

Stochastic approximation driven particle swarm optimization with simultaneous perturbation – *Who will guide the guide?*

Serkan Kiranyaz^{a,*}, Turker Ince^b, Moncef Gabbouj^a

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

^b Izmir University of Economics, Department of Electronics and Telecommunications Engineering, TR-35330, Izmir, Turkey

ARTICLE INFO

Article history:

Received 19 February 2010

Received in revised form 10 May 2010

Accepted 12 July 2010

Available online 12 August 2010

Keywords:

Particle swarm optimization

Stochastic approximation

Multi-dimensional search

Gradient descent

ABSTRACT

The need for solving multi-modal optimization problems in high dimensions is pervasive in many practical applications. Particle swarm optimization (PSO) is attracting an ever-growing attention and more than ever it has found many application areas for many challenging optimization problems. It is, however, a known fact that PSO has a severe drawback in the update of its global best (*gbest*) particle, which has a crucial role of guiding the rest of the swarm. In this paper, we propose two efficient solutions to remedy this problem using a stochastic approximation (SA) technique. In the first approach, *gbest* is updated (moved) with respect to a global estimation of the gradient of the underlying (error) surface or function and hence can avoid getting trapped into a local optimum. The second approach is based on the formation of an alternative or artificial global best particle, the so-called *aGB*, which can replace the native *gbest* particle for a better guidance, the decision of which is held by a fair competition between the two. For this purpose we use simultaneous perturbation stochastic approximation (SPSA) for its low cost. Since SPSA is applied only to the *gbest* (not to the entire swarm), both approaches result thus in a negligible overhead cost for the entire PSO process. Both approaches are shown to significantly improve the performance of PSO over a wide range of non-linear functions, especially if SPSA parameters are well selected to fit the problem at hand. A major finding of the paper is that even if the SPSA parameters are not tuned well, results of SA-driven (SAD) PSO are still better than the best of PSO and SPSA. Since the problem of poor *gbest* update persists in the recently proposed extension of PSO, called multi-dimensional PSO (MD-PSO), both approaches are also integrated into MD-PSO and tested over a set of unsupervised data clustering applications. As in the basic PSO application, experimental results show that the proposed approaches significantly improved the quality of the MD-PSO clustering as measured by a validity index function. Furthermore, the proposed approaches are generic as they can be used with other PSO variants and applicable to a wide range of problems.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

The Merriam Webster dictionary defines optimization as *the mathematical procedures (as finding the maximum of a function) involved in this*. More specifically, consider the problem of finding a root θ^* (either minimum or maximum point) of the gradient equation: $g(\theta) \equiv \partial L(\theta)/\partial \theta = 0$ for some differentiable function $L: R^p \rightarrow R^1$. When g is present and L is a differentiable and uni-modal function, there are powerful deterministic methods for finding the global θ^* such as traditional steepest descent and Newton-Raphson methods. However, in many real problems g cannot be observed directly and/or L is multi-modal, in which case the aforementioned approaches may be trapped into some deceiving local optima. This brought the era of the stochastic optimization algorithms, which

can estimate the gradient and may avoid being trapped into a local optimum due to their stochastic nature. One of the most popular stochastic optimization techniques is stochastic approximation (SA), in particular the form that is called “gradient free” SA. Among many SA variants proposed by several researchers such as Styblinski and Tang [37], Kushner [25], Gelfand and Mitter [13], and Chin [6], the one and somewhat different SA application is called simultaneous perturbation SA (SPSA) proposed by Spall in 1992 [35]. The main advantage of SPSA is that it often achieves a much more economical operation in terms of loss function evaluations, which are usually the most computationally intensive part of an optimization process.

Particle swarm optimization (PSO) was introduced by Kennedy and Eberhart [20] 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 [42], which basically show a natural behavior when they search for some target (e.g. food). Henceforth, PSO exhibits

* Corresponding author.

