Fractional Particle Swarm Optimization in Multi-Dimensional Search Space

Serkan Kiranyaz, Turker Ince, Alper Yildirim and Moncef Gabbouj

Abstract-- **In this paper, we propose two novel techniques, which successfully address several major problems in the field of Particle Swarm Optimization (PSO) and promise a significant breakthrough over complex, multi-modal optimization problems at high dimensions. The first one, so-called Multi-Dimensional (MD) PSO, 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 multidimensional search space where the optimum dimension is unknown, swarm particles can seek both positional and dimensional optima. This eventually removes the necessity of setting a fixed dimension *a priori*, which is a common drawback for the family of swarm optimizers. Nevertheless, MD PSO is still susceptible to premature convergences due to lack of divergence. Among many PSO variants in the literature, none yields a robust solution especially over multi-modal, complex problems at high dimensions. To address this problem we propose Fractional Global Best Formation (FGBF) technique, which 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. We investigated both individual and mutual applications of the proposed techniques over two well-known domains, nonlinear function minimization and data clustering. An extensive set of experiments show that in both application domains, MD PSO with FGBF exhibits an impressive speed gain and converges to the global optima at the true dimension regardless of the search space dimension, swarm size and the complexity of the problem.**

Index Terms—**Particle Swarm Optimization, multi-dimensional search, Fractional Global Best Formation**

I. INTRODUCTION

THE behavior of a single organism in a swarm is often insignificant but their collective and social behavior is of paramount importance. The particle swarm optimization (PSO) was introduced by Kennedy and Eberhart [27] in 1995 as a population based stochastic search and optimization process. It is originated from the computer simulation of the individuals (particles or living organisms) in a bird flock or fish school [53], which basically show a natural behavior when they search for some target (e.g. food). The goal is, therefore, to converge to the global optima of some multi-dimensional and possibly nonlinear function or system. Henceforth, PSO follows the same path of other evolutionary algorithms (EAs) [4] such as Genetic Algorithm (GA) [18], Genetic Programming (GP) [28], Evolution Strategies (ES), [5] and Evolutionary Programming (EP), [16]. The common point of all is that EAs are in 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.

In a PSO process, a swarm of particles (or agents), each of which represent 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 [27]) and the PSO process also keeps track of the global best solution so

far achieved by the swarm with its particle index (the so called *gbest* in [27]). So during their journey with discrete time iterations, the velocity of each agent in the next iteration is computed by the best position of the swarm (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 position of the agent in the next iteration.

As a stochastic search algorithm in multi-dimensional (MD) search space, PSO exhibits some major problems similar to the aforementioned EAs. The first one is due to the fact that any stochastic optimization technique depends on the parameters of the optimization problem where it is applied and variation of these parameters significantly affects the performance of the algorithm. This problem is a crucial one for PSO where parameter variations may result in large performance shifts [33]. The second one is due to the direct link of the information flow between particles and *gbest*, which then "guides" the rest of the swarm and thus resulting in the creation of similar particles with some loss of diversity. Hence this phenomenon increases the probability of being trapped in local optima [44] and it is the main cause of the premature convergence problem especially when the search space is of high dimensions [50] and the problem to be optimized is multi-modal [44]. Another reason for the 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 [51]. Various modifications and PSO variants have been proposed in order to address this problem such as [1], [7], [8], [9], [10], [13], [23], [25], [26], [29], [31], [32], [33], [34], [39], [40], [41], [42], [44], [46], [47], [51], [52], [54], [55], [56] and [58].

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 version 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. On one hand, most of them require additional parameters to accomplish this and thus making the PSO variants even more parameter dependent. On the other hand, the main problem is in fact the incapability of using the available diversity on one or more (dimensional) components of each particle (i.e. certain dimensions of a particle position in the search space), because all components continuously and abruptly change as the PSO process updates the particle's position. Note that one or more components of any particle might already be diverted well enough to be in a close vicinity of the global optimum (for that dimension). This potential is then wasted with the (velocity) update in the next iteration, which changes all the dimensions at once. Therefore, in the proposed method, we collect all such promising (or simply the best) components from each particle and fractionally create an artificial global best (GB) candidate, the *aGB*, which will be the swarm's GB particle if it is better than the previous GB and current *gbest*. Note that whenever a better (real) *gbest* particle or *aGB* particle emerges, it will replace the current GB particle. Without using any of the aforementioned modifications, we shall, therefore, show that the proposed fractional PSO can avoid the local optima and thus yield the optimum (or near optimum) solution even in high dimensional search spaces and usually in earlier stages.

Another major drawback of the aforementioned PSO variants including the basic method is that they can only be applied to a search space with a fixed dimension. However, in many optimization problems, the optimum dimension is also unknown (e.g. data clustering, spatial segmentation, optimization of the dimensional functions, etc.) and should thus be determined within the PSO process. Only few studies have been so far presented in this area, i.e. [1] and [36]. In [36], Omran et al. presented Dynamic Clustering PSO (DCPSO), which is in fact a hybrid clustering algorithm where binary PSO is used (only) to determine the number of clusters (and hence the dimension of the solution space) along with the selection of initial cluster centers whilst the traditional *K-means* method [48]

performs the clustering operation in that dimension (over the initial cluster centers). In [1], Abraham et al. presented the Multi-Elitist PSO (MEPSO), another variant of the basic PSO algorithm to address the premature convergence problem. In addition to being non-generic PSO variants that are applicable only to clustering problems, both [1] and [36] do not clarify whether or not they can cope with higher dimensions of the solution space since the maximum number of clusters used in their experiments are only 6 and 10, respectively. This is also true for most of the static (fixed dimensional) PSO variants due to the aforementioned fact that the probability of getting trapped into a local optimum significantly increases in higher dimensions [50].

In order to address these problems, we propose a multi-dimensional PSO (MD PSO) technique, which can work along with the fractional GB formation scheme (FGBF) to avoid the premature convergence problem. The proposed methods are generic since both FGBF scheme and the MD search process can dynamically be integrated into the PSO's native algorithm. Yet they are also not linked to each other, i.e. one can be performed without the other but we shall show if the MD search is required by the problem, the best performance is then achieved by their mutual operation. Furthermore, no additional parameter is needed to perform the proposed techniques. Furthermore, MD PSO voids the need of fixing the dimension of the solution space in advance.

The rest of the paper is organized as follows. Section II surveys related work on PSO. The proposed techniques, MD PSO and FGBF are presented in detail in Section III. Section IV is dedicated to applications over two problem domains, nonlinear function minimization and data clustering; whereas Section V provides the experiments conducted and discusses the results. Finally, Section VI concludes the paper.

II. Related Work

A. The Basic PSO algorithm

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

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

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

$y_{a,j}(t)$: 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 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 minimum of *f* in *N* dimensional space. Then the personal best of particle *a* can be updated in iteration *t*+1 as,

$$y_{a,j}(t+1) = \begin{cases} y_{a,j}(t) & \text{if } f(x_a(t+1)) > f(y_a(t)) \\ x_{a,j}(t+1) & \text{else} \end{cases} \forall j \in [1,N] \quad (1)$$

Since *gbest* is the index of the GB particle, then $\hat{y}(t) = y_{gbest}(t) = \min(y_1(t),..,y_S(t))$. Then for each iteration in a PSO process, positional updates are performed for each particle, $a \in [1,S]$ and along each dimensional component, $j \in [1,N]$, as follows:

$$v_{a,j}(t+1) = w(t)\,v_{a,j}(t) + c_1 r_{1,j}(t)\left(y_{a,j}(t) - x_{a,j}(t)\right) + c_2 r_{2,j}(t)\left(\hat{y}_j(t) - x_{a,j}(t)\right)$$
$$x_{a,j}(t+1) = x_{a,j}(t) + v_{a,j}(t+1) \quad (2)$$

where w is the inertia weight, [46] 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 a uniform distribution. Recall from the earlier discussion that the first term in the summation is the *memory* term, which represents the contribution of previous 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 and Eberhart [46], 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 the iterations of a PSO run, which updates the positions of the particles using (2). 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. $f < \varepsilon_C$ where f is the fitness function and ε_C is the cut-off error). Accordingly, the general pseudo-code of the *bPSO* is presented in Table I.

Table I: Pseudo-code for *bPSO* algorithm

bPSO (*termination criteria:{IterNo,* ε_C *,...},* $V_{\max}$)

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 ($f(y_a(t)) < \min\left(f(\hat{y}(t-1), f(y_i(t))\right)$) then *gbest* = *a* and $\hat{y}(t) = y_a(t)$
   3.2. End For.
   3.3. If any *termination criterion* is met, then **Stop**.
   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 Eq. (2)
         3.4.1.2. If($\left|v_{a,j}(t+1)\right| > V_{\max}$) then clamp it to $\left|v_{a,j}(t+1)\right| = V_{\max}$
         3.4.1.3. Compute $x_{a,j}(t+1)$ using Eq. (2)
      3.4.2. End For.
   3.5. End For.
4. End For.

Velocity clamping also called as "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 [13]. Some important PSO variants and improvements will be covered in the next section.

B. PSO Variants and Improvements

The first set of improvements have been proposed for enhancing the problem-dependent performance of PSO due to the strong parameter dependency. There are mainly two types of approaches; the first one is through self-adaptation, which has been applied to PSO by Clerc in [10], Yasuda et. al. in [57], Zhang et. al. in [60] and Shi and Eberhart in [47]. The other approach is via performing hybrid techniques, which are employed along with PSO by Angeline in [2], Reynolds et. al. in [43], Higashi and Iba in [23], Esquivel and Coello Coello in [15] and many others. The rest of the PSO variants presented in this section contain some improvements trying to avoid the premature convergence problem via introducing diversity to swarm particles.

Note that according to the velocity update equation in (2), the velocity of the *gbest* particle will only depend on the memory term since $x_{gbest} = y_{gbest} = \hat{y}$. To address this problem Van den Bergh introduced a new PSO variant, the PSO with guaranteed convergence, (GCPSO) [50]. In GCPSO, a different velocity update equation is used for the *gbest* particle based on two threshold values that can be adaptively set during the process. It is claimed that GCPSO usually performs better than the *bPSO* when applied to uni-modal functions and comparable for multi-modal problems; however, due to its fast rate of convergence, GCPSO may be more likely to trap in local minima with a guaranteed convergence whereas the *bPSO* may not. Based on GCPSO, Van den Bergh proposed the Multi-start PSO (MPSO) [50], which repeatedly runs GCPSO over randomized particles and stores the (local) optimum at each iteration. Yet, similar to *bPSO* and many of its variants, the performance still degrades significantly as the dimension of the search space increases [50].

In [52], a cooperative approach to PSO (CPSO) has been proposed. This is another variation of the *bPSO*, which employs cooperative behavior to improve the performance. In this approach, multiple swarms are used to optimize different components of the solution vector in a cooperative way. 80% of all test cases which are run on 30 dimensional space, CPSO performed better than *bPSO*. Comparable results are obtained from a recent PSO variant, the comprehensive learning PSO, CLPSO, proposed in [31]. CLPSO basically follows a comprehensive learning strategy, where all swarm particles' historical best information is used to update a particle's velocity. The authors concluded that CLPSO is not the best choice for solving uni-modal functions; however it can generate better quality solutions more frequently when compared with eight other PSO variants. A similar approach has also been presented by Mendes et. al in [34], which proposed the fully informed particle swarm, the so-called FIPS. In their work, using the particles' previous best values to update the velocity of the particle is the main approach and several neighbourhood topologies, such as pyramid, square, ring, circle, etc. were examined. There are many other PSO variants, which can be found in [14] and [37]. Yet most of them present either little or moderate performance improvements at the expense of additional parameters and/or computational complexity. More importantly, they still suffer from the high dimensions and multi-modality of the problem where it becomes easier to trap into local optima especially during the earlier stages of a PSO process. In order to provide an unbiased performance measure, we shall not use any of such improvements with the proposed techniques, multi-dimensional (MD) PSO and fractional GB formation (FGBF) as detailed next.

III. THE MD PSO AND FGBF TECHNIQUES

In this section, we introduce two novel techniques for PSO. The first is a multidimensional extension, the so-called MD PSO, which presents a substantial improvement over PSO via inter-dimensional navigation. However, it usually suffers in high dimensions from premature convergence to a local optimum, similar to other PSO variants.

To remedy this shortcoming, we will then propose a second technique, called FGBF and present their mutual application over two typical problems: nonlinear function minimization and data clustering.

A. MD PSO Algorithm

Instead of operating at a fixed dimension *N*, the MD PSO algorithm is designed to seek both positional and dimensional optima within a dimension range, ($D_{\min} \le N \le D_{\max}$). In order to accomplish this, each particle has two sets of components, each of which has been subjected to two independent and consecutive processes. The first one is a regular positional PSO, i.e. the traditional velocity updates and following positional moves in *N* dimensional search (solution) space. The second one is a 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 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 the optimum dimension).

In a MD PSO process and at time (iteration) *t*, each particle *a* in the swarm, $\xi = \{x_1,..,x_a,..,x_S\}$, is represented by the following characteristics:

$xx_{a,j}^{xd_a(t)}(t)$: j^{th} component (dimension) of the position of particle *a*, in dimension $xd_a(t)$

$vx_{a,j}^{xd_a(t)}(t)$: j^{th} component (dimension) of the velocity of particle *a*, in dimension $xd_a(t)$

$xy_{a,j}^{xd_a(t)}(t)$: 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*

$\hat{xy}_j^d(t)$: j^{th} component (dimension) of the global best position of swarm, in dimension *d*

$xd_a(t)$**:** Dimension component of particle *a*

$vd_a(t)$**:** Velocity component of dimension of particle *a*

$\tilde{xd}_a(t)$**:** Personal best dimension component of particle *a*

Figure 1 shows sample MD PSO and *bPSO* particles with index *a*. The *bPSO* 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 given between 2 and 9; therefore the particle contains 8 sets of positional components (one for each dimension). In this example, the current dimension where the particle *a* resides is 2 ($xd_a(t) = 2$) whereas its personal best dimension is 3 ($\tilde{xd}_a(t) = 3$). Therefore, at time *t,* a positional PSO update is first performed over the positional elements, $xx_a^2(t)$ and then the particle may move to another dimension by the dimensional PSO.

