

MULTI-DIMENSIONAL PARTICLE SWARM OPTIMIZATION FOR DYNAMIC CLUSTERING

Serkan Kiranyaz, Turker Ince, Alper Yildirim and Moncef Gabbouj

Abstract: This paper addresses dynamic data clustering as an optimization problem and propose techniques for finding optimal (number of) clusters in a multi-dimensional data or feature space. In order to accomplish this objective we first 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. To address this problem we propose Fractional Global Best Formation (FGBF) technique, which basically collects all promising 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. We investigated both individual and mutual applications of the proposed techniques and demonstrated that the best clustering performance can be achieved by their mutual operation. In order to test and evaluate their clustering performance in terms of *accuracy*, *robustness* and *scalability*, a synthetic data-set, which contains *ground-truth* clusters and offers a broad range of complexity levels is used. An extensive set of experiments demonstrate that the proposed dynamic clustering technique based on MD PSO with FGBF can extract the optimal (number of) clusters by converging to the global optimum of the validity index function at the true dimension.

Index Terms: Particle Swarm Optimization, multi-dimensional search, Fractional Global Best Formation, dynamic clustering.

I. INTRODUCTION

Clustering is useful in pattern analysis, classification, machine learning, information retrieval, spatial segmentation and many other application domains. Cluster validity analysis is the assessment of the clustering method’s output using a specific criterion for optimality, i.e. the so-called clustering validity index function. Therefore, the optimality of any clustering method can only be assessed with respect to the validity index, which is defined over a specific data (feature) representation with a proper distance (similarity) metric. Given a validity index, clustering is a multi-modal problem especially in high dimensions, which contains many sub-optimum solutions such as over- and under-clustering. Therefore, well-known deterministic methods such as K-means, Max-Min [21], [10], FCM [4], [10], etc. are susceptible to get trapped to the closest local minimum since they are nothing but greedy descent methods, which start from a random point in the solution space and perform a *localized* search. This fact eventually turns the focus on stochastic

Evolutionary Algorithms (EAs) [2] such as Genetic Algorithms (GAs) [8], Genetic Programming (GP) [12], Evolution Strategies (ES), [3] and Evolutionary Programming (EP), [6], all of which are motivated by the natural evolution process and thus make use of evolutionary operators. The common point of all is that EAs are in population based nature and can perform a *globalized* search. So they may avoid becoming trapped in a local optimum and find the optimum solution; however, this is never guaranteed. Many works in the literature have shown that EA based methods outperform their deterministic counterparts, i.e. [5], [8], [10] and [19]. However, EAs suffer from the sensitivity of high parameter dependence (e.g. crossover and mutation probabilities, internal parameters, etc.), in that one has to tune some or all parameters to suit the application.

Conceptually speaking, Particle Swarm Optimization (PSO) [11], which has obvious ties with the EA family, lies somewhere in between GA and EP. Yet unlike GA, PSO has no complicated evolutionary operators such as *crossover*, *selection* and *mutation*. 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. Particles are initially distributed randomly over the search space and the goal is to converge to the global optimum of a function or a system. Several researchers have shown that PSO show better clustering performance than the aforementioned techniques [5], [15], [16]; however, when the problem is in multi-modal nature PSO may also become trapped in local optima [17] due to the premature convergence problem especially when the search space is of high dimensions [22].

In data clustering where the number of clusters corresponds to the dimension of the search space, the solution space is highly multi-modal and complex, particularly in high dimensions. Therefore, PSO has so far been applied to simple clustering problems [5], [15], [16], where the data space is limited and usually in low dimensions and the number of clusters (hence the solution space dimension) is kept reasonably low (e.g. <10). Moreover, all clustering methods mentioned earlier are in *static* nature, that is, the number of clusters has to be specified *a priori*. This is also true for PSO since in its basic form it can only be applied to a search space with a fixed dimension. Particularly in data clustering, the optimal (true) number of clusters is unknown and should thus be determined within the (PSO) process.