E-mail address: serkan@cs.tut.fi (S. Kiranyaz).

certain similarities with the other evolutionary algorithms (EAs) [4] such as genetic algorithm (GA) [14], genetic programming (GP) [24], evolution strategies (ES) [5], and evolutionary programming (EP) [11]. The common point of all is that EAs are population based methods and they may avoid being trapped in a local optimum; 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 (or solution) space. The 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. Each particle keeps track of its position in the search space and its best solution so far achieved. This is the personal best position (the so-called *pbest* in [20]) and the PSO process also keeps track of the global best (GB) solution so far achieved by the swarm with its particle index (the so-called *gbest* in [20]). So during their journey with discrete time iterations, the velocity of each particle in the next iteration is computed by using the best position of the swarm (personal best position of the particle *gbest* as the *social* component), its personal best position (*pbest* as the *cognitive* component), and its current velocity (the *memory* term). Both *social* and *cognitive* components contribute randomly to the position of the particle 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 [26]. 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 likelihood of being trapped in local optima [32] and it is the main cause of the premature convergence problem especially when the search space is of high dimensions [40] and the problem to be optimized is multi-modal [32]. Therefore, at any iteration of a PSO process, *gbest* is the most important particle; however, it has the poorest update equation, i.e. when a particle becomes *gbest*, it resides on its personal best position (*pbest*) and thus both *social* and *cognitive* components are nullified in the velocity update equation. Although it guides the swarm during the following iterations, ironically it lacks the necessary guidance to do so effectively. In that, if *gbest* is (likely to get) trapped in a local optimum, so is the rest of the swarm due to the aforementioned direct link of information flow. This deficiency has been raised in a recent work [22] where an artificial GB particle, the *aGB*, is created at each iteration as an alternative to *gbest*, and replaces the native *gbest* particle as long as it achieves a better fitness score. In that study, it has been shown that such an enhanced *guidance* alone is indeed sufficient in most cases to achieve global convergence performance on multi-modal functions and even in high dimensions. However, the underlying mechanism for creating the *aGB* particle, the so-called fractional GB formation (FGBF), is not generic in the sense that it is rather problem dependent, which requires (the estimate of) individual dimensional fitness scores. This may be quite hard or infeasible for certain problems.

In order to address this drawback efficiently, in this paper we shall propose two approaches. The first one moves *gbest* efficiently or simply put, *guides* it with respect to the function (or error surface). The idea behind this is quite simple: since the velocity update equation of *gbest* is quite poor, SPSA as a simple yet powerful search technique is used to *drive* it instead. Due to its stochastic nature the likelihood of getting trapped into a local optimum is further decreased and with the SA, *gbest* is driven according to (an approximation of) the gradient of the function. The second approach has

a similar idea with the FGBF proposed in [22], i.e. an *aGB* particle is created by SPSA this time, which is applied over the personal best (*pbest*) position of the *gbest* particle. The *aGB* particle will then guide the swarm instead of *gbest* if and only if it achieves a better fitness score than the (personal best position of) *gbest*. Note that both approaches only deal with the *gbest* particle and hence the internal PSO process remains as is. That is, neither of the proposed approaches is a PSO variant by itself; rather a solution for the problem of the original PSO caused by poor *gbest* update. Furthermore, we shall demonstrate that the proposed approaches have a negligible computational cost overhead, e.g. only few percent increase of the computational complexity, which can be easily compensated with a slight reduction either in the swarm size or in the iteration number. Both approaches of SA-driven PSO (SAD PSO) will be tested and evaluated against the basic PSO (*bPSO*) over several benchmark uni- and multi-modal functions in high dimensions. Moreover, they are also applied to the multi-dimensional extension of PSO, the MD-PSO technique proposed in [22], which can find the optimum dimension of the solution space and hence voids the need of fixing the dimension of the solution space in advance. SAD MD-PSO is then tested and evaluated against the standalone MD-PSO application over several data clustering problems where both complexity and the dimension of the solution space (the true number of clusters) are varied significantly.

The rest of the paper is organized as follows. Section 2 surveys the basic PSO (*bPSO*), MD-PSO methods with the related work in data clustering. The proposed techniques applied over both PSO and MD-PSO are presented in Section 3. Section 4 presents the experimental results over two problem domains, non-linear function minimization and data clustering. Finally, Section 5 concludes the paper.