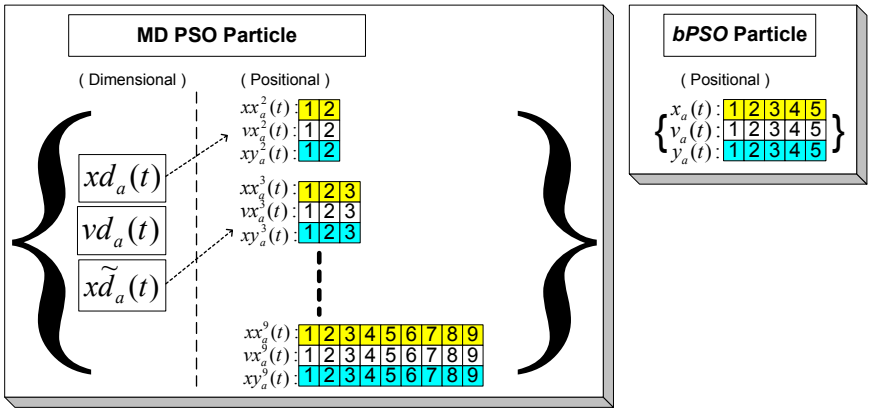

Figure 1: Sample MD PSO (right) vs. *bPSO* (left) particle structures. For MD PSO $[D_{\min}=2,\ D_{\max}=9]$ and at the current time *t*, $xd_a(t)=2$ and $\widetilde{xd}_a(t)=3$. For *bPSO N*=5.

Let f denote the dimensional fitness function that is to be optimized within a certain dimension range, ($D_{\min} \le N \le D_{\max}$). Without loss of generality assume that the objective is to find the minimum (position) of f at 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,

$$xy_{a,j}^{xd_a(t+T)}(t+T)=\begin{Bmatrix} xy_{a,j}^{xd_a(t)}(t) & if\ \ f(xx_a^{xd_a(t+T)}(t+T))>f(xy_a^{xd_a(t)}(t)) \\ xx_{a,j}^{xd_a(t+T)}(t+T) & else \end{Bmatrix} \forall j \in [1, xd_a(t)] \quad (3)$$

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

$$\widetilde{xd}_a(t+1)=\begin{Bmatrix} \widetilde{xd}_a(t) & if\ \ f(xx_a^{xd_a(t+1)}(t+1))>f(xy_a^{\widetilde{xd}_a(t)}(t)) \\ xd_a(t+1) & else \end{Bmatrix} \quad (4)$$

Recall that *gbest(d)* is the index of the global best particle at dimension d then $\hat{xy}^{dbest}(t)=xy_{gbest(dbest)}^{dbest}(t)=\min(xy_1^{dbest}(t),..,xy_S^{dbest}(t))$. For a particular iteration t, and for a particle $a \in [1,S]$, first the positional components are updated in the current dimension, $xd_a(t)$, and then the dimensional update is performed to determine the next (t+1st) dimension, $xd_a(t+1)$. The positional update is performed for each dimension component, $j \in [1, xd_a(t)]$ as follows:

$$\begin{gathered} vx_{a,j}^{xd_a(t)}(t+1)=w(t)vx_{a,j}^{xd_a(t)}(t)+c_1r_{1,j}(t)\left(xy_{a,j}^{xd_a(t)}(t)-xx_{a,j}^{xd_a(t)}(t)\right)+c_2r_{2,j}(t)\left(\hat{xy}_j^{xd_a(t)}(t)-xx_{a,j}^{xd_a(t)}(t)\right) \\ xx_{a,j}^{xd_a(t)}(t+1)=xx_{a,j}^{xd_a(t)}(t)+vx_{a,j}^{xd_a(t)}(t+1) \end{gathered} \quad (5)$$

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 fly to another dimension afterwards with the following dimensional update equations:

$$\begin{gathered} vd_a(t+1)=\left\lfloor vd_a(t)+c_1r_1(t)\left(\widetilde{xd}_a(t)-xd_a(t)\right)+c_2r_2(t)\left(dbest-xd_a(t)\right)\right\rfloor \\ xd_a(t+1)=xd_a(t)+vd_a(t+1) \end{gathered} \quad (6)$$

where $\lfloor . \rfloor$ is the *floor* operator. Unlike in Eq. (2), an inertia weight is not used for positional velocity update, since no benefit was obtained experimentally for dimensional PSO. To avoid exploding, along with the positional velocity limit $V_{\max}$, two more clamping operations are applied for dimensional PSO components, such as $\left|vd_{a,j}(t+1)\right| < VD_{\max}$ and the initial dimension range set by the user, $D_{\min} \le xd_a(t) \le D_{\max}$. Accordingly, the

general pseudo-code of the MD PSO technique is given in Table II.

Table II: Pseudo-code of MD PSO algorithm

MD PSO (*termination criteria:{IterNo,* ε_C *, ...},* $V_{\max}$, $VD_{\max}$, $D_{\min}$, $D_{\max}$)

1. For $\forall a \in [1, S]$ do:
   - 1.1. Randomize $xd_a(0), vd_a(0)$
   - 1.2. Initialize $\tilde{xd}_a(0) = xd_a(0)$
   - 1.3. For $\forall d \in [D_{\min}, D_{\max}]$ do:
     - 1.3.1. Randomize $xx_a^d(0)$, $xv_a^d(0)$
     - 1.3.2. Initialize $xy_a^d(0) = xx_a^d(0)$
   - 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 ($f(xx_a^{xd_a(t)}(t)) < \min\left(f(xy_a^{xd_a(t)}(t-1), \min_{p \in S-\{a\}} \left(f(xx_p^{xd_a(t)}(t)) \right) \right)$) then do:
       - *3.1.1.1.* $xy_a^{xd_a(t)}(t) = xx_a^{xd_a(t)}(t)$
       - *3.1.1.2.* If ($f(xx_a^{xd_a(t)}(t)) < f(xy_{gbest(xd_a(t))}^{xd_a(t)}(t-1))$) then $gbest\,(xd_a(t))=a$
       - *3.1.1.3.* If ($f(xx_a^{xd_a(t)}(t)) < f(xy_a^{\tilde{xd}_a(t-1)}(t-1))$) then $\tilde{xd}_a(t) = xd_a(t)$
       - *3.1.1.4.* If ($f(xx_a^{xd_a(t)}(t)) < f(\hat{xy}^{dbest}(t-1))$) then $dbest = xd_a(t)$
     - 3.1.2. End If.
   - 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 $vx_{a,j}^{xd_a(t)}(t+1)$ using Eq. (5)
       - *3.4.1.2.* if($\left|vx_{a,j}^{xd_a(t)}(t+1)\right| > V_{\max}$) then clamp it to $\left|vx_{a,j}^{xd_a(t)}(t+1)\right| = V_{\max}$
       - *3.4.1.3.* Compute $xx_{a,j}^{xd_a(t)}(t+1)$ using Eq. (5)
     - 3.4.2. End For.
     - 3.4.3. Compute $vd_a(t+1)$ using Eq. (6)
     - 3.4.4. if($\left|vd_a(t+1)\right| > VD_{\max}$) then clamp it to $\left|vd_a(t+1)\right| = VD_{\max}$
     - 3.4.5. Compute $xd_a(t+1)$ using Eq. (6)
     - 3.4.6. if($xd_a(t+1) < D_{\min}$) then $xd_a(t+1) = D_{\min}$)
     - 3.4.7. if($xd_a(t+1) > D_{\max}$) then $xd_a(t+1) = D_{\max}$)
   - 3.5. End For.
4. End For.

Once the MD PSO process terminates, the optimum solution will be $\hat{xy}^{dbest}$ at the optimum dimension, *dbest*, achieved by the particle *gbest(dbest)* and finally the best (fitness) score achieved will naturally be $f(\hat{xy}^{dbest})$.

B. FGBF Algorithm

Fractional GB formation (FGBF) is designed to avoid 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 *bPSO* process, an artificial GB particle (*aGB*) is

(fractionally) formed by selecting the most promising (or simply the best) particle (dimensional) components from the entire swarm. Therefore, especially during the initial steps, the FGBF can most of the time be 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 most promising (or simply the best) components among them. This process naturally uses the available diversity among individual dimensional components and thus it can prevent the swarm from trapping in local optima. Suppose for a swarm ξ, FGBF is performed in a PSO process at a (fixed) 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 ($y_{a,j}(t)$), $j \in [1, N]$). *aGB* particle, first of all, does not use a velocity term, since instead of velocity updates, the *aGB* particle is fractionally (re-) created from the dimensions of some swarm particles. Consequently, $y_{aGB}(t)$ is set to the best of $x_{aGB}(t)$ and $y_{aGB}(t-1)$. As a result, the FGBF process creates one *aGB* particle providing a (potential) GB solution ($y_{aGB}(t)$). Let $f(a, j)$ be the dimensional fitness score of the j^{th} component of particle *a*. Suppose that all dimensional fitness scores ($f(a, j), \forall a \in [1, S]$) can be computed in step 3.1 and FGBF pseudo-code as given in Table III can then be plugged in between steps 3.3 and 3.4 of *bPSO*'s pseudo-code.

Table III: Pseudo-code of FGBF in *bPSO*

FGBF in *bPSO* (ξ, $f(a, j)$)

1. Create a new *aGB* particle, $\{x_{aGB,j}(t),\ y_{aGB,j}(t)\}\ for\ \forall j \in [1, N]$
2. Let $a[j] = \arg \min_{a \in \xi\ j \in [1,N]} (f(a, j))$ be the index array of particles yielding the minimum $f(a, j)$ for the j^{th} dimensional component.
3. $x_{aGB,j}(t) = x_{a[j],j}(t)\ for\ \forall j \in [1, N]$
4. If ($f(x_{aGB}(t)) < f(y_{aGB}(t-1))$) then $y_{aGB}(t) = x_{aGB}(t)$
5. Else $y_{aGB}(t) = y_{aGB}(t-1)$
6. If ($f(y_{aGB}(t)) < f(\tilde{y}(t))$ then $\tilde{y}(t) = y_{aGB}(t)$

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). For those problems without any constraints (e.g. nonlinear function minimization), the best dimensional components can simply be selected whereas in others (e.g. clustering), some promising components, which satisfy the constraints, are first selected, grouped and the most suitable one in each group is then used for FGBF. Here, the internal nature of the problem will determine the "suitability" of the selection. Take for instance the function minimization problem as illustrated in Figure 2 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 ranked 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 *gbest*.

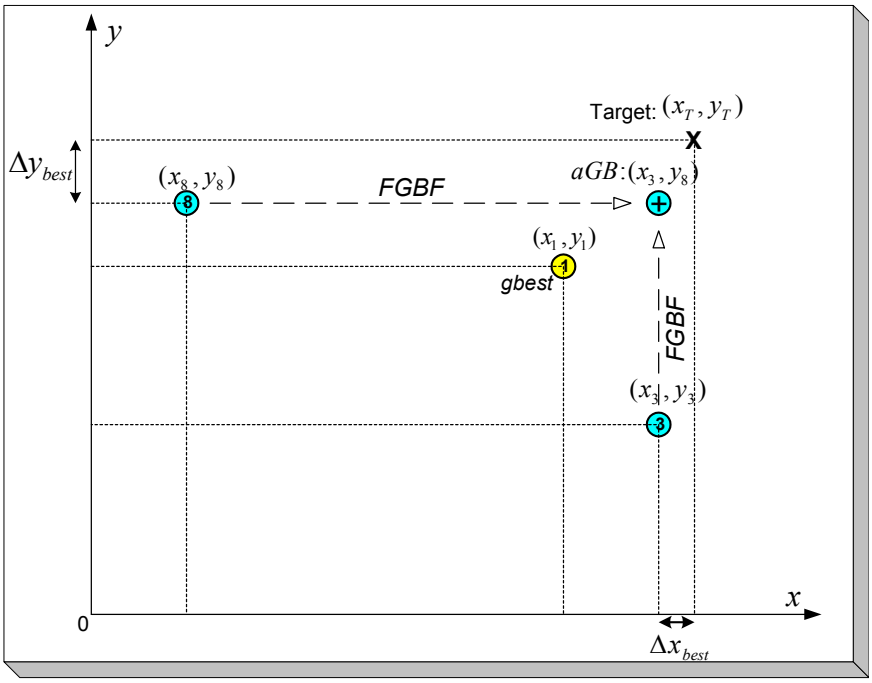

Figure 2: A sample FGBF in 2D space.

C. *FGBF Algorithm over MD PSO*

The previous section introduced the principles of FGBF when applied in a *bPSO* process on a single dimension. In this section we present its generalized form with the proposed MD PSO where there is one *gbest* particle per (potential) dimension of the solution space. For this purpose, recall from the earlier discussion that in a particular iteration, *t*, each MD PSO particle, *a*, has the following components: position ($xx_{a,j}^{xd_a(t)}(t)$), velocity ($vx_{a,j}^{xd_a(t)}(t)$) and the personal best position ($xy_{a,j}^{xd_a(t)}(t)$) for each potential dimensions in solution space (i.e. $xd_a(t) \in [D_{\min}, D_{\max}]$ and $j \in [1, xd_a(t)]$) and their respective counterparts in the dimensional PSO process (i.e. $xd_a(t)$, $vd_a(t)$ and $x\tilde{d}_a(t)$). *aGB* particle does not need dimensional components where a single positional component with the maximum dimension $D_{\max}$ is created to cover all dimensions in the range, $\forall d \in [D_{\min}, D_{\max}]$, and as explained earlier, there is no need for the velocity term either, since *aGB* particle is fractionally (re-) created from the dimensions of some swarm particles. Furthermore, the aforementioned competitive selection ensures that $xy_{aGB}^{d}(t), \forall d \in [D_{\min}, D_{\max}]$ is set to the best of the $xx_{aGB}^{d}(t)$ and $xy_{aGB}^{d}(t-1)$. As a result, the FGBF process creates one *aGB* particle providing (potential) GB solutions ($xy_{aGB}^{d}(t)$) for all dimensions in the given range (i.e. $\forall d \in [D_{\min}, D_{\max}]$). Let $f(a, j)$ be the dimensional fitness score of the j^{th} component of particle *a*, which has the current dimension, $xd_a(t)$ and $j \in [1, xd_a(t)]$. At a particular time *t*, all dimensional fitness scores ($f(a, j), \forall a \in [1, S]$) can be computed in step 3.1 and FGBF pseudo-code for MD PSO as given in Table IV can then be plugged in between steps 3.2 and 3.3 of the MD PSO's pseudo-code. We will next present the applications of both techniques on two well-known problem domains.