In this paper, we shall address data clustering as an optimization problem and propose techniques for finding optimal (number of) clusters in a multi-dimensional data or feature space. In order to accomplish this objective, the proposed techniques are designed to provide robust and generic solutions to the aforementioned problems of the evolutionary approaches, particularly the one which is the most promising among all, the PSO. The first proposed technique, the so-called Fractional Global Best Formation (FGBF), is designed to alleviate the premature

convergence problem by collecting all promising components from each particle and fractionally create an artificial Global Best (GB) particle, the *aGB*, which may guide the swarm better than the swarm's native *gbest* particle [5] in such a way that the swarm can converge to the global optimum (or near-optimum) solution even in high dimensions and usually in earlier stages. In order to achieve a *dynamic* clustering where the optimum number of clusters is also determined within the process, we shall further propose a novel optimization technique, the so-called Multi-Dimensional Particle Swarm Optimization (MD PSO), which re-forms the native structure of swarm particles in such a way that they can make inter-dimensional passes with a dedicated *dimensional* PSO process. Therefore, in a multidimensional search space where the optimum dimension is unknown, swarm particles can seek for both positional and dimensional optima. This eventually negates the necessity of setting a fixed dimension *a priori*, which is a common drawback for the family of swarm optimizers. The proposed techniques are generic and are also not linked to each other, i.e. one can be performed without the other but we shall show that the best clustering performance can be achieved by their mutual operation. Finally, no additional parameter is needed to perform both proposed techniques and MD PSO further 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 and data clustering. The proposed techniques, MD PSO and FGBF are presented in Section III whereas their dynamic clustering application is discussed in Section IV. Section V provides the experiments conducted over synthetic data-sets and discusses the results. Finally, Section VI concludes the paper.

II. RELATED WORK

A. 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 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 [11]) 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 [11]). 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 an evolutionary search algorithm in multi-dimensional (MD) search space, PSO exhibits some major problems similar to the aforementioned EAs. For instance it also shows the aforementioned parameter dependency where parameter variations may result in large performance shifts [14]. 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 [17] and it is the main cause of the premature convergence problem especially when the search space is of

high dimensions [22] and the problem to be optimized is multi-modal [17]. 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 [23].

Each particle a in the swarm, $\xi = \{x_1, \dots, x_a, \dots, x_S\}$, is represented by the following characteristics:

$x_{a,j}(t)$: j^{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 $\hat{y}(t) = y_{gbest}(t) = \min(y_1(t), \dots, y_S(t))$. 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:

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

where w is the inertia weight, [20] 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 [20], 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).

B. Data Clustering

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 [7]. 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 [10]. *K-means* [21] 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. 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.

A hard clustering technique based on the *bPSO* was first introduced by Omran et al. in [15] 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, $\xi = \{x_1, \dots, x_a, \dots, x_S\}$, is formed as $x_a(t) = \{c_{a,1}, \dots, c_{a,j}, \dots, c_{a,K}\} \Rightarrow x_{a,j}(t) = c_{a,j}$ 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 the data space dimension, N , is now different than the solution space dimension, K . Furthermore, the fitness (validity index) 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. A throughout survey about them can be found in [9]. 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 [15], 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. This is also true for dynamic clustering schemes, DCPSO [16]

and MEPSO [1], both of which eventually present results only in low dimensions and for simple data distributions.

III. MD PSO AND FGBF TECHNIQUES

In this section, we shall 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 shall then propose a second technique, the FGBF and present their mutual application for dynamic 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, $(D_{\min} \leq N \leq D_{\max})$. 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, $\xi = \{x_1, \dots, x_a, \dots, x_S\}$, 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 $xd_a(t)$

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

$x\hat{y}_j^d(t)$: 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

$x\tilde{d}_a(t)$: 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 ($x\tilde{d}_a(t)=3$). 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.