2. Related work

2.1. The basic PSO technique

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 with S particles, $\xi = \{x_1, \dots, x_a, \dots, x_S\}$, 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,

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

Since *gbest* is the index of the GB particle, then $\hat{y}(t) = y_{gbest}(t) = \arg \min_{v_i \in [1, S]} (f(y_i(t)))$. Then for each iteration in a PSO process, positional updates are performed for each particle, $a \in [1, S]$ and along

Table 1
Pseudo-code for *bPSO* algorithm.

```

bPSO (termination criteria: {IterNo,  $\varepsilon_C$ , ...,  $V_{\max}$ })
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  $\forall t \in [1, \text{IterNo}]$  do:
  3.1. For  $\forall a \in [1, S]$  do:
    3.1.1. Compute  $y_a(t)$  using Eq. (1)
    3.1.2. If  $\left( f(y_a(t)) < \min \left( f(\hat{y}(t-1)), f(y_i(t)) \right) \right)_{1 \leq i < a}$  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  $|v_{a,j}(t+1)| > V_{\max}$  then clamp it to  $|v_{a,j}(t+1)| = V_{\max}$ 
      3.4.1.3. Compute  $x_{a,j}(t+1)$  using Eq. (2)
    3.4.2. End For.
  3.5. End For.
4. End For.

```

each dimensional component, $j \in [1, N]$, as follows:

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

where w is the inertia weight [34], and c_1, c_2 are 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 the 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 [34], into the velocity update equation, it is widely accepted as the basic form of the 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 1.

Velocity clamping, also called "dampening" with a 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 [9]. Such oscillations are indeed crucial since they broaden the search capability of the swarm; however, they have a potential drawback of oscillating continuously around the optimum point. Therefore, such oscillations should be dampened and convergence is achieved with the proper use of velocity clamping and the inertia weight. Furthermore, this is the *bPSO* algorithm where the particle *gbest* is determined within the entire swarm. Another major topological approach, the so-called *lbest*, also exists where the swarm is divided into overlapping neighborhoods of particles and instead of defining *gbest* and $\hat{y}(t) = y_{gbest}(t)$ over the entire swarm, for a particular neighborhood N_i , the (local) best particle is referred as *lbest* with the position $\hat{y}_i(t) = y_{lbest}(t)$. Neighbors can be defined with respect to particle indices (i.e. $i \in \{j-l, j+l\}$) or by using some other topological forms [38]. It is obvious that *gbest* is a special

case of *lbest* scheme where the neighborhood is defined as the entire swarm. The *lbest* approach is one of the earlier attempts, which usually improves the diversity; however, it is slower than the *gbest* approach [21] and requires more parameters and setting of a suitable neighborhood topology. Even in [15], *gbest* approach is found to be superior than several *lbest* topologies and therefore is preferred in this context.

2.2. MD-PSO algorithm

Instead of operating at a fixed dimension N , the MD-PSO algorithm [22] 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 velocity 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
- $x\hat{y}_j^d(t)$: 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
- $x\vec{d}_a(t)$: personal best dimension component of particle a

Fig. 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 ($x\vec{d}_a(t)=3$). Therefore, at time t , a positional PSO update is first performed over the positional elements, $xx_{a,j}^2(t)$ and then the particle may move to another dimension by the dimensional PSO.