Table IV: Pseudo-code for FGBF in MD PSO

FGBF in MD PSO ($f(a, j)$)

1. Create a new *aGB* particle, $\{xx^d_{aGB,j}(t),\ xy^d_{aGB,j}(t)\}\ for\ \forall d \in [D_{\min}, D_{\max}], \forall j \in [1, d]$
2. Let $a[j] = \arg \min_{a \in [1,S]\, j \in [1, D_{\max}]} (f(a, j))$ be the index array of particles yielding the minimum $f(a, j)$ for the j^{th} dimension.
3. For $\forall d \in [D_{\min}, D_{\max}]$ do:
   - 3.1. $xx^d_{aGB,j}(t) = xx^d_{a[j],j}(t)\ for\ \forall j \in [1, d]$
   - 3.2. If ($f(xx^d_{aGB}(t)) < f(xy^d_{aGB}(t-1))$) then $xy^d_{aGB}(t) = xx^d_{aGB}(t)$
   - 3.3. Else $xy^d_{aGB}(t) = xy^d_{aGB}(t-1)$
   - 3.4. If ($f(xy^d_{aGB}(t)) < f(xy^d_{gbest(d)}(t))$ then $xy^d_{gbest(d)}(t) = xy^d_{aGB}(t)$
4. End For.
5. Re-assign *dbest*: $dbest = \arg \min_{d \in [D_{\min}, D_{\max}]} (f(xy^d_{gbest(d)}(t))$

IV. Applications

Two problem domains are considered in this paper where the proposed PSO algorithms are applied. The first one is nonlinear function minimization where several benchmark functions are used. This allows us to test the performance over multi-dimensional search spaces and against both uni- and multi-modalities. The second domain is data clustering, which provides certain constraints in multi-dimensional solution space and allows the performance evaluation in the presence of significant variation in data distribution with an impure validity index. Both problem domains can efficiently validate MD PSO's performance regarding convergence to the global solution in the right dimension with or without FGBF. In this way, we can truly evaluate the contribution and the significance of FGBF especially over multi-modal optimization problems in high dimensions.

A. Nonlinear Function Minimization

We selected seven benchmark functions and biased them with a dimensional term in order to test the performance of MD PSO. The functions given in Table V provide a good mixture of complexity and modality and have been widely studied by several researchers, e.g. see [3], [15], [23], [33], [45] and [46]. The dimensional bias term, $\Psi(d)$, has the form of $\Psi(d) = K|d - d_0|^{\alpha}$ where the constants *K* and α are properly set with respect to the dynamic range of the function to be minimized. Note that the variable $D_{\min} \le d_0 \le D_{\max}$ is the target dimension in which the global minimum can be truly reached and all functions thus have the global minimum $F_n(x, d_0) = 0$, when $d = d_0$. *Sphere, De Jong* and *Rosenbrock* are the uni-modal functions and the rest are multi-modal, meaning that they have many deceiving local minima. On the macroscopic level *Griewank* demonstrates certain similarities with uni-modal functions especially when the dimensionality is above 20; however, in low dimensions it bears a significant noise, which creates many local minima due to the second multiplication term with *cosine* components. Yet with the addition of dimensional bias term $\Psi(d)$, even uni-modal functions eventually become multi-modal since they now have a local minimum at every dimension (which is their global minimum at that dimension without $\Psi(d)$) but only one global minimum at dimension d_0.

Table V: Benchmark functions with dimensional bias

Function	Formula	Initial Range $\pm x_{max}$	Dimension Range $[D_{min}, D_{max}]$
Sphere	$F_1(x,d_0)=\left(\sum_{i=1}^{d} x_i^2\right)+(d-d_0)^4$	±150	[2, 100]
De Jong	$F_2(x,d_0)=\left(\sum_{i=1}^{d} ix_i^4\right)+(d-d_0)^4$	±50	[2, 100]
Rosenbrock	$F_3(x,d)=\left(\sum_{i=1}^{d} 100(x_{i+1}-x_i^2)^2+(x_i-1)^2\right)+(d-d_0)^4$	±50	[2, 100]
Rastrigin	$F_4(x,d_0)=\left(\sum_{i=1}^{d} 10+x_i^2-10Cos(2\pi x_i)\right)+(d-d_0)^4$	±50	[2, 100]
Griewank	$F_5(x,d_0)=\left(\frac{1}{4000}\sum_{i=1}^{d} x_i^2-\prod_{i=1}^{d} Cos\left(\frac{x_i}{\sqrt{i+1}}\right)\right)+0.2(d-d_0)^2$	±500	[2, 100]
Schwefel	$F_6(x,d_0)=\left(418.9829\,d+\sum_{i=1}^{d} x_i\,Sin\left(\sqrt{\lvert x_i\rvert}\right)\right)+40(d-d_0)^2$	±500	[2, 100]
Giunta	$F_7(x,d_0)=\left(\sum_{i=1}^{d} Sin\left(\frac{16}{15}x_i-1\right)+Sin^2\left(\frac{16}{15}x_i-1\right)+\frac{1}{50}Sin\left(4\left(\frac{16}{15}x_i-1\right)\right)+\frac{268}{1000}\right)+\sqrt{\lvert d-d_0\rvert}$	±500	[2, 100]

Recall from the earlier remarks that a MD PSO particle *a* represents a potential solution at a certain dimension and therefore, the j^{th} component of a *d*-dimensional point ($x_j, j\in[1,d]$) is stored in its positional component, $xx_{a,j}^{d}(t)$ at time *t*. Step 3.1 in MD PSO's pseudo-code computes the (dimensional) fitness score ($f(a,j)$) of the j^{th} component (x_j) and at step 2 in FGBF process, the index of the particle with those x_j's yielding minimum $f(a,j)$ is then stored in the array $a[j]$. Except the non-seperable functions, *Rosenbrock* and *Griewank*, the assignment of $f(a,j)$ for particle *a* is straightforward (e.g. $f(a,j)=x_j^2$ for *Sphere*, $f(a,j)=x_j Sin\left(\sqrt{\lvert x_j\rvert}\right)$ for *Schwefel*, etc., simply using the term with j^{th} component of the summation). For *Rosenbrock* we can set $f(a,j)=(x_{j+1}-x_j^2)^2+(x_j-1)^2$ since the *aGB* particle, which is fractionally formed by those x_j's minimizing the j^{th} summation term, eventually minimizes the function. Finally for *Griewank* one can approximate $f(a,j)\approx x_j^2$ for particle *a* and the FGBF operation then finds and uses such x_j that can come to a close vicinity of the global minimum at dimension *j* on a macroscopic scale, so that the native PSO process can then have a higher chance of avoiding those noise-like local optima, and thus eventually converge to the global optimum.

B. Data Clustering

1) Problem Definition

As the process of identifying natural groupings in a multidimensional data based on some distance metric (e.g. *Euclidean*), data clustering can be divided into two main categories: hierarchical and partitional [17]. Each

category then has a wealth of sub-categories and different algorithmic approaches for finding the clusters. Clustering can also be performed in two different modes: hard (or crisp) and fuzzy. In the former mode, the clusters are disjoint, non-overlapping and any data point belongs to a single cluster whereas in the latter case it can belong to all the clusters with some degree of membership [24]. *K-means* [48] is a well known and widely used clustering method, which first assigns each data point to one of the *K* cluster *centroids* and then updates them to the *mean* of their associated points. Starting from a random set of *K* centroids, this cycle is then iteratively performed until the convergence criteria, $\Delta_{Kmeans} < \varepsilon$ is reached where the objective function, Δ_{Kmeans} can be expressed as,

$$\Delta_{Kmeans} = \sum_{k=1}^{K} \sum_{x_p \in c_k} \left\| c_k - x_p \right\|^2 \tag{7}$$

where c_k is the k^{th} cluster center, x_p is the p^{th} data point in cluster c_k and $\|.\|$ is the distance metric in Euclidean space. As a hard clustering method, *K-means* suffers from the following drawbacks:

- The number of clusters *K*, needs to be set in advance.
- The performance of the method depends on the initial (random) centroid positions as the method converges to the closest local optima.
- The method is also dependent on the data distribution.

The fuzzy version of *K-means*, the so-called Fuzzy C-means (FCM) (yet sometimes also called as Fuzzy K-means) was proposed by Bezdek in [6] and has become the most popular fuzzy clustering method so far. It is a fuzzy extension of K-means whilst FCM usually achieves a better performance than *K-means* [19] and is less data dependent; however, it still suffers from the same drawbacks, i.e. the number of clusters should be fixed *a priori* and unfortunately it may also converge to local optima [24]. Zhang and Hsu in [59] proposed a novel fuzzy clustering technique, the so-called K harmonic means (KHM), which is less sensitive to initial conditions and promises further improvements. Experimental results demonstrate that KHM outperforms both *K-means* and FCM [20], [59]. There are many other variants that are skipped here since clustering is only an application field for the proposed PSO techniques and hence out of the main scope of this paper. An extensive survey over various types of clustering techniques can be found in [24] and [38].

A hard clustering technique based on the basic PSO (*bPSO*) was first introduced by Omran et al. in [35] and this work showed that the *bPSO* can outperform *K-means*, FCM, KHM and some other state-of-the-art clustering methods in any (evaluation) criteria. This is indeed an expected outcome due to the PSO's aforementioned ability to cope up with the local optima by maintaining a guided random search operation through the swarm particles. In clustering, similar to other PSO applications, each particle represents a potential solution at a particular time *t*, i.e. the particle *a* in the swarm, $\xi = \{x_1,..,x_a,..,x_S\}$, is formed as $x_a(t) = \{c_{a,1},..,c_{a,j},..,c_{a,K}\} \Rightarrow x_{a,j}(t) = c_{a,j}$ where $c_{a,j}$ is the j^{th} (potential) cluster centroid in *N* dimensional data space and *K* is the number of clusters fixed in advance. Note that contrary to nonlinear function minimization in the earlier section, the data space dimension, *N*, is now different than the solution space dimension, *K*. Furthermore, the fitness function, *f* that is to be optimized, is formed with respect to two widely used criteria in clustering:

- *Compactness*: Data items in one cluster should be similar or close to each other in *N* dimensional space and different or far away from the others when belonging to different clusters.
- *Separation*: Clusters and their respective centroids should be distinct and well-separated from each

other.

The fitness functions for clustering are then formed as a regularization function fusing both *Compactness* and *Separation* criteria and in this problem domain they are known as clustering validity indices. Omran et al. used the following validity index in their work [35],

$$f(x_a, Z) = w_1 \bar{d}_{\max}(x_a, Z) + w_2(Z_{\max} - d_{\min}(x_a)) + w_3 Q_e(x_a)$$

$$where \quad Q_e(x_a) = \frac{1}{K}\sum_{j=1}^{K} \frac{\sum_{\forall z_p \in x_{a,j}} \left\| x_{a,j} - z_p \right\|}{\left\| x_{a,j} \right\|} \tag{8}$$

where Q_e is the quantization error (or the average intra-cluster distance), $\bar{d}_{\max}$ is the maximum average *Euclidean* distance of data points, $Z = \{z_p, z_p \in x_{a,j}\}$, to their centroids, x_a. $Z_{\max}$ is a constant value for theoretical maximum inter-cluster distance, and $d_{\min}$ is the minimum centroid (inter-cluster) distance in the cluster centroid set x_a. The weights, w_1, w_2, w_3 are user defined regularization coefficients. So the minimization of the validity index $f(x_a, Z)$ will simultaneously try to minimize the intra-cluster distances (for better *Compactness*) and maximize the inter-cluster distance (for better *Separation*). In such a regularization approach, different priorities (weights) can be assigned to both sub-objectives via proper setting of weight coefficients. Another traditional and well-known validity index is Dunn's index [12], which suffers from two drawbacks: It is computationally expensive and sensitive to noise [22]. Several variants of Dunn's index were proposed in [38] where robustness against noise is improved. There are many other validity indices, i.e. proposed by Turi [49], Davies and Bouldin [11], Halkidi and Vazirganis [21], etc. A throughout survey can be found in [22]. Most of them presented promising results; however, none of them can guarantee the "optimum" number of clusters in every clustering scheme. Especially for the aforementioned PSO-based clustering in [35], the clustering scheme further depends on weight coefficients and may, therefore, result in over- or under-clustering particularly in complex data distributions.

Although PSO-based clustering outperforms many well-known clustering methods, it still suffers from two major drawbacks. The number of clusters, *K*, (being the solution space dimension as well) should still be specified in advance and similar to other *bPSO* applications, the method tends to trap in local optima particularly when the complexity of the clustering scheme increases. This also involves the dimension of the solution space, i.e. convergence to "optimum" number of "true" clusters can only be guaranteed for low dimensions. Recall from the earlier discussion that this is also true for dynamic clustering schemes, DCPSO [36] and MEPSO [1], both of which eventually present results only in low dimensions ($K \leq 10$ in [36] and $K \leq 6$ in [1]) and for simple data distributions. The degradation is likely to be more severe particularly for DCPSO since it entirely relies on *K-means* for actual clustering.

2) Clustering based on MD PSO with FGBF

Based on the earlier discussion it is obvious that the clustering problem requires the determination of the solution space dimension (i.e. number of clusters, *K*) and an effective mechanism to avoid local optima traps (both dimensionally and spatially) particularly in complex clustering schemes in high dimensions (e.g. $K > 10$). The former requirement justifies the use of the proposed MD PSO technique while the latter calls for FGBF. At time *t*, the particle *a* in the swarm, $\xi = \{x_1, .., x_a, .., x_S\}$, has the positional component formed as,

$xx_a^{xd_a(t)}(t) = \{c_{a,1},..,c_{a,j},..,c_{a,xd_a(t)}\} \Rightarrow xx_{a,j}^{xd_a(t)}(t) = c_{a,j}$ meaning that it represents a potential solution (i.e. the cluster centroids) for the $xd_a(t)$ number of clusters whilst j^{th} component being the j^{th} cluster centroid. Apart from the regular limits such as (spatial) velocity, $V_{\max}$, dimensional velocity, $VD_{\max}$ and dimension range $D_{\min} \le xd_a(t) \le D_{\max}$, the N dimensional data space is also limited with some practical spatial range, i.e. $X_{\min} < xx_a^{xd_a(t)}(t) < X_{\max}$. In case this range is exceeded even for a single dimension j, $xx_{a,j}^{xd_a(t)}(t)$, then all positional components of the particle for the respective dimension $xd_a(t)$ are initialized randomly within the range (i.e. refer to step 1.3.1 in MD PSO pseudo code) and this further contributes to the overall diversity. The following validity index is used to obtain computational simplicity with minimal or no parameter dependency,

$$f(xx_a^{xd_a(t)}, Z) = Q_e(xx_a^{xd_a(t)})(xd_a(t))^{\alpha} \quad where$$
$$Q_e(xx_a^{xd_a(t)}) = \frac{1}{xd_a(t)} \sum_{j=1}^{xd_a(t)} \frac{\sum_{\forall z_p \in xx_{a,j}^{xd_a(t)}} \left\| xx_{a,j}^{xd_a(t)} - z_p \right\|}{\left\| xx_a^{xd_a(t)} \right\|} \tag{9}$$

where Q_e is the quantization error (or the average intra-cluster distance) as the *Compactness* term and $(xd_a(t))^{\alpha}$ is the *Separation* term, by simply penalizing higher cluster numbers with an exponential, $\alpha > 0$. Using $\alpha = 1$, the validity index yields the simplest form (i.e. only the nominator of Q_e) and becomes entirely parameter-free.

