwasaki, N. (2003). Adaptive particle swarm optimization. *Proceedings of the IEEE international conference on Systems, Man, and Cybernetics, 2*, 1554–1559.
- Zhang, W.-J., & Xie, X.-F. (2003). DEPSO: Hybrid particle swarm with differential evolution operator. *Proceedings of the IEEE International Conference on System, Man, and Cybernetics, 4*, 3816–3821.