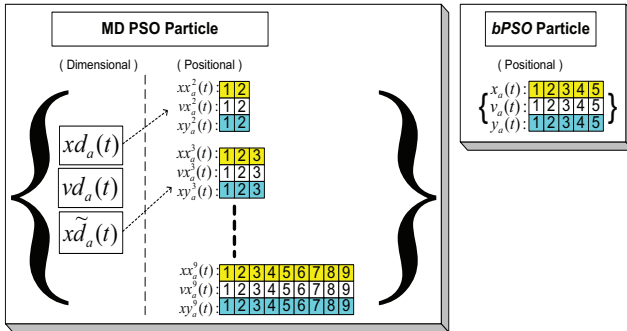


Figure 1: Sample MD PSO (right) vs. $bPSO$ (left) particle structures. For MD PSO $[D_{\min}=2, D_{\max}=9]$ and at the current time t , $xd_a(t)=2$ and $x\tilde{d}_a(t)=3$. For $bPSO$ $N=5$.

Let f denote the dimensional fitness function that is to be optimized within a certain dimension range, ($D_{\min} \leq N \leq D_{\max}$).

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,

$$xy_{a,j}^{xd_a(t+T)}(t+T) = \begin{cases} xy_{a,j}^{xd_a(t)}(t) & \text{if } f(xx_a^{xd_a(t+T)}(t+T)) > f(xy_a^{xd_a(t)}(t)) \\ xx_a^{xd_a(t+T)}(t+T) & \text{else} \end{cases} \quad (3)$$

$\forall j \in [1, xd_a(t)]$

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

$$x\tilde{d}_a(t+1) = \begin{cases} x\tilde{d}_a(t) & \text{if } f(xx_a^{xd_a(t+1)}(t+1)) > f(xy_a^{x\tilde{d}_a(t)}(t)) \\ xd_a(t+1) & \text{else} \end{cases} \quad (4)$$

Recall that $gbest(d)$ is the index of the global best particle at dimension d then

$$x\hat{y}^{dbest}(t) = xy_{gbest(dbest)}^{dbest}(t) = \min(xy_1^{dbest}(t), \dots, xy_S^{dbest}(t)).$$

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 ($t+1^{\text{st}}$) dimension, $xd_a(t+1)$. The positional update is performed for each dimension component, $j \in [1, xd_a(t)]$ as follows:

$$\begin{aligned} vx_{a,j}^{xd_a(t)}(t+1) &= w(t)vx_{a,j}^{xd_a(t)}(t) + \\ &c_1r_{1,j}(t)(xy_{a,j}^{xd_a(t)}(t) - xx_{a,j}^{xd_a(t)}(t)) + \\ &c_2r_{2,j}(t)(x\hat{y}_j^{xd_a(t)}(t) - xx_{a,j}^{xd_a(t)}(t)) \\ xx_{a,j}^{xd_a(t)}(t+1) &= xx_{a,j}^{xd_a(t)}(t) + vx_{a,j}^{xd_a(t)}(t+1) \end{aligned} \quad (5)$$

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:

$$vd_a(t+1) = \left\lfloor \begin{aligned} &vd_a(t) + c_1r_1(t)(x\tilde{d}_a(t) - xd_a(t)) \\ &+ c_2r_2(t)(dbest - xd_a(t)) \end{aligned} \right\rfloor \quad (6)$$

$$xd_a(t+1) = xd_a(t) + vd_a(t+1)$$

where $\lfloor \cdot \rfloor$ 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 $|vd_{a,j}(t+1)| < VD_{\max}$ and the initial dimension range set by the user, $D_{\min} \leq xd_a(t) \leq D_{\max}$.

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

Due to space limitations the pseudo-code of the MD PSO technique is skipped in this paper.

B. FGBF Algorithm over MD PSO

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 I can then be plugged in between steps 3.3 and 3.4 of $bPSO$'s pseudo-code.

FGBF in $bPSO$ ($\xi, f(a, j)$)	
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 $\tilde{y}(t) = y_{aGB}(t)$

Table I: Pseudo-code of FGBF in $bPSO$

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.