On the other hand, (hard) clustering has some constraints. Let $C_j = \{xx_{a,j}^{xd_a(t)}(t)\} = \{c_{a,j}\}$ be the set of data points assigned to a (potential) cluster centroid $xx_{a,j}^{xd_a(t)}(t)$ for a particle a at time t. The partitions $C_j,\ \forall j \in [1, xd_a(t)]$ should maintain the following constraints:

1. Each data point should be assigned to one cluster set, i.e. $\bigcup_{j=1}^{xd_a(t)} C_j = Z$
2. Each cluster should contain at least one data point, i.e. $C_j \neq \{\phi\},\ \forall j \in [1, xd_a(t)]$
3. Two clusters should have no common data points, i.e. $C_i \cap C_j = \{\phi\},\ i \neq j \ and \ \forall i, j \in [1, xd_a(t)]$

In order to satisfy the 1st and 3rd (hard) clustering constraints, before computing the clustering fitness score via the validity index function in (9), all data points are first assigned to the *closest* centroid. Yet there is no guarantee for the fulfillment of the 2nd constraint since $xx_a^{xd_a(t)}(t)$ is set (updated) by the internal dynamics of the MD PSO process and hence any dimensional component (i.e. a potential cluster candidate), $xx_{a,j}^{xd_a(t)}(t)$, can be in an abundant position (i.e. no closest data point exists). To avoid this, a high penalty is set for the fitness score of the particle, i.e. $f(xx_a^{xd_a(t)}, Z) \approx \infty,\ if\ \{xx_{a,j}^{xd_a(t)}\} = \{\phi\}$ for any j.

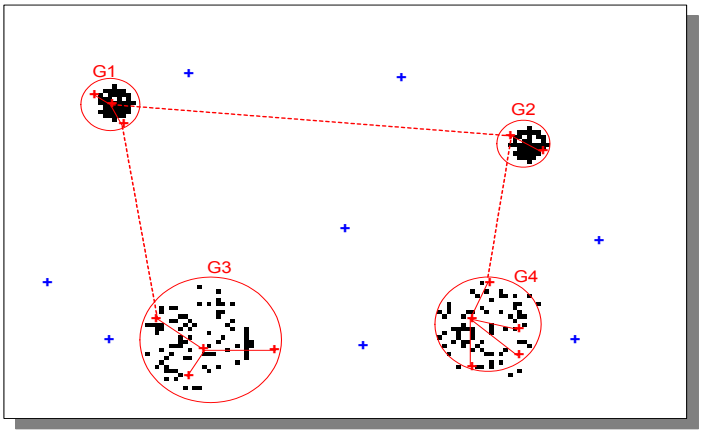

Figure 3: The formation of the centroid subset in a sample clustering example. The black dots represent data points over 2D space and each colored '+' represents one centroid (dimension) of a swarm particle.

The major outlines so far given are sufficient for the standalone application of the MD PSO technique for a dynamic clustering application; however, the FGBF operation presents further difficulties since for the *aGB* creation the selection of the best or the most promising dimensions (i.e. the cluster centroids) among all dimensions of swarm particles is not straightforward. Recall that in step 2 of the FGBF pseudo code, the index array of such particles yielding the minimum $f(a,j)$ for the j^{th} dimension, can be found as, $a[j] = \arg \min_{a\in[1,S]\, j\in[1,D_{\max}]} (f(a,j))$. This was straightforward for the nonlinear function minimization where each dimension of the solution space is distinct and corresponds to an individual dimension of the data space. However, in the clustering application, any (potential) cluster centroid of each particle, $xx_{a,j}^{xd_a(t)}(t)$, is updated independently and can be any arbitrary point in *N* dimensional data space. Furthermore, data points assigned to the j^{th} dimension of a particle *a,* ($\forall z_p \in xx_{a,j}^{xd_a(t)}(t)$), also depend on the distribution of the other dimensions (centroids), i.e. the "closest" data points are assigned to the j^{th} centroid only because the other centroids happen to be at a farther location. Inserting this particular dimension (centroid) into another particle (say *aGB*, in case selected), might create an entirely different assignment (or cluster) including the possibility of having no data points assigned to it and thus violating the 2nd clustering constraint. To avoid this problem, a new approach is adopted for step 2 to obtain $a[j]$. At each iteration, a subset among all dimensions of swarm particles is first formed by verifying the following: a dimension of any particle is selected into this subset if and only if there is at least one data point that is closest to it. Henceforth, the creation of the *aGB* particle within this verified subset ensures that the 2nd clustering constraint will (always) be satisfied. Figure 3 illustrates the formation of the subset on a sample data distribution with 4 clusters. Note that in the figure, all dimensions of the entire swarm particles are shown as '+' but the red ones belonging to the subset have at least one (or more) data points closest whereas the blue ones have none and hence they are discarded.

Once the subset centroids are selected, then the objective is to compose $a[j]$ with the most promising $D_{\max}$ centroids selected from the subset in such a way that each dimensional component of the *aGB* particle with *K* dimensions ($xx_{aGB,j}^{K}(t), \forall j \in [1,K]$), which is formed from $a[j]$ (see step 3.1 of FGBF in MD PSO pseudo code) can represent one of the true clusters, i.e., being in a close vicinity of its centroid. To accomplish this, only such

$D_{\max}$ dimensions that fulfill the two clustering criteria, *Compactness* and *Separation* are selected and then stored in $a[j]$. To achieve well-separated clusters and to avoid the selection of more than one centroid representing the same cluster, *spatially* close centroids are first grouped using a Minimum Spanning Tree (MST) [30] and then a certain number of centroid groups, say $d \in [D_{\min}, D_{\max}]$, can be obtained simply by breaking (d-1) longest MST branches. From each group, one centroid, which provides the highest *Compactness* score (i.e. minimum dimensional fitness score, $f(a, j)$) is then selected and inserted into $a[j]$ as the j^{th} dimensional component. During the computation of the validity index $f(xx_a^{xd_a(t)}, Z)$ in (9), $f(a, j)$ can simply be set as the j^{th} term of the summation in Q_e expression, such as,

$$f(a,j) = \frac{\sum_{\forall z_p \in xx_{a,j}^{xd_a(t)}} \left\| xx_{a,j}^{xd_a(t)} - z_p \right\|}{\left\| xx_a^{xd_a(t)} \right\|} \tag{10}$$

In Figure 3, a sample MST is formed using 14 subset centroids as the nodes and 13 branches are shown as the red lines connecting the closest nodes (in a minimum span). Breaking the 3 longest branches (shown as the dashed lines) thus reveals the 4 groups (G1,..,G4) among which one centroid yielding the minimum $f(a, j)$ can then be selected as an individual dimension of the *aGB* particle with 4 dimensional components (i.e. d=K=4, $xx_{aGB,j}^{K}(t), \forall j \in [1, K]$).

V. Experimental Results

An extensive set of experiments was conducted over the two application domains discussed in the previous section and the results will be presented in the following sub-sections.

A. Nonlinear Function Minimization

Both proposed techniques, the standalone MD PSO and MD PSO with FGBF, are tested over 7 benchmark functions given in Table V. We use the termination criteria as the combination of the maximum number of iterations allowed (*iterNo* = 5000) and the cut-off error ($\varepsilon_C = 10^{-4}$). Table V also presents both positional, $\pm x_{\max}$, and dimensional $[D_{\min}, D_{\max}]$ range values whereas others are empirically set as $V_{\max} = x_{\max}/2$ and $VD_{\max} = 18$, respectively. Unless stated otherwise, these range values are used in all experiments presented in this section.

The first set of experiments was performed for comparative evaluation of the standalone MD PSO vs. *bPSO* over both uni- and multi-modal functions. Figure 4 presents typical plots where both techniques are applied over the uni-modal function, *De Jong* using the swarm size, S=160. Red curves of both plots in Figure 4 and all the rest of the figures in this section belong to the GB particle (whether it is a new *gbest* or the *aGB* particle when FGBF is used) and the corresponding plots of the blue curve is obtained from the *gbest* particle when the termination criteria is met (e.g. *gbest* = 74 for *bPSO* and $f(y_{74}(158)) = 9.21 \times 10^{-5} < \varepsilon_C$). Naturally, the true dimension ($d_0 = 20$) is set

in advance for the *bPSO* process and it converges to the global optima within 158 iterations as shown in the right plot whereas MD PSO spent 700 iterations to have the GB particle in the target dimension ($d_0 = 20$) and then only 80 iterations more to satisfy the termination criteria. Recall that its objective is to find the true dimension where the global optimum exists and at this dimension, its internal process becomes identical with the *bPSO*. Yet in the overall sense, the standalone MD PSO is slower, but over an extensive set of experiments we observed that it has the same convergence behavior to the global optima with the *bPSO*. For instance, their performance is degraded in higher dimensions, e.g. for the same function but at $d_0 = 50$, both require -on the average- 5 times more iterations to find the global minimum.

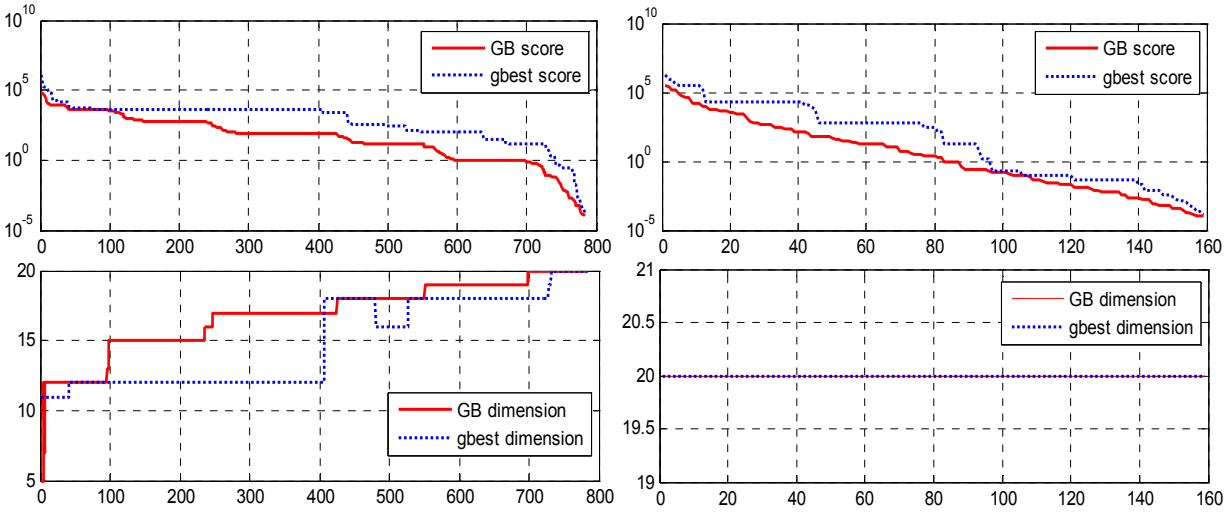

Figure 4: Fitness score (top in log-scale) and dimension (bottom) plots vs. iteration number for MD PSO (left) and *bPSO* (right) operations both of which run over *De Jong* function.

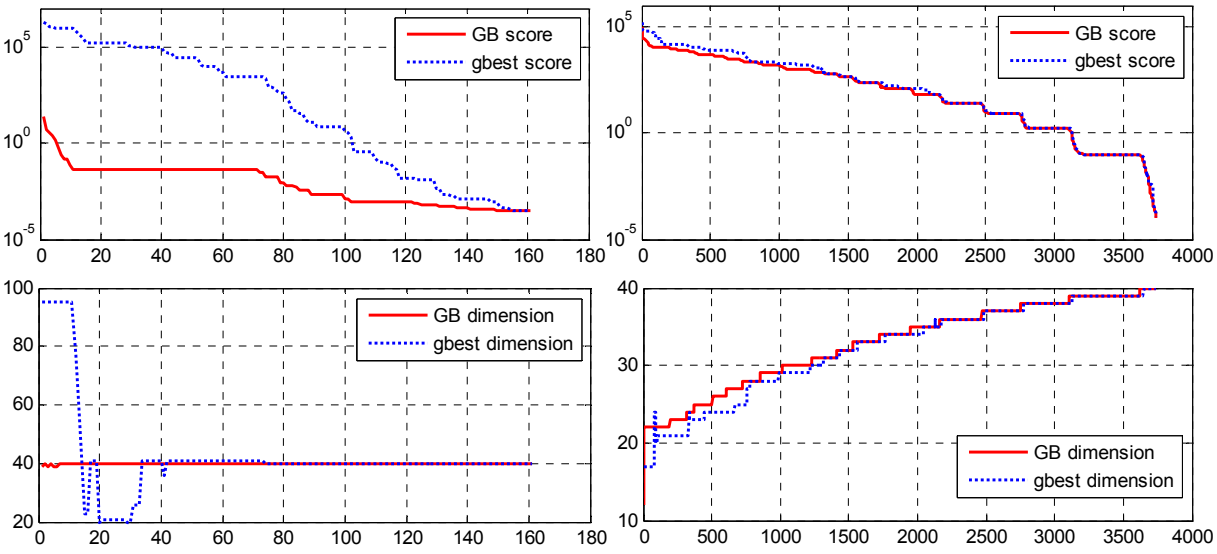

Figure 5: Fitness score (top in log-scale) and dimension (bottom) plots vs. iteration number for a MD PSO run over *Sphere* function with (left) and without (right) FGBF.