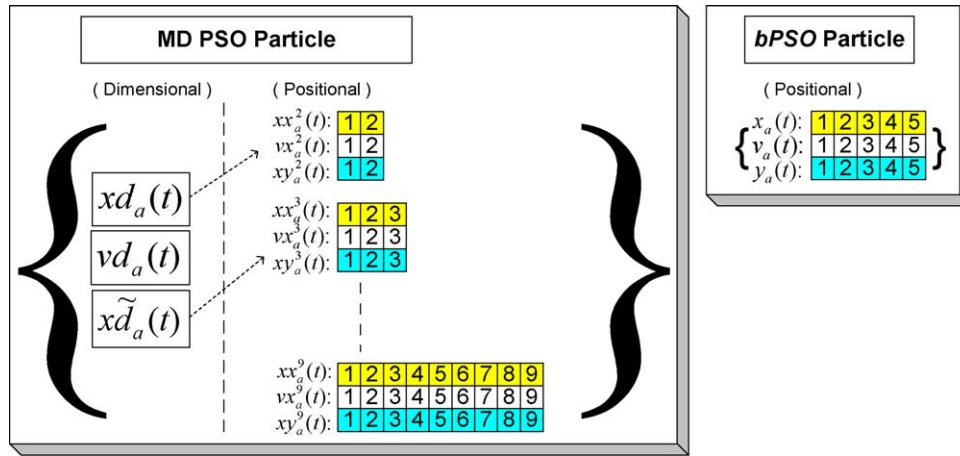


Fig. 1. Sample MD-PSO (right) vs. bPSO (left) particle structures. For MD-PSO [$D_{min}=2, D_{max}=9$] and at the current time $t, x_{d_a}(t)=2$ and $\tilde{x}_{d_a}(t)=3$. For bPSO $N=5$.

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. $x_{d_a}(t)=x_{d_a}(t+T)$), then the personal best position can be updated in iteration $t+T$ as follows:

$$xy_{a,j}^{x_{d_a}(t+T)}(t+T) = \begin{cases} xy_{a,j}^{x_{d_a}(t)}(t) & \text{if } f(xy_{a,j}^{x_{d_a}(t+T)}(t+T)) > f(xy_{a,j}^{x_{d_a}(t)}(t)) \\ x_{a,j}^{x_{d_a}(t+T)}(t+T) & \text{else} \end{cases} \quad \forall j \in [1, x_{d_a}(t)] \quad (3)$$

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

$$\tilde{x}_{d_a}(t+1) = \begin{cases} \tilde{x}_{d_a}(t) & \text{if } f(xy_{a,j}^{x_{d_a}(t+1)}(t+1)) > f(xy_{a,j}^{\tilde{x}_{d_a}(t)}(t)) \\ x_{d_a}(t+1) & \text{else} \end{cases} \quad (4)$$

Recall that $gbest(d)$ is the index of the global best particle at dimension d then $x_{j}^{\tilde{y}^{dbest}}(t) = xy_{j}^{dbest}(t) = arg \min_{v_i \in [1, S]} (f(xy_i^{dbest}(t)))$. For a particular iteration t , and for a particle $a \in [1, S]$, first the positional components are updated in the current dimension, $x_{d_a}(t)$, and then the dimensional update is performed to determine the next ($t+1$ st) dimension, $x_{d_a}(t+1)$. The positional update is performed for each dimension component, $j \in [1, x_{d_a}(t)]$ as follows:

$$vx_{a,j}^{x_{d_a}(t)}(t+1) = w(t)vx_{a,j}^{x_{d_a}(t)}(t) + c_1r_{1,j}(t)(xy_{a,j}^{x_{d_a}(t)}(t) - vx_{a,j}^{x_{d_a}(t)}(t)) + c_2r_{2,j}(t)(xy_j^{x_{d_a}(t)}(t) - vx_{a,j}^{x_{d_a}(t)}(t)) \quad (5)$$

$$xx_{a,j}^{x_{d_a}(t)}(t+1) = xx_{a,j}^{x_{d_a}(t)}(t) + vx_{a,j}^{x_{d_a}(t)}(t+1)$$

Note that the particle's new position, $xx_{a,j}^{x_{d_a}(t)}(t+1)$, will still be in the same dimension, $x_{d_a}(t)$; however, the particle may fly to another dimension afterwards with the following dimensional update equations:

$$vd_a(t+1) = \lfloor vd_a(t) + c_1r_1(t)(\tilde{x}_{d_a}(t) - x_{d_a}(t)) + c_2r_2(t)(dbest - x_{d_a}(t)) \rfloor \quad (6)$$

$$x_{d_a}(t+1) = x_{d_a}(t) + vd_a(t+1)$$

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, $D_{min} \leq x_{d_a}(t) \leq D_{max}$. Once the MD-PSO process terminates, the optimum solution will be $x_{j}^{\tilde{y}^{dbest}}$ at the optimum dimension, $dbest$, achieved by the particle $gbest(dbest)$ and finally the best (fitness) score achieved will naturally be $f(x_{j}^{\tilde{y}^{dbest}})$. Accordingly, the general pseudo-code of the MD-PSO technique is given in Table 2. More detailed information of MD-PSO can be found in [22].