FGBF has its generalized form when applied with 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 $\tilde{xd}_a(t)$). 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. $\forall d \in [D_{\min}, D_{\max}]$). Due to space limitations the pseudo-code of the FGBF over MD PSO is skipped.

IV. THE PROPOSED DYNAMIC CLUSTERING

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, $\xi = \{x_1, \dots, x_a, \dots, x_S\}$, has the positional component formed as,

$xx_a^{xd_a(t)}(t) = \{c_{a,1}, \dots, c_{a,j}, \dots, c_{a,xd_a(t)}\} \Rightarrow xx_{a,j}^{xd_a(t)}(t) = c_{a,j}$ 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

$D_{\min} \leq xd_a(t) \leq D_{\max}$, 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,

$f(xx_a^{xd_a(t)}, Z) = Q_e(xx_a^{xd_a(t)})(xd_a(t))^\alpha$ where

$$Q_e(xx_a^{xd_a(t)}) = \frac{1}{xd_a(t)} \sum_{j=1}^{xd_a(t)} \frac{\sum_{\forall z_p \in xx_{a,j}^{xd_a(t)}} \|xx_{a,j}^{xd_a(t)} - z_p\|}{\|xx_a^{xd_a(t)}\|} \quad (7)$$

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.

$$C_j \neq \{\emptyset\}, \forall j \in [1, xd_a(t)]$$

3. Two clusters should have no common data points, i.e.

$$C_i \cap C_j = \{\emptyset\}, i \neq j \text{ and } \forall i, j \in [1, xd_a(t)]$$

In order to satisfy the 1st and 3rd (hard) clustering constraints, before computing the clustering fitness score via the validity index function in (7), all data points are first assigned to the *closest* centroid. Yet there is no guarantee for the fulfillment of the 2nd constraint since $xx_{a,j}^{xd_a(t)}(t)$ 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 $\{xx_{a,j}^{xd_a(t)}\} = \{\emptyset\}$ for any j .

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 is not straightforward since in clustering, 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 2

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.

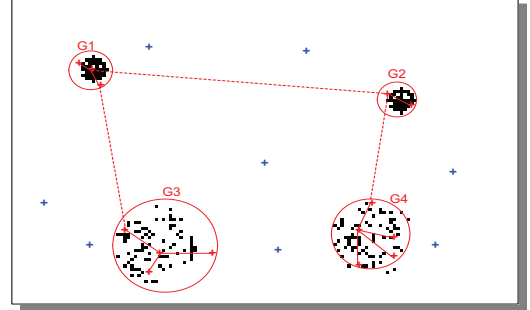


Figure 2: The formation of the centroid subset in a sample clustering example. The black dots represent data points over 2D space and each colored ‘+’ represents one centroid (dimension) of a swarm particle.

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) [13] 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 (7), $f(a, j)$ can simply be set as the j^{th} term of the summation in Q_e expression, such as,

$$f(a, j) = \frac{\sum_{\forall z_p \in xx_{a,j}^{xd_a(t)}} \|xx_{a,j}^{xd_a(t)} - z_p\|}{\|xx_a^{xd_a(t)}\|} \quad (8)$$

In Figure 2, 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

In order to test the application of the proposed techniques over data clustering, we create 11 synthetic data spaces as shown in Figure 3. 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. 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 \mathcal{E}_C value for all clustering schemes. The positional range, $\pm x_{\max}$, can be set simply as the natural boundaries of the 2D data space and dimensional range values $[D_{\min}, D_{\max}]$ are set as 2 and 80 respectively. The positional/dimensional velocity limits are empirically set as $V_{\max} = x_{\max}/2$ and $VD_{\max} = 20$.