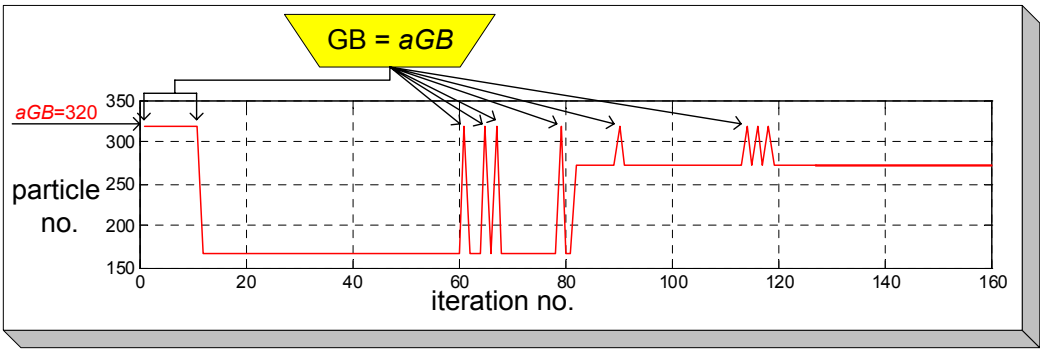

Figure 6: Particle index plot for the MD PSO with FGBF operation shown in Figure 5.

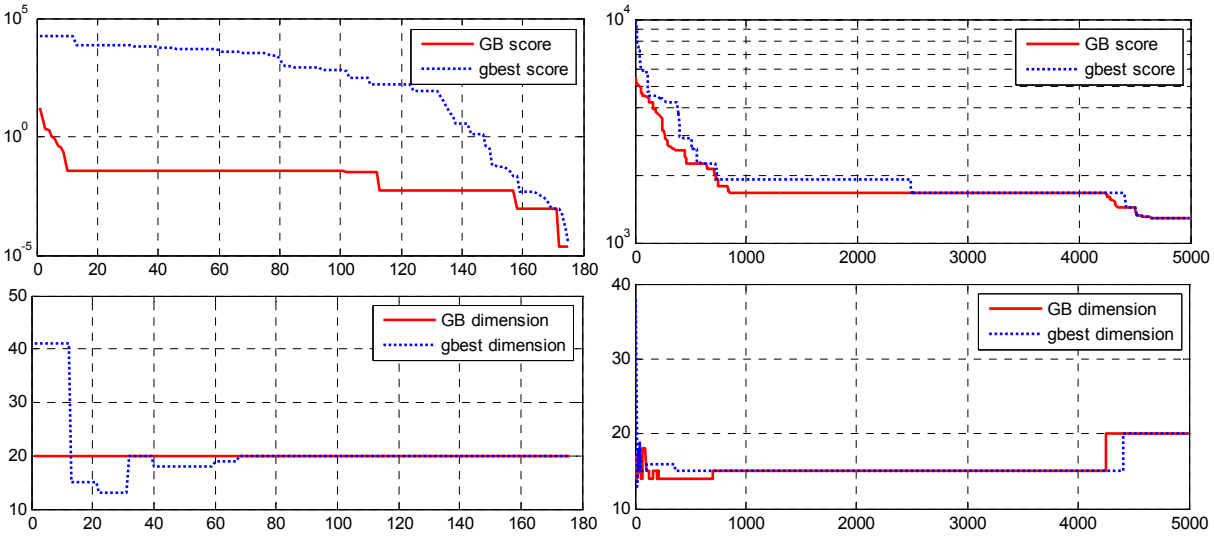

Figure 7: Fitness score (top in log-scale) and dimension (bottom) plots vs. iteration number for a MD PSO run over *Schwefel* function with (left) and without (right) FGBF.

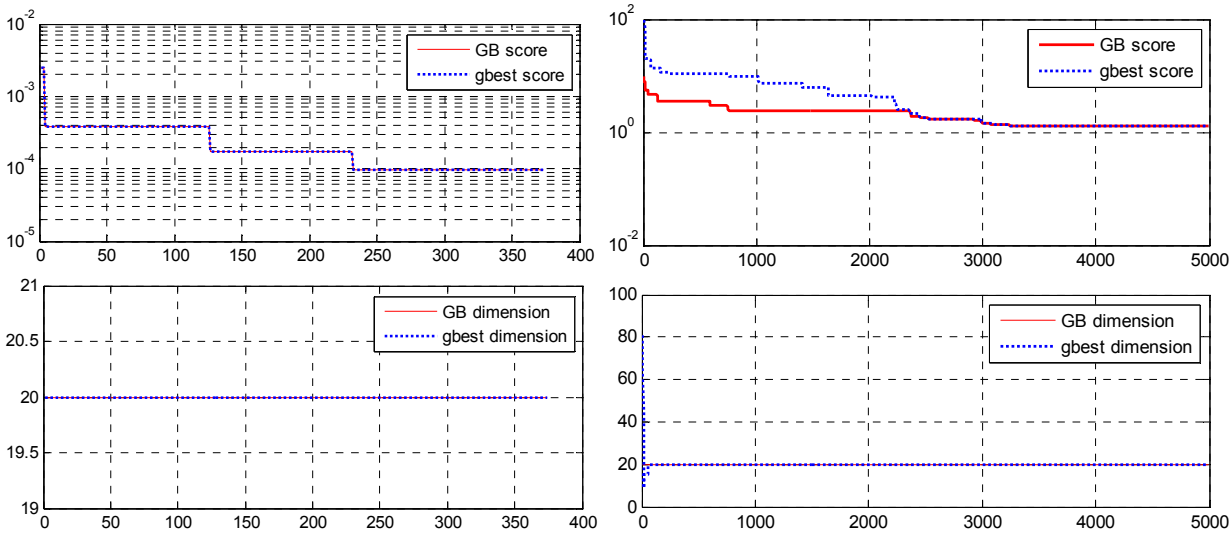

Figure 8: Fitness score (top in log scale) and dimension (bottom) plots vs. iteration number for a MD PSO run over *Giunta* function with (left) and without (right) FGBF.

A significant speed improvement can be achieved when MD PSO is performed with FGBF. A typical MD PSO run using the swarm size, S=320, over another uni-modal function, *Sphere*, but at a higher (target) dimension, is shown in Figure 5. Note that the one with FGBF (left) took only 160 iterations whereas the standalone MD PSO (right) is completed within 3740 iterations. Note also that within a few iterations, the process with FGBF already

found the true dimension, $d_0 = 40$, and after only 10 iterations, the *aGB* particle already came in a close vicinity of the global minimum, (i.e. $f(xy_{aGB}^{40}(10)) \cong 4 \times 10^{-2}$). As shown in Figure 6, the particle index plot for this operation clearly shows the time instances where *aGB* (with index number 320) becomes the GB particle, e.g. the first 14 iterations and then occasionally in the rest of the process.

Besides the significant speed improvement for uni-modal functions, the primary contribution of FGBF technique becomes most visible when applied over multi-modal functions where the *bPSO* (and the standalone MD PSO) is generally not able to converge to the global optimum even at the low dimensions. Figure 7 and Figure 8 present two (standalone MD PSO vs. MD PSO with FGBF) applications (using a swarm size 320) over *Schwefel* and *Giunta* functions at $d_0 = 20$. Note that when FGBF is used, MD PSO can directly have the *aGB* (as being the GB) particle in the target dimension ($d_0 = 20$) at the beginning of the operation, furthermore, the PSO process benefits from having an *aGB* particle that is indeed in a close vicinity of the global minimum. This eventually helps the swarm to move towards the *right* direction thereafter. Without this mechanism, both standalone PSO applications are eventually trapped into local minima due to the highly multi-modal nature of these functions. This is quite evident in the right-hand plots of both figures, and except for few minority cases, also true for the other multi-modal functions. In higher dimensions standalone MD PSO applications over multi-modal functions yield even worse results such as earlier traps to local minima and possibly in a wrong dimension. For example in standalone MD PSO operations over *Schwefel* and *Giunta* with $d_0 = 80$, the GB scores at *t* = 4999 ($f(x\hat{y}^{80})$) are 8955.39 and 1.83, respectively.

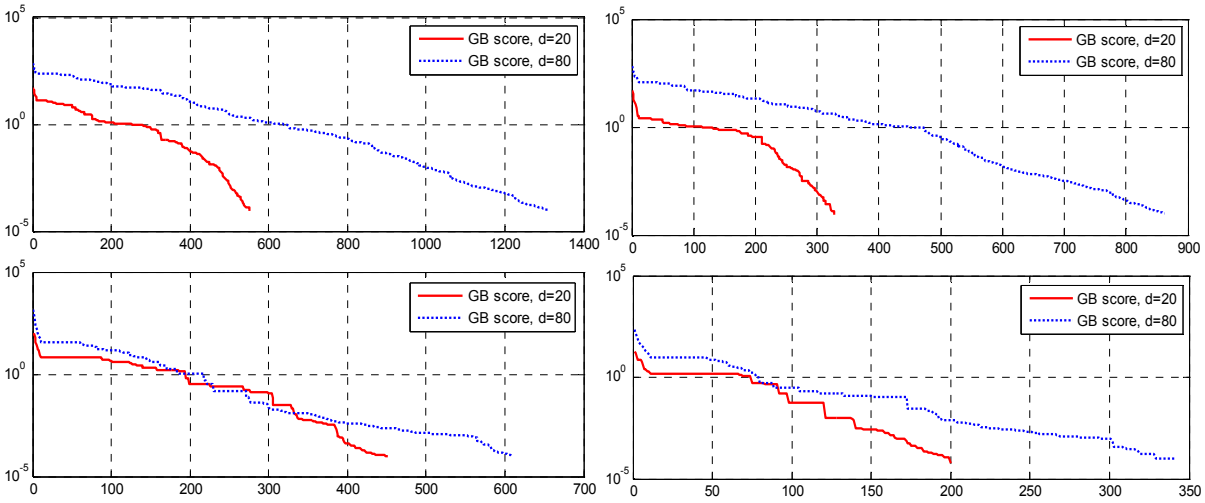

Figure 9: MD PSO with FGBF operation over *Griewank* (top) and *Rastrigin* (bottom) functions with $d_0 = 20$(red) and $d_0 = 80$(blue) using the swarm size, *S*=80 (left) and *S*=320 (right).

An observation worth mentioning here is that MD PSO with FGBF is usually affected by the higher dimensions but its performance degradation usually occurs as a certain amount of delay, not as the entrapment to a local minimum. For instance, when applied over *Schwefel* and *Giunta* at $d_0 = 80$, the convergence to global optima is still achieved only in a slightly delayed manner, i.e. in 119 and 484 iterations respectively. Moreover, Figure 9 presents fitness plots for applications of MD PSO with FGBF using two different swarm sizes over two more multi-modal functions, *Griewank* and *Rastrigin*. Similar to earlier results, the global minimum in the true dimension is reached for both functions; however, operations at $d_0 = 20$ (red curves) take usually a few hundreds

of iterations less than the ones at $d_0 = 80$ (blue curves).

Unlike *bPSO*, the swarm size has a direct effect on the performance of MD PSO with FGBF, that is, a larger swarm size increases the speed performance, which is quite evident in Figure 9 between the corresponding plots in the left- and the right-side. This is due to the fact that with the larger swarm size, the probability of having better dimensional components (closer to the global optimum at that dimension) of the *aGB* particle increases, thus yielding a better *aGB* particle formation in general. Note that this is also clear in the plots at both sides, i.e. at the beginning (say within the first 10-15 iterations when *aGB* is usually the GB particle) the drop in the fitness score is much steeper on the right-hand plots with respect to the ones on the left.

For an overall performance evaluation both proposed methods are tested over 7 benchmark functions using three different swarm sizes (160, 320 and 640) and target dimensions (20, 50 and 80). For each setting, 100 runs are performed and the 1st and 2nd order statistics (mean, μ and standard deviation, σ) of the operation time (total number of iterations) and the two components of the solution, the fitness score achieved in the resulting dimension (*dbest*), are presented in Table VI. During each run, the operation terminates when the fitness score drops below the cut-off error ($\varepsilon_C = 10^{-4}$) and it is assumed that the global minimum of the function in the target dimension is reached, henceforth, the score is set to 0 and obviously, $dbest = d_0$. Therefore, for a particular function, target dimension, d_0 and swarm size, *S*, obtaining μ=0 as the average score means that the method converges to the global minimum in the target dimension at every run. On the other hand, having the average iteration number as 5000 indicates that the method cannot converge to the global minimum at all, instead it gets trapped in a local minimum. The statistical results enlisted in Table VI approve earlier observations and remarks about the effects of modality, swarm size and dimension over the performance (both speed and accuracy). Particularly for the standalone MD PSO application, increasing the swarm size improves the speed of convergence wherever the global minimum is reached for uni-modal functions, *Sphere* and *De Jong*, whilst reducing the score significantly on the others. The score reduction is particularly visible on higher dimensions, e.g. for $d_0 = 80$, compare the highlighted average scores of the top 5 functions. Note that especially on *De Jong* at $d_0 = 50$, none of the standalone MD PSO runs with *S*=160 can converge to the global minimum whilst they all can with a higher swarm population (i.e. 320 or 640).

Both dimension and modality have a direct effect on the performance of the standalone MD PSO. On uni-modal functions, its convergence speed decreases with increasing dimension, e.g. see the highlighted average values of the iteration numbers for *Sphere* at $d_0 = 20$ vs. $d_0 = 50$. On multi-modal functions, regardless of the dimension and swarm size, all standalone MD PSO runs trap in local minima (except perhaps few runs on *Rosenbrock* at $d_0 = 20$); however, the fitness performance still depends on the dimension, that is, the final score tends to increase in higher dimensions indicating an earlier entrapment at a local minimum. Regardless of the swarm size, this can easily be seen in all multi-modal functions except *Griewank* and *Giunta* both of which show higher modalities in lower dimensions. Especially *Griewank* becomes a plain *Sphere* function when the dimensionality exceeds 20-25. This is the reason of the performance improvement (or score reduction) from $d_0 = 20$ to $d_0 = 50$ but note that the worst performance (highest score average) is still encountered at $d_0 = 80$.