2.3. Data clustering by MD-PSO

As the process of identifying natural groupings in a multi-dimensional data based on some distance metric (e.g. Euclidean), data clustering can be divided into two main categories: hierarchical and partitional [12]. Each category then has a wealth of sub-categories and different algorithmic approaches for finding the clusters where extensive survey can be found in [19,31].

A hard clustering technique based on the basic PSO (bPSO) was first introduced by Omran et al. in [29] and this work showed that the bPSO can outperform K-means, Fuzzy C-means (FCM), K-harmonic means (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 from 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.

Table 2
Pseudo-code of MD-PSO algorithm.

```

MD-PSO (termination criteria: {IterNo, εc, ...}, Vmax, VDmax, Dmin, Dmax)
1. For ∀a ∈ [1, S] do:
  1.1. Randomize xda(0)
  1.2. Initialize xda(0) = xda(0)
  1.3. For ∀d ∈ [Dmin, Dmax] do:
    1.3.1. Randomize xxad(0)
    1.3.2. Initialize xyad(0) = xxad(0)
  1.4. End For.
2. End For.
3. For ∀t ∈ [1, IterNo] do:
  3.1. For ∀a ∈ [1, S] do:
    3.1.1. If ( f(xxaxda(t)(t)) < min ( f(xyaxda(t)(t-1)), minp ∈ S-[a] ( f(xxpxda(t)(t)) ) ) )
      then do:
        3.1.1.1. xyaxda(t)(t) = xxaxda(t)(t)
        3.1.1.2. If ( f(xxaxda(t)(t)) < f(xygbest(xda(t))(t-1)) ) then gbest(xda(t)) = a
        3.1.1.3. If ( f(xxaxda(t)(t)) < f(xyadbest(t-1)) ) then xda(t) = xda(t)
        3.1.1.4. If ( f(xxaxda(t)(t)) < f(xyadbest(t-1)) ) then dbest = xda(t)
        3.1.2. End If.
      3.2. End For.
    3.3. If the termination criteria are met, then Stop.
    3.4. For ∀a ∈ [1, S] do:
      3.4.1. For ∀j ∈ [1, xda(t)] do:
        3.4.1.1. Compute vxa,jxda(t)(t+1) using Eq. (5)
        3.4.1.2. If ( |vxa,jxda(t)(t+1)| > Vmax ) then clamp it to |vxa,jxda(t)(t+1)| = Vmax
        3.4.1.3. Compute xxa,jxda(t)(t+1) using Eq. (5)
        3.4.2. End For.
        3.4.3. Compute vda(t+1) using Eq. (6)
        3.4.4. If ( |vda(t+1)| > VDmax ) then clamp it to |vda(t+1)| = VDmax
        3.4.5. Compute xda(t+1) using Eq. (6)
        3.4.6. If ( xda(t+1) < Dmin ) then ( xda(t+1) = Dmin )
        3.4.7. If ( xda(t+1) > Dmax ) then ( xda(t+1) = Dmax )
      3.5. End For.
    4. End For.
  
```

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. in [29] used the following validity index in their work,

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

where $Q_e(x_a) = \frac{1}{K} \sum_{j=1}^K \frac{\sum_{z_p \in x_{a,j}} \|x_{a,j} - z_p\|}{\|x_{a,j}\|}$ (7)

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; however, this makes the approach strictly parameter dependent. Another traditional and well-known validity index is Dunn's index [8], which suffers from two drawbacks: It is computationally expensive and sensitive to noise [16]. Several variants of Dunn's index were proposed in [31] where robustness against noise is improved. There are many other validity indices, i.e. proposed by Turi [39], Davies and Bouldin [7], Halkidi et al. [16], etc. A throughout survey can be found in [16]. 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 [29], 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) must 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 is also true for dynamic clustering schemes, DCPSO [30] and MEPSO [1], both of which eventually presented results only in low dimensions and for simple data distributions.