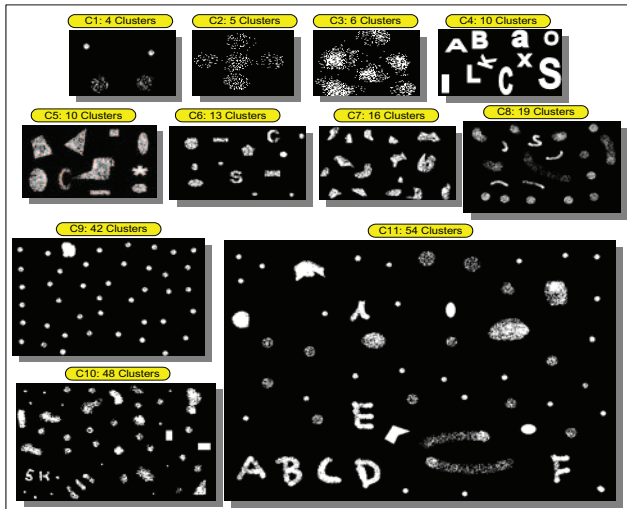


Figure 3: 2D synthetic data spaces carrying different clustering schemes.

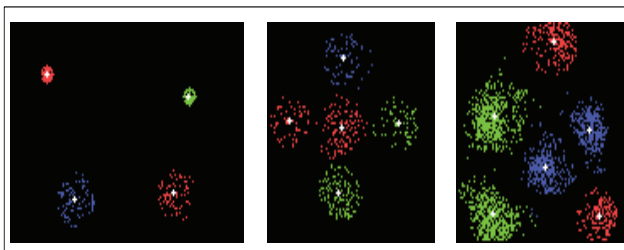


Figure 4: The standalone MD PSO (and $bPSO$) clustering for data spaces C1-3 shown in Figure 3.

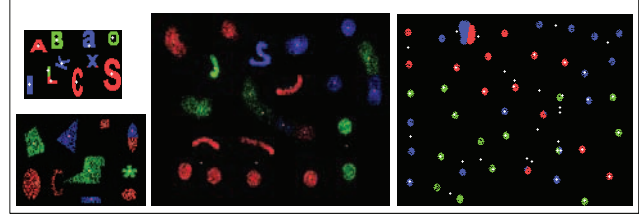


Figure 5: Erroneous $bPSO$ clustering over data spaces C4, C5, C6 and C9 shown in Figure 3.

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 3 are shown in Figure 4 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 g_{best} particle) is shown with a white '+'.

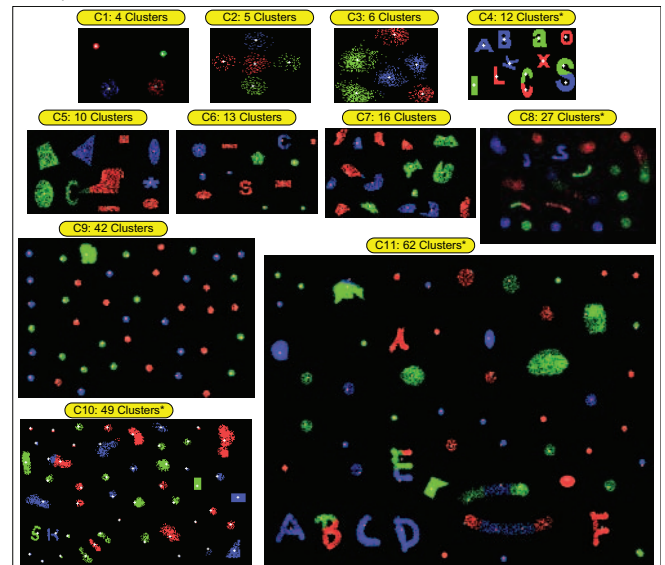


Figure 6: Typical clustering results via MD PSO with FGBF. Over-clustered samples are indicated with *.

The accuracy of both $bPSO$ and standalone MD PSO tends to degrade with the increasing dimensionality and complexity. Figure 5 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 clusters or more, serious 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.