Table VI: Statistical results from 100 runs over 7 benchmark functions

Functions			Standalone MD PSO						MD PSO with FGBF					
			Score		Iter. No		*dbest*		Score		Iter. No		*dbest*	
	S	d_0	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Sphere	**160**	**20**	0	0	**1475**	117	20	0	0	0	**166**	27	20	0
		50	0	0	**4605**	280	50	0	0	0	**172**	37	50	0
		80	**45.066**	61.23	4659	1046	78.5	1.234	0	0	169	34	80	0
	320	**20**	0	0	**1024**	58	20	0	0	0	**133**	16	20	0
		50	0	0	**3166**	631	50	0	0	0	**131**	25	50	0
		80	**5.949**	7.584	4712	626	79	0.858	0	0	126	24	80	0
	640	**20**	0	0	**932**	161	20	0	0	0	**93**	15	20	0
		50	0	0	**2612**	701	50	0	0	0	**101**	15	50	0
		80	**0.317**	0.462	4831	343	79.7	0.47	0	0	95	15	80	0
De Jong	**160**	**20**	0	0	**1047**	74	20	0	0	0	**6**	1	20	0
		50	0.705	0.462	5000	0	49.3	0.47	0	0	18	28	50	0
		80	**5184.4**	1745.7	5000	0	71.65	0.745	0	0	102	24	80	0
	320	**20**	0	0	**833**	66	20	0	0	0	**4**	1	20	0
		50	0	0	**4004**	166	50	0	0	0	**6**	1	50	0
		80	**811.02**	323.37	5000	0	74.85	0.587	0	0	19	20	80	0
	640	**20**	0	0	**643**	69	20	0	0	0	**2**	1	20	0
		50	0	0	**3184**	145	50	0	0	0	**4**	1	50	0
		80	**67.452**	24.75	5000	0	77.3	0.47	0	0	7	1	80	0
Rosenbrock	**160**	**20**	0.0008	0.0004	4797	903	20	0	0	0	1619	1785	20	0
		50	0.009	0.003	5000	0	50	0	0	0	398	149	50	0
		80	**7.617**	7.834	500	0	78.75	0.786	0	0	367	98	80	0
	320	**20**	0.001	0.002	4317	1221	20	0	0	0	325	239	20	0
		50	0.009	0.004	5000	0	50	0	0	0	257	33	50	0
		80	**0.49**	0.495	5000	0	79.65	0.587	0	0	258	47	80	0
	640	**20**	0.0009	0.002	4560	982	20	0	0	0	160	37	20	0
		50	0.006	0.003	5000	0	50	0	0	0	193	58	50	0
		80	**0.018**	0.005	5000	0	80	0	0	0	189	23	80	0
Rastrigin	**160**	**20**	2.137	0.565	5000	0	19.35	0.49	0	0	**304**	40	20	0
		50	24.051	5.583	5000	0	48.2	0.41	0	0	**392**	49	50	0
		80	**212.375**	165.196	5000	0	77.15	1.871	0	0	**375**	55	80	0
	320	**20**	1.506	0.531	5000	0	19.35	0.489	0	0	228	21	20	0
		50	9.788	4.181	5000	0	48.95	0.224	0	0	277	47	50	0
		80	**77.317**	31.909	5000	0	77.45	0.605	0	0	272	65	80	0
	640	**20**	0.96	0.527	5000	0	19.501	0.513	0	0	**164**	15	20	0
		50	7.308	2.581	5000	0	49.3	0.657	0	0	**206**	30	50	0
		80	**28.709**	8.399	5000	0	78.3	0.933	0	0	**195**	47	80	0
Griewank	**160**	**20**	3.262	14.485	4163	1486	19.101	4.025	0	0	438	78	20	0
		50	0.232	0.271	4779	673	49.2	0.696	0	0	725	36	50	0
		80	**9.499**	5.731	5000	0	73.9	2.435	0	0	1029	57	80	0
	320	**20**	0.019	0.016	4219	1601	20	0	0	0	395	76	20	0
		50	0.007	0.009	4537	454	50	0	0	0	618	34	50	0
		80	**3.059**	1.857	5000	0	76.45	1.099	0	0	866	43	80	0
	640	**20**	0.017	0.017	4205	1638	20	0	0	0	325	74	20	0
		50	0.016	0.02	4306	877	50	0	0	0	531	37	50	0
		80	**0.675**	0.385	5000	0	78.25	0.55	0	0	710	44	80	0
Schwefel	**160**	**20**	1432.3	336.76	5000	0	16.30	1.418	0	0	215	33	20	0
		50	6036.2	597.33	5000	0	45.45	1.234	0	0	199	29	50	0
		80	11304	1489.8	5000	0	74.7	2.226	0	0	161	35	80	0
	320	**20**	1261.7	320.709	5000	0	16.20	1.508	0	0	168	36	20	0
		50	5288.8	576.132	5000	0	45.55	1.394	0	0	146	24	50	0
		80	8882.8	1434.3	5000	0	75.8	1.814	0	0	120	22	80	0
	640	**20**	946.612	264.635	5000	0	16.85	1.268	0	0	121	21	20	0
		50	4412.2	921.062	5000	0	45.2	2.067	0	0	103	15	50	0
		80	8032.1	1180.9	5000	0	75.2	2.447	0	0	85	12	80	0
Giunta	**160**	**20**	1.488	0.69	5000	0	20	0	0	0	**793**	626	20	0
		50	0.776	0.207	5000	0	50	0	0	0	**699**	281	50	0
		80	1.131	0.14	5000	0	80	0	0	0	**863**	293	80	0
	320	**20**	1.003	0.582	5000	0	20	0	0	0	128	92	20	0
		50	0.543	0.127	5000	0	50	0	0	0	283	113	50	0
		80	1.140	0.756	5000	0	80	0	0	0	456	115	80	0
	640	**20**	0.675	0.517	5000	0	20	0	0	0	**1**	1	20	0
		50	0.418	0.140	5000	0	50	0	0	0	**4**	4	50	0
		80	0.789	0.145	5000	0	80	0	0	0	**20**	6	80	0

As the entire statistics in the right side of Table VI indicate, MD PSO with FGBF finds 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. Moreover, the mutual application of the proposed techniques significantly improves the convergence speed, e.g. compare the highlighted average iteration numbers with the standalone MD PSO's. Dimensionality, modality and swarm size might still be important factors over the speed and have the same effects as mentioned earlier, i.e. the speed degrades with modality and dimensionality, whereas it improves with increasing swarm size. Their effects, however, vary significantly among the functions, e.g. as highlighted in Table VI, the swarm size can enhance the speed radically for *Giunta* but only merely for *Griewank*. The same statement can be made concerning the dimensionality of *De Jong* and *Sphere*.

Based on the results in Table VI, we can perform comparative evaluations with some of the promising PSO variants such as [2], [15], [44] and [45] where similar experiments are performed over some or all of these benchmark functions. They have, however, the advantage of fixed dimension whereas MD PSO with FGBF finds the true dimension on the fly. Furthermore, it is rather difficult to make speed comparisons since none of them really find the global minimum for most functions; instead they have demonstrated some incremental performance improvements in terms of score reduction with respect to some other competing technique(s). For example in [2], a tournament selection mechanism is formed among particles and the method is applied over 4 functions (*Sphere*, *Rosenbrock*, *Rastrigin* and *Griewank*). Although the method is performed over a reduced positional range, ±15, and at low dimensions (10, 20 and 30), they got varying average scores between the range, {0.3, 1194}. As a result, they reported both better and worse performances than the *bPSO*, depending on the function. In [15], *bPSO* and two PSO variants, *GCPSO* and mutation-extended PSO over three neighborhood topologies are applied to some common multi-modal functions, *Rastrigin, Schwefel* and *Griewank*. Although the dimension is rather low (30), none of the topologies over any PSO variant converged to the global minimum and they reported average scores varying in the range of {0.0014, 4762}. In [44], a diversity guided PSO variant, *ARPSO*, along with two competing methods, *bPSO* and *GA* are applied over the multi-modal functions (*Rastrigin, Rosenbrock* and *Griewank*) at three different dimensions (20, 50 and 100). The range is kept quite reduced for *Rosenbrock* and *Rastrigin*, ±100 and ±5.12, respectively and for each run; the number of evaluations (product of iterations and the swarm size) is kept from 400000 to 2000000, depending on the dimensionality. The experimental results have shown that none of the three methods converged to the global minimum except ARPSO over (only) *Rastrigin* at dimension 20. Only when ARPSO runs till stagnation where no fitness improvements occur within 200000 evaluations, it can also find the global minimum over *Rastrigin* at higher dimensions (50 and 100). However, in practical sense, this indicates that the total number of iterations might be in the magnitude of 10^5 or even more. Recall that the number of iterations required for MD PSO with FGBF convergence to the global minimum is less than 400 for any dimension. *ARPSO* performed better than *bPSO* and *GA* over *Rastrigin* and *Rosenbrock* but worse over *Griewank*. The CPSO proposed in [52] was applied over five functions among which four of them are common (*Sphere*, *Rastrigin, Rosenbrock* and *Griewank*). The dimension of all functions is fixed to 30 and in this dimension, CPSO performed better than *bPSO* in 80% of the experiments. Finally in [45] dynamic sociometries via *ring* and *star* have been introduced among the swarm particles and the performance of various combinations of swarm size and sociometry over 6 functions (the ones used in this paper except *Schwefel*) has been reported. Although the tests are performed over comparatively reduced positional ranges and at a low dimension (30), the experimental results indicate that none of the sociometry and swarm size combination converged to the global minimum of multi-modal functions except only for some dimensions of the *Griewank* function.

B. Data Clustering

In order to test the application of the proposed techniques over clustering, we create 11 synthetic data spaces as shown in Figure 10. For illustration purposes each data space is formed in 2D; however, clusters are formed with different shapes, densities, sizes and inter-cluster distances to test the robustness of clustering application of the proposed techniques against such variations. Furthermore, recall that the number of clusters determines the (true) dimension of the solution space in a PSO application and hence it is also kept varying among data spaces to test the converging accuracy to the true (solution space) dimension. As a result, significantly varying complexity levels are established among the 11 data spaces to perform a general-purpose evaluation of each technique.

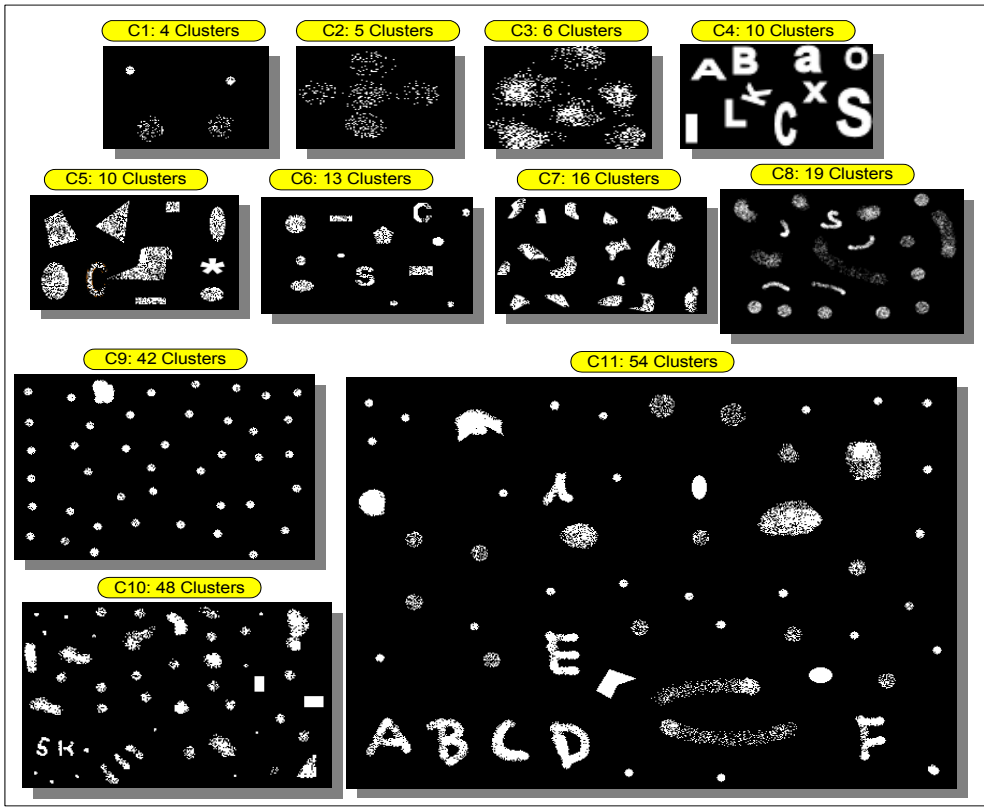

Figure 10: 2D synthetic data spaces carrying different clustering schemes.

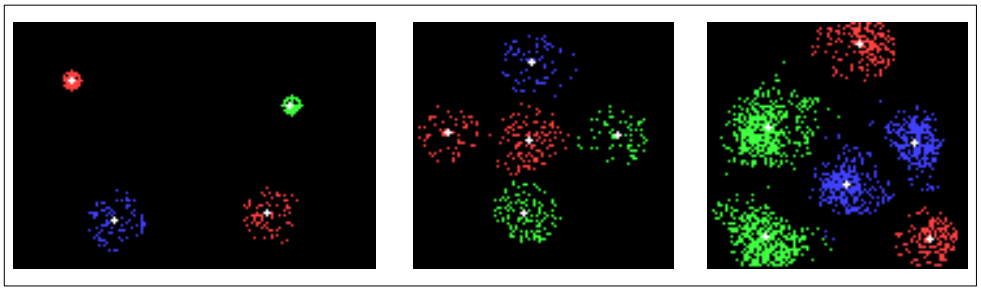

Figure 11: The standalone MD PSO (and *bPSO*) clustering for data spaces C1-3 shown in Figure 10.

Unless stated otherwise, the maximum number of iterations is set to 2000; however, the use of cut-off error as a termination criterion is avoided since it is not feasible to set a unique ε_C value for all clustering schemes. The same range values given in Section V.A are also used in all experiments except the positional range, $\pm x_{\max}$, since it can now be set simply as the natural boundaries of the 2D data space. The first set of clustering operations is performed

for a comparative evaluation of the standalone MD PSO vs. *bPSO* over the simple data spaces where they can yield accurate results, e.g. the results of clustering over the three data spaces at the top row in Figure 10 are shown in Figure 11 where each cluster is represented in one of the three color codes (red, green and blue) for illustration purposes and each cluster centroid (each dimensional component of the *gbest* particle) is shown with a white '+'.

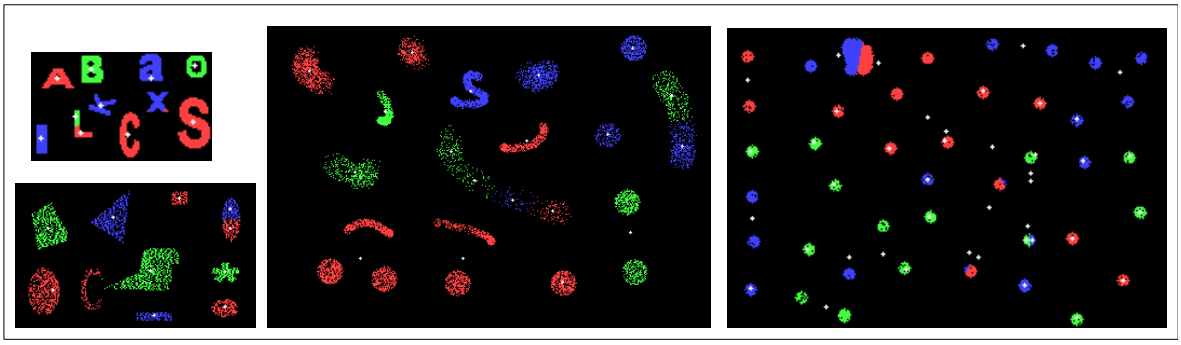

Figure 12: Erroneous *bPSO* clustering over data spaces C4, C5, C6 and C9 shown in Figure 10.