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) where in a recent work [22] MD-PSO technique has been successfully used. At time t , the particle a in the swarm, $\xi = \{x_1, \dots, x_a, \dots, x_S\}$, has the positional component formed as, $xx_a^{xd_a(t)}(t) = \{c_{a,1}, \dots, c_{a,j}, \dots, c_{a,x_d_a(t)}\} \Rightarrow xx_a^{xd_a(t)}(t) = c_{a,j}$ meaning that it represents a potential solution (i.e. the cluster centroids) for the $x_d_a(t)$ 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 $D_{\min} \leq x_d_a(t) \leq D_{\max}$, 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 $x_d_a(t)$ are initialized randomly within the range (i.e. refer to step 1.3.1 in MD-PSO pseudo-code in Table 2) and this further contributes to the overall diversity. In this work as well as in [22], the following validity index is used to obtain computational simplicity with minimal or no parameter dependency,

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

$$Q_e(xx_a^{xd_a(t)}) = \frac{1}{x_d_a(t)} \sum_{j=1}^{x_d_a(t)} \frac{\sum_{z_p \in xx_{a,j}^{xd_a(t)}} \|xx_{a,j}^{xd_a(t)} - z_p\|}{\|xx_{a,j}^{xd_a(t)}\|} \quad (8)$$

where Q_e is the quantization error (or the average intra-cluster distance) as the *Compactness* term and $(x_d_a(t))^\alpha$ 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 $C_j, \forall j \in [1, x_d_a(t)]$ should maintain the following constraints:

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

In order to satisfy the 1st and 3rd (hard) clustering constraints, before computing the clustering fitness score via the validity index function in (8), 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 .

Table 3
Pseudo-code for SPSA technique.

<p>SPSA (<i>IterNo</i>, <i>a</i>, <i>c</i>, <i>A</i>, α, γ)</p> <ol style="list-style-type: none"> 1. Initialize $\hat{\theta}_1$ 2. For $\forall k \in [1, \text{IterNo}]$ do: <ol style="list-style-type: none"> 2.1. Generate zero-mean, <i>p</i>-dimensional perturbation vector: Δ_k 2.2. Let $a_k = a/(A+k)^\alpha$ and $c_k = c/k^\gamma$ 2.3. Compute $L(\hat{\theta}_k + c_k \Delta_k)$ and $L(\hat{\theta}_k - c_k \Delta_k)$ 2.4. Compute $\hat{g}_k(\hat{\theta}_k)$ using (11) 2.5. Compute $\hat{\theta}_{k+1}$ using (10) 3. End For.
--

3. The proposed techniques: SAD PSO and SAD MD-PSO

3.1. SPSA overview

The goal of deterministic optimization methods is to minimize a loss function $L: R^p \rightarrow R^1$, which is a differentiable function of θ and the minimum (or maximum) point θ^* corresponds to the zero-gradient point, i.e.

$$g(\theta) \equiv \left. \frac{\partial L(\theta)}{\partial \theta} \right|_{\theta=\theta^*} = 0 \tag{9}$$

As mentioned earlier, in cases where more than one point satisfies this equation (e.g. a multi-modal problem), then such algorithms may only converge to a local minimum. Moreover, in many practical problems, g is not readily available. This makes the SA algorithms quite popular and they are in the general SA form:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k(\hat{\theta}_k) \tag{10}$$

where $\hat{g}_k(\hat{\theta}_k)$ is the estimate of the gradient $g(\theta)$ at iteration k and a_k is a scalar gain sequence satisfying certain conditions [35]. There are two common SA methods: finite difference SA (FDSA) and simultaneous perturbation SA (SPSA). FDSA adopts the traditional Kiefer-Wolfowitz approach to approximate gradient vectors as a vector of p partial derivatives where p is the dimension of the loss function. On the other hand, SPSA has all elements of $\hat{\theta}_k$ perturbed simultaneously using only two measurements of the loss function as,