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 6, 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 Eq. (7) as the fitness function and for some complex clustering schemes it may, therefore, yields its minimum score at a slightly higher number of clusters.

VI. CONCLUSIONS

In this paper, we first proposed two novel and generic 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. 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. Yet the convergence performance of MD PSO is still limited to the same level as *bPSO*, which suffers from the lack of diversity among 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” than 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.

A novel dynamic clustering technique based on MD PSO with FGBF is then proposed and tested over a data set that has 11 synthetic data spaces in 2D, each of which contains *ground-truth* clusters with significantly varying cluster properties and complexity levels. This data set allows us to test the clustering performance of the proposed technique and make comparative evaluations with the *bPSO* clustering, which has already been shown to be a superior technique than several well-known methods (e.g. K-means, FCM, Max-Min, etc.). 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*. An expected outcome and also evident from the experiments is that the speed and accuracy performances of *bPSO* drastically degrade with increasing complexity and this is also true for the standalone MD PSO (without FGBF). Furthermore the experimental results show that only such cooperation can indeed 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 such results indeed correspond to the global minimum of the validity index function used. As a result, the *true* number of clusters and accurate localization of centroids (and hence clustering) are achieved at the expense of increased computational complexity in per iteration, mainly due to the usage of MST.

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] T. Back and H.P. Schwefel, “An overview of evolutionary algorithm for parameter optimization”, *Evolution. Comput.*, 1, pp. 1–23, 1993.
- [3] 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.
- [4] R. Balasubramanian and J. Allebach, “A new approach to palette selection for color images”, *Imaging Technology*, Vol. 17, pp. 284-290, Dec. 1991.
- [5] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, John Wiley & Sons, 2005.
- [6] U.M. Fayyad, G.P. Shapire, P. Smyth and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*, MIT Press, Cambridge, MA, 1996.
- [7] H. Frigui, R. Krishnapuram, “Clustering by competitive agglomeration”, *Pattern Recognition*, vol. 30, pp. 1109-1119, 1997.
- [8] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, pp. 1-25. MA, 1989.
- [9] M. Halkidi, Y. Batistakis, M. Vazirgiannis, “On Cluster Validation Techniques”, *Journal of Intelligent Information Systems*, vol. 17 no. 2, 3. pp. 107-145, 2001.
- [10] A. K. Jain, M.N. Murthy and P.J. Flynn, “Data Clustering: A Review”, *ACM Computing Reviews*, Nov 1999.
- [11] J. Kennedy, R. Eberhart., “Particle swarm optimization”, in *Proc. of IEEE Int. Conf. On Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, 1995.
- [12] J. Koza, *Genetic Programming: On the Programming of Computers by means of Natural Selection*, MIT Press, Cambridge, Massachusetts, 1992.
- [13] J. R. Kruskal, “On the shortest spanning subtree of a graph and the traveling salesman problem”, *Proc. of AMS*, 71, 1956.
- [14] 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.
- [15] 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.
- [16] M. G. Omran, A. Salman, and A.P. Engelbrecht, *Particle Swarm Optimization for Pattern Recognition and Image Processing*, Springer Berlin, 2006.
- [17] J. Riget and J. S. Vesterstrom, “A Diversity-Guided Particle Swarm Optimizer - The ARPSO”, Technical report, Department of Computer Science, University of Aarhus, 2002.
- [18] 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.
- [19] P. Scheunders, “A comparison of clustering algorithms applied to color image quantization,” *Pattern Recog. Lett.*, vol. 18, pp. 1379–1384, 1997.
- [20] Y. Shi and R.C. Eberhart, “A Modified Particle Swarm Optimizer”, In *Proc. of the IEEE Congress on Evolutionary Computation*, pp. 69-73, 1998.
- [21] J.T. Tou and R.C. Gonzalez, *Pattern Recognition Principles*, London, Addison-Wesley, 1974.
- [22] F. Van den Bergh, “An Analysis of Particle Swarm Optimizers”, PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.
- [23] 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.