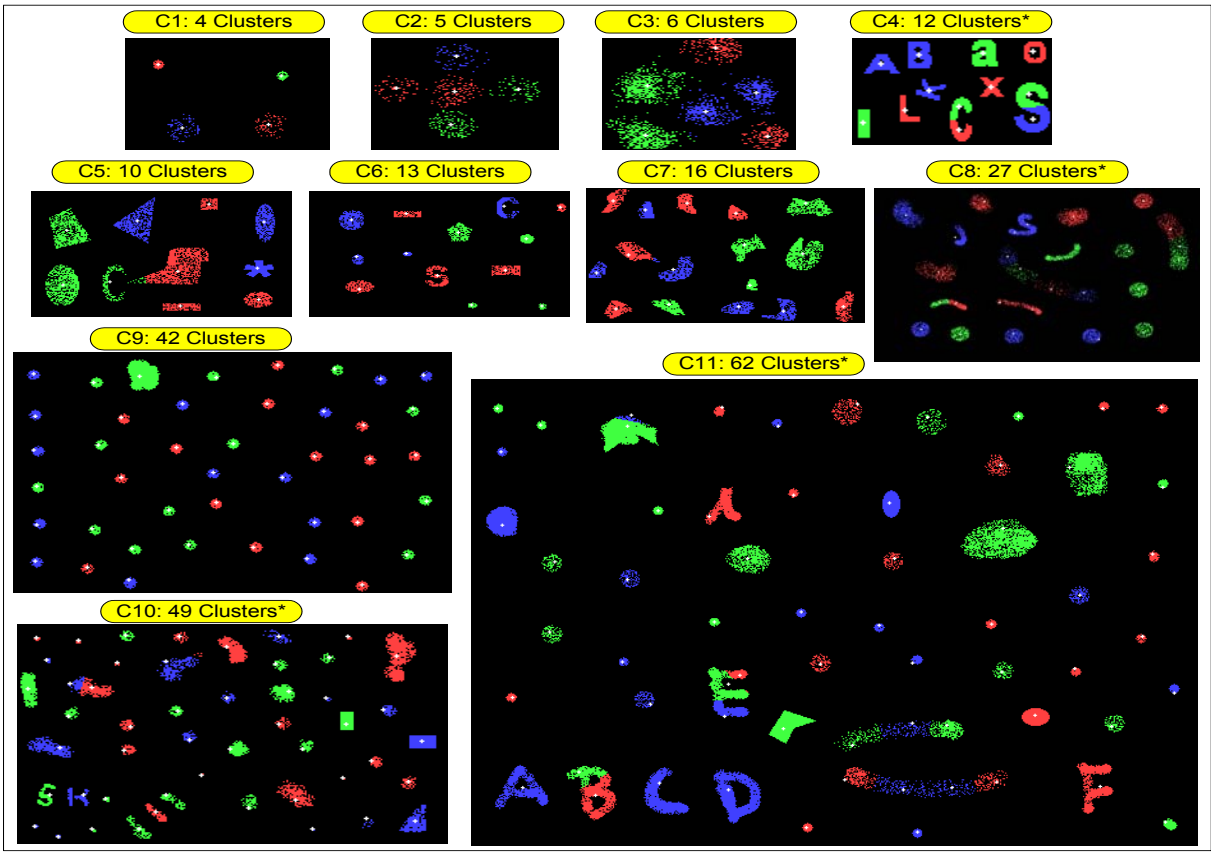

Figure 13: Typical clustering results via MD PSO with FGBF. Over-clustered samples are indicated with *.

The accuracy of both *bPSO* and MD PSO tends to degrade with the increasing dimensionality and complexity. Figure 12 presents typical clustering results for $K \geq 10$ and whilst running each *bPSO* operation till iteration number reaches 20000 (i.e. stagnation). K=10 is indeed not a too high dimension for *bPSO* but it particularly suffers from the highly complex clustering schemes in C4 and C5 (i.e. varying sizes, shapes and densities among clusters). Over a simpler data space, e.g. C6 with 13 clusters, we noticed that *bPSO* occasionally yields accurate clustering but for those data spaces with 20-25 clusters or more, clustering errors become inevitable regardless of the level of complexity and errors tend to increase significantly in higher dimensions as a natural consequence of

earlier local traps. A typical example is C9, which has 42 clusters in the simplest form (uniform size, shape and density) and the clustering result presents many over- and under- clustering schemes with many occasional miss-located centroids. Much worse performance can be expected from their applications for C10 and C11.

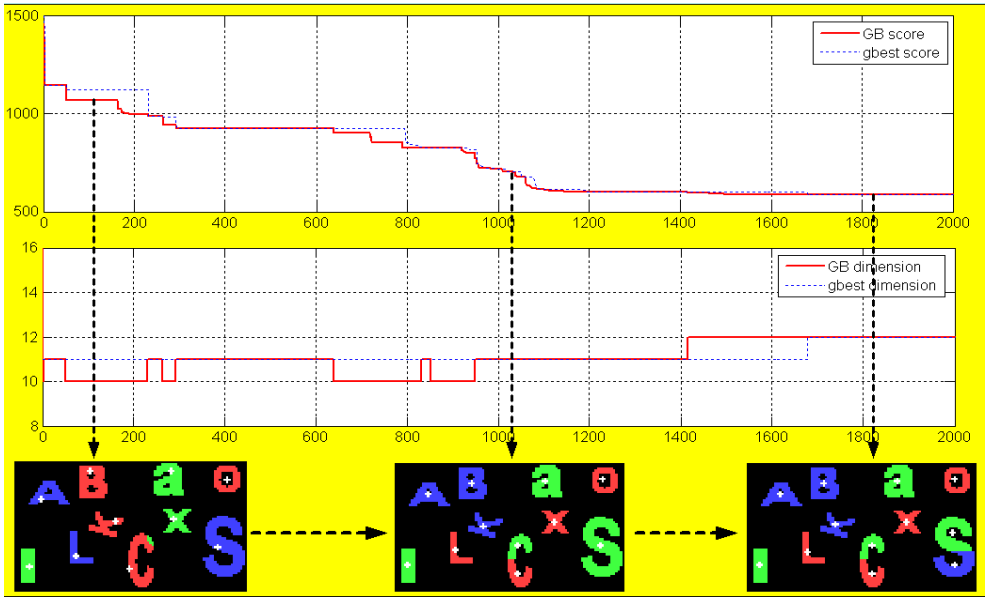

Figure 14: Fitness score (top) and dimension (bottom) plots vs. iteration number for a MD PSO with FGBF clustering operation over C4. 3 clustering snapshots at iterations 105, 1050 and 1850, are presented below.

As stated earlier, MD PSO with FGBF, besides its speed improvement, has its primary contribution over the accuracy of the clustering, i.e. converging to the true number of clusters, K, and correct localization of the centroids. As typical results shown in Figure 13, MD PSO with FGBF meets the expectations on clustering accuracy, but occasionally results in a slightly higher number of clusters. This is due to the use of a simple but quite impure validity index in (9) as the fitness function and for some complex clustering schemes it may, therefore, yields its minimum score at a slightly higher number of clusters. A sample clustering operation validating this fact is shown in Figure 14. Note that the (true) number of clusters is 10, which is eventually reached at the beginning of the operation, yet the minimum score achieved with K=10 (~750) remains higher than the one with K=11 ((~610) and than the final outcome, K=12 (~570) too. The main reason for this is that the validity index in (9) over long (and loose) clusters such as 'C' and 'S' in the figure, yields a much higher fitness score with one centroid than two or perhaps more and therefore, over all data spaces with such long and loose clusters (e.g. C4, C8, C10 and C11), the proposed method yields a slight over-clustering but never under-clustering. Improving the validity index or adapting a more sophisticated one such as Dunn's index [12] or many others, might improve the clustering accuracy; however, this is beyond the scope of this paper.

An important observation worth mentioning is that clustering complexity (modality) affects the proposed methods' mutual performance much more than the total cluster number (dimension). For instance, MD PSO with FGBF clustering (with S=640) over data space C9 can immediately find out the true cluster number and accurate location of the centroids with a slight offset (see Figure 15), whereas this takes around 1900 iterations for C8. Figure 16 shows time instances where *aGB* (with index number 640) becomes the GB particle. It immediately (at the 1st iteration) provides a "near optimum" GB solution with 43 clusters and then the MD PSO process (at the 38th iteration) eventually finds the global optimum with 42 clusters (i.e. see the 1st snapshot in Figure 15). Afterwards the ongoing PSO process corrects the slight positional offset of the cluster centroids (e.g. compare 1st and 2nd

snapshots in Figure 15). So when the clusters are compact, uniformly distributed and have similar shape, density, and size, thus yielding the simplest form, it becomes quite straightforward for FGBF to select the 'most promising' dimensions with a greater accuracy. As the complexity (modality) increases, different centroid assignments and clustering combinations have to be assessed to converge towards the global optimum, which eventually becomes a slow and tedious process.

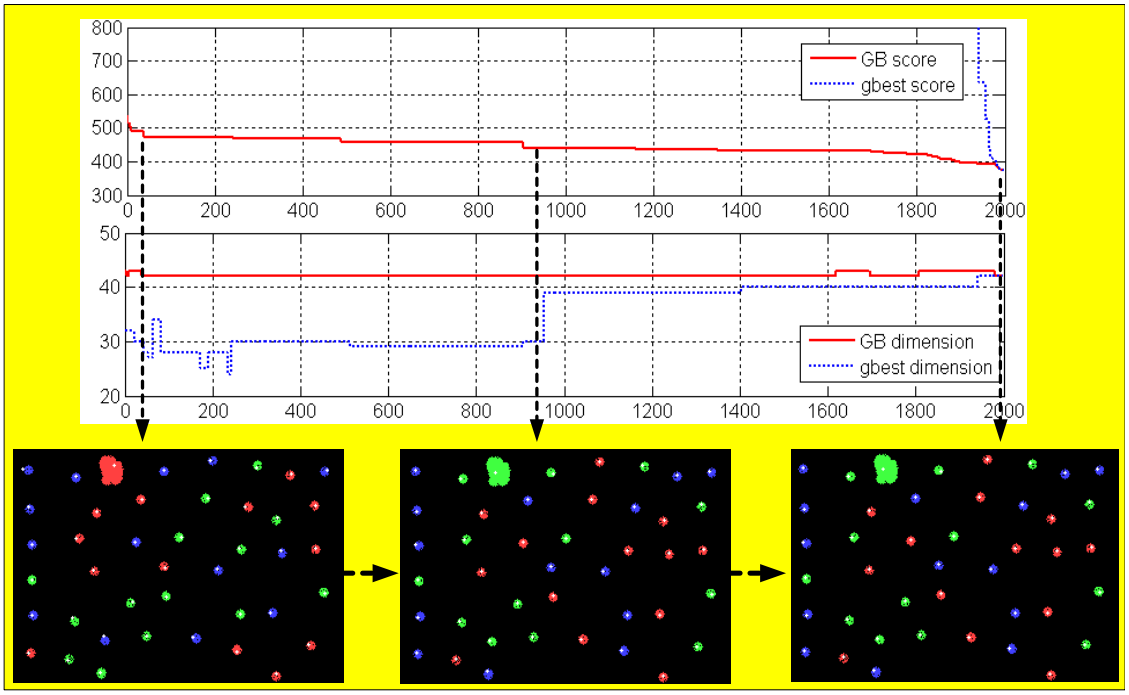

Figure 15: Fitness score (top) and dimension (bottom) plots vs. iteration number for a MD PSO with FGBF clustering operation over C9. 3 clustering snapshots at iterations 40, 950 and 1999, are presented below.

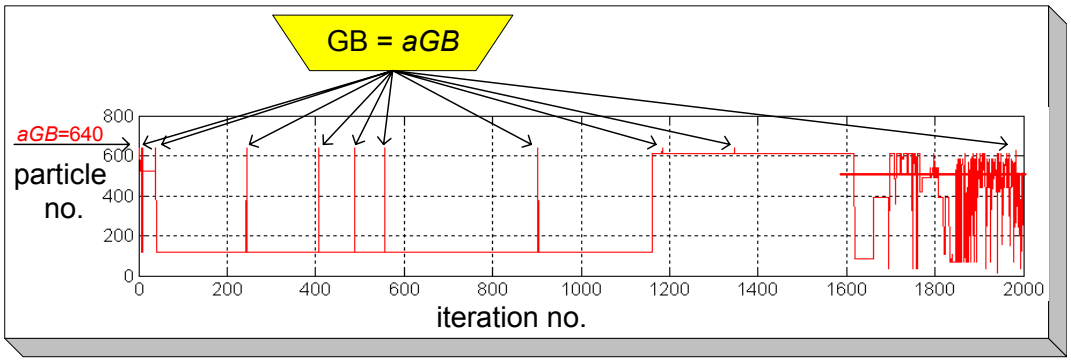

Figure 16: Particle index plot for the MD PSO with FGBF clustering operation shown in Figure 15.

Recall from the earlier discussion about the application of the proposed methods over nonlinear function minimization (both standalone MD PSO and MD PSO with FGBF), a certain speed improvement occurs in terms of reduction in iteration number and a better fitness score is achieved, when using a larger swarm. However, the computational complexity (per iteration) also increases since the number of evaluations (fitness computations) is proportional to the number of particles. The same trade-off also exists for clustering application and, furthermore, a significantly higher computational complexity of the mutual application of the proposed methods can occur due to the spatial MST grouping for the selection of the well-separated centroids. As explained in section IV.B, MST is the essence of choosing the "most promising" dimensions (centroids) so as to form the best possible *aGB* particle.

However, it is a costly ($O(N_{SS}^2)$) operation where N_{SS} is the subset size, which is formed by those dimensions (potential centroids) having at least one data item closest to it. Therefore, N_{SS} tends to increase if a larger swarm size is used and/or MD PSO with FGBF clustering is performed over large (with many data items) and highly complex data spaces.

Table VII presents average processing times per iteration over all sample data spaces and using 4 different swarm sizes. All experiments are performed on a computer with P-IV 3GHz CPU and 1GB RAM. Note that the processing times tend to increase in general when data spaces get larger (with more data items) but the real factor is the complexity. The processing for a highly complex data structure, such as C10, may require several times more computations (or time) than a simpler but comparable-size data space, such as C5. Therefore, on such highly complex data spaces, the swarm size should be kept low, e.g. $80 \le S \le 160$, for the sake of a reasonable processing time.

Table VII: Processing time (in msec) per iteration for MD PSO with FGBF clustering using 4 different swarm sizes. Number of data items is presented in parenthesis with the sample data space.

S	**C1 (238)**	**C2 (408)**	**C3 (1441)**	**C4 (1268)**	**C5 (3241)**	**C6 (1314)**	**C7 (3071)**	**C8 (5907)**	**C9 (2192)**	**C10 (3257)**	**C11 (12486)**
80	19.7	57	140	688	864.1	231.5	690.8	1734.5	847.7	4418.2	8405.1
160	31.7	104.9	357.3	1641	1351.3	465.5	2716.3	3842.8	1699.4	13693.7	26608.6
320	62.3	222.9	1748.8	4542.5	3463.9	1007	3845.7	7372.7	4444.5	55280.6	62641.9
640	153.4	512	3389.4	17046.4	8210.1	4004.5	11398.6	23669.8	14828.2	159642.3	212884.6

VI. CONCLUSIONS

In this paper, we proposed two novel PSO techniques, namely, MD PSO and FGBF, as a cure to common drawbacks of the family of PSO methods such as *a priori* knowledge of the search space dimension and pre-mature convergence to local optima. The first proposed technique, the (standalone) MD PSO, efficiently addresses the former drawback by defining a new particle formation 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. Such flexibility negates the requirement of setting the dimension in advance since swarm particles can now converge to the global solution at the optimum dimension, in a simultaneous manner.

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 premature convergence to local optima especially when multi-modal problems are optimized at 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 proposed 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" then the contemporary *gbest* particle. Therefore, for those problems where either the exact dimensional fitness evaluation or its approximation is possible, FGBF can be conveniently used to improve the global convergence ability without changing the native structure of the swarm.

In order to test and evaluate the MD PSO's performance over the first problem domain, the nonlinear function

minimization, seven benchmark functions were biased with a dimensional term so that they have the global minimum only at a particular dimension. We have shown that the standalone MD PSO (without FGBF) converges to the global minimum in the target dimension over uni-modal functions and we then investigated the effects of swarm size, dimensionality and modality over the performance –both accuracy and speed. A comparative evaluation with *bPSO* has shown that the standalone MD PSO is slower than *bPSO* due to its additional dimensional search process but has the same convergence behavior, as expected. The performance of both methods degrades with the increasing modality and dimensionality due to the aforementioned reasons. When used with FGBF, MD PSO exhibits such an impressive speed gain that their mutual performance surpasses *bPSO* by several magnitudes. Experimental results show that except in few minority cases, the convergence to the global minimum at the target dimension is achieved within fewer than 1000 iterations on the average, mostly only within few hundreds or even less. Yet the major improvement occurs in the convergence accuracy, both positional and dimensional. MD PSO with FGBF finds the global minimum at the target dimension for all runs over all functions without any exception. This is a substantial achievement in the area of PSO-based nonlinear function minimization.

Similar remarks can be made for the applications of *bPSO* and the standalone MD PSO over data clustering within which the (clustering) complexity can be thought of as synonymous to (function) modality, i.e. speed and accuracy performances of both methods drastically degrade with increasing complexity. Needless to say that the true number of clusters has to be set in advance for *bPSO* whereas MD PSO finds it on the fly and hence exhibits a slower convergence pace than *bPSO*. When it is performed with FGBF, a significant speed improvement is achieved and only such cooperation can provide accurate clustering results over complex data spaces. Since the clustering performance also depends on the validity index used, occasional over-clustering can be encountered where we have shown that such results indeed correspond to the global minimum of the validity index function used. As a result, the true number of clusters and accurate centroid localization are achieved at the expense of increased computational complexity due to the usage of MST. In order to keep the overall computational cost within feasible limits, we also investigated the effects of swarm size over complexity and recommended a proper range for a practical use.

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 used conveniently with (either of) them.

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, Oct. 25, 2007.

[2] P.J. Angeline, "Using Selection to Improve Particle Swarm Optimization", In *Proc. of the IEEE Congress on Evolutionary Computation*, pp. 84-89. IEEE Press, 1998.

[3] P. I. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences", In *Evolutionary Programming VII, Conf. EP'98, Springer Verlag, Lecture Notes in Computer Science No. 1447*, pp.601-410, California. USA, Mar. 1998.

[4] T. Back and H.P. Schwefel, "An overview of evolutionary algorithm for parameter optimization", *Evolution. Comput. 1*, pp. 1–23, 1993.

[5] T. Back and F. Kursawe, "Evolutionary algorithms for fuzzy logic: a brief overview", In *Fuzzy Logic and Soft Computing*, World Scientific, pp. 3–10, Singapore, 1995.

[6] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*, New York: Plenum, 1981.

[7] X. Chen, Y. Li, "A Modified PSO Structure Resulting in High Exploration Ability With Convergence Guaranteed", *IEEE Trans. on Systems, Man, and Cybernetics, Part B,* vol. 37, Issue 5, pp. 1271 – 1289, Oct. 2007.

[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] K. M. Christopher, K. D. Seppi, "The Kalman Swarm. A New Approach to Particle Motion in Swarm Optimization", In *Proc. of the Genetic and Evolutionary Computation Conf., GECCO*, pp. 140-150, 2004.
[10] M. Clerc, "The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization", In *Proc. of the IEEE Congress on Evolutionary Computation*, vol. 3, pp. 1951-1957, July 1999.
[11] D. L. Davies, and D.W. Bouldin, "A cluster separation measure", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pp. 224–227, 1979.
[12] J. C. Dunn, "Well separated clusters and optimal fuzzy partitions", *Journal of Cybernetics*, vol. 4, pp. 95–104, 1974.
[13] R. Eberhart, P. Simpson, and R. Dobbins, *Computational Intelligence. PC Tools,* Academic Press, Inc., Boston, MA, USA, 1996.
[14] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, John Wiley & Sons, 2005.
[15] S. C. Esquivel and C. A. Coello Coello, "On the Use of Particle Swarm Optimization with Multimodal Functions", In *IEEE Trans. on Evolutionary Computation*, vol. 2, pp. 1130-1136, 2003.
[16] U.M. Fayyad, G.P. Shapire, P. Smyth and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining,* MIT Press, Cambridge, MA, 1996.
[17] H. Frigui, R. Krishnapuram, "Clustering by competitive agglomeration", *Pattern Recognition*, vol. 30, pp. 1109-1119, 1997.
[18] D. Goldberg, Genetic *Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, pp. 1-25. MA, 1989.
[19] G. Hammerly, "Learning structure and concepts in data through data clustering", PhD thesis, June 26, 2003.
[20] G. Hammerly and C. Elkan, "Alternatives to the k-means algorithm that find better clusterings," in *Proc. of the 11th ACM CIKM*, pp. 600–607, 2002.
[21] M. Halkidi, M. Vazirgiannis, "Clustering Validity Assessment: Finding the Optimal Partitioning of a Data Set", In *Proc. of First IEEE Int. Conf. on Data Mining (ICDM'01)*, pp. 187-194, 2001.
[22] M. Halkidi, Y. Batistakis, M. Vazirgiannis, "On Cluster Validation Techniques", *Journal of Intelligent Information Systems*, vol. 17 no. 2, 3. pp. 107-145, 2001.
[23] H. Higashi and H. Iba, "Particle Swarm Optimization with Gaussian Mutation", In *Proc. of the IEEE Swarm Intelligence Symposium*, pp. 72-79, 2003.
[24] A. K. Jain, M.N. Murthy and P.J. Flynn, "Data Clustering: A Review", *ACM Computing Reviews*, Nov 1999.
[25] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer and its adaptive variant", *IEEE Trans. on Systems, Man, and Cybernetics, Part B,* vol. 35, Issue 6, pp. 1272 – 1282, Dec. 2005.
[26] B. Kaewkamnerdpong, and P. J. Bentley, "Perceptive Particle Swarm Optimization: An Investigation", In *Proc. of IEEE Swarm Intelligence Symposium*, pp. 169-176, California, 8-10 June 2005.
[27] J. Kennedy, R Eberhart., "Particle swarm optimization", in Proc. *of IEEE Int. Conf. On Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, 1995.
[28] J. Koza, *Genetic Programming: On the Programming of Computers by means of Natural Selection,* MIT Press, Cambridge, Massachussetts, 1992.
[29] 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.
[30] J. R. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem", *Proc. of AMS,* 71*,* 1956.
[31] J. J. Liang, A. K. Qin, "Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions", *IEEE Trans. On Evolutionary Computation*, vol. 10, no. 3, pp. 281-295, June, 2006.
[32] M. Lovberg, "Improving Particle Swarm Optimization by Hybridization of Stochastic Search Heuristics and Self-Organized Criticality", MSc thesis, Department of Computer Science, University of Aarhus, Denmark, 2002.
[33] 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.
[34] R. Mendes, J. Kennedy and J. Neves, "The fully informed particle swarm: Simpler, maybe better", *IEEE Trans. on Evolutionary Computation*, vol. 8, no 3, pp. 204–210, June 2004.
[35] M. Omran, A. Salman and A. P. Engelbrecht, "Image Classification using Particle Swarm Optimization", In *Conf. on Simulated Evolution and Learning*, vol. 1, pp. 370–374, 2002.
[36] M. G. Omran, A. Salman, and A.P. Engelbrecht, "Dynamic Clustering using Particle Swarm Optimization with Application in Image Segmentation", In *Pattern Analysis and Applications*, vol. 8, pp. 332-344, 2006.
[37] M. G. Omran, A. Salman, and A.P. Engelbrecht, *Particle Swarm Optimization for Pattern Recognition and Image Processing,* Springer Berlin, 2006.
[38] N.R. Pal and J. Biswas, "Cluster validation using graph theoretic concepts", *Pattern Recognition*, pp. 847–857, 1997.
[39] B. Peng, R.G Reynolds, and J. Brewster, "Cultural Swarms", In *Proc. of the IEEE Congress on Evolutionary Computation,* vol. 3, pp. 1965-1971, 2003.
[40] T. Peram, K. Veeramachaneni, and C.K. Mohan, "Fitness-Distance-Ratio based Particle Swarm Optimization", In *Proc. of the IEEE Swarm Intelligence Symposium*, pp. 174-181. IEEE Press, 2003.
[41] A. C. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Particle Swarm Optimization with Self-Adaptive Acceleration Coefficients", In *Proc. of the First Int. Conf. on Fuzzy Systems and Knowledge Discovery*, pp. 264-268, 2003.
[42] A. C. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Particle Swarm Optimiser with Time Varying Acceleration Coefficients", In *Proc. of the Int. Conf. on Soft Computing and Intelligent Systems*, pp. 240-255, 2002.

[43] R. G. Reynolds, B. Peng, and J. Brewster, "Cultural swarms II: Virtual algorithm emergence", In *Proc. of IEEE Congress on Evolutionary Computation 2003 (CEC 2003)*, pp. 1972-1979, Australia, 2003.
[44] J. Riget and J. S. Vesterstrom, "A Diversity-Guided Particle Swarm Optimizer - The ARPSO", Technical report, Department of Computer Science, University of Aarhus, 2002.
[45] M. Richards, D. Ventura, "Dynamic sociometry in particle swarm optimization," In *Proc. of the Sixth Int. Conf. on Computational Intelligence and Natural Computing*, pp. 1557-1560, North Carolina, September 2003.
[46] Y. Shi and R.C. Eberhart, "A Modified Particle Swarm Optimizer", In *Proc. of the IEEE Congress on Evolutionary Computation*, pp. 69-73, 1998.
[47] Y. Shi and R.C. Eberhart, "Fuzzy Adaptive Particle Swarm Optimization", In *Proc. of the IEEE Congress on Evolutionary Computation*, IEEE Press, vol. 1, pp. 101-106, 2001.
[48] J.T. Tou and R.C. Gonzalez, *Pattern Recognition Principles*, London, Addison-Wesley, 1974.
[49] R.H. Turi, "Clustering-based colour image segmentation", PhD Thesis, Monash University, Australia, 2001.
[50] F. Van den Bergh, "An Analysis of Particle Swarm Optimizers", PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.
[51] F. Van den Bergh and A.P. Engelbrecht, "A New Locally Convergent Particle Swarm Optimizer", In *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 96-101, 2002.
[52] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization", *IEEE Trans. on Evolutionary Computation*, vol. 3, pp. 225-239, June 2004.
[53] E.O. Wilson, *Sociobiology: The new synthesis*, Cambridge, MA: Belknap Press, 1975.
[54] X. Xie, W. Zhang, and Z. Yang, "A Dissipative Particle Swarm Optimization", In *Proc. of the IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1456-1461, 2002.
[55] X. Xie, W. Zhang, and Z. Yang, "Adaptive Particle Swarm Optimization on Individual Level", In *Proc. of the Sixth Int. Conference on Signal Processing*, vol. 2, pp. 1215-1218, 2002.
[56] X. Xie, W. Zhang, and Z. Yang, "Hybrid Particle Swarm Optimizer with Mass Extinction", In *Proc. of the Int. Conference on Communication, Circuits and Systems*, vol. 2, pp. 1170-1173, 2002.
[57] K. Yasuda, A. Ide, and N. Iwasaki, "Adaptive Particle Swarm Optimization", In *Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics*, vol. 2, pp. 1554 -1559, 2003.
[58] W-J. Zhang and X-F. Xie, "DEPSO: Hybrid Particle Swarm with Differential Evolution Operator", In *Proc. of the IEEE International Conference on System, Man, and Cybernetics*, vol. 4, pp. 3816-3821, 2003.
[59] B. Zhang and M. Hsu, "K-Harmonic Means - A Data Clustering Algorithm", Hewlett-Packard Labs Technical Report HPL-1999-124, 1999.
[60] W-J. Zhang, Y. Liu, and M. Clerc, "An adaptive PSO algorithm for reactive power optimization", *Advances in Power System Control Operation and Management (APSCOM)*, vol. 1, pp. 302-307, Hong Kong, 2003.