$$\hat{g}_k(\hat{\theta}_k) = \frac{L(\hat{\theta}_k + c_k \Delta_k) - L(\hat{\theta}_k - c_k \Delta_k)}{2c_k} \begin{bmatrix} \Delta_{k1}^{-1} \\ \Delta_{k2}^{-1} \\ \vdots \\ \Delta_{kp}^{-1} \end{bmatrix} \tag{11}$$

where the p -dimensional random variable Δ_k is usually chosen as *Bernoulli* ± 1 distribution and c_k is a scalar gain sequence satisfying certain conditions [35]. Spall in [35] presents conditions for convergence of SPSA (i.e. $\hat{\theta}_k \rightarrow \theta^*$) and show that under certain conditions both SPSA and FDSA have the same convergence ability – yet SPSA needs only 2 measurements whereas FDSA needs $2p$. This makes SPSA our natural choice for driving g_{best} in both approaches. Table 3 presents the general pseudo-code of the SPSA technique.

SPSA has five parameters as given in Table 3. Spall in [36] recommended to use values for A (the stability constant), α , and γ as 60, 0.602 and 0.101, respectively. However, he also concluded that “the choice of both gain sequences is critical to the performance of the SPSA as with all stochastic optimization algorithms and the choice of their respective algorithm coefficients”. This especially makes the choice of gain parameters a and c critical for a particular problem, i.e. Maryak and Chin in [27] varied them with respect to the problem whilst keeping the other three (A , α , and γ) as recommended.

Recently Maeda and Kuratani in [28] has used SPSA with the b PSO in a hybrid algorithm called Simultaneous Perturbation PSO

(SP-PSO) over a limited set of problems and reported some slight improvements over the b PSO. Both SP-PSO variants they proposed involved the insertion of the $\hat{g}_k(\hat{\theta}_k)$ directly over the velocity equations of all swarm particles with the intention of improving their local search capability. This may, however, present some drawbacks. First of all, performing SPSA at each iteration and for all particles will double the computational cost of the PSO since SPSA will require an additional function evaluation at each iteration.¹ Secondly, such an abrupt adding of SPSA’s $\hat{g}_k(\hat{\theta}_k)$ term directly into the b PSO may degrade the original PSO workout, i.e. the collective swarm updates and interactions, and require an accurate scaling between the parameters of the two methods, PSO’s and SPSA. Otherwise, one can dominant the other, and hence their combination may turn out to be a noisy variant of either method. This is perhaps the reason of the limited success, if any, achieved by SP-PSO. As we discuss next and demonstrate its elegant performance experimentally, SPSA should not be mixed up with PSO as such, rather should only be used to *guide* it if SPSA can drive the PSO’s native guide, the g_{best} , better than PSO.

3.2. SAD PSO

In this work two distinct SAD PSO approaches are proposed, each of which is only applied to g_{best} whilst keeping the internal PSO process intact. Since both SPSA and PSO are iterative processes, in both approaches SPSA can thus easily be integrated into PSO by using the same iteration count (i.e. $t \equiv k$). In other words, at a particular iteration t in the PSO process, only the SPSA steps 2.1–2.5 in Table 3 are inserted accordingly into the PSO process. The following sub-sections will detail each approach.

A1) First SAD PSO approach: g_{best} update by SPSA

In this approach, at each iteration g_{best} particle is updated using SPSA. This requires the adaptation of the SPSA elements (parameters and variables) and integration of the internal SPSA part (within the loop) appropriately into the PSO pseudo-code, as shown in Table 4. Note that such a “plug-in” approach will not change the internal PSO structure and only affects the g_{best} particle’s movement. It only costs two extra function evaluations and hence at each iteration the total number of evaluations is increased from S to $S+2$ (recall that S is the swarm size).

Since the fitness of each particle’s current position is computed within the PSO process, it is possible to further diminish this cost to only *one* extra fitness evaluation per iteration. Let $\hat{\theta}_k + c_k \Delta_k = x_a(t)$ in step 3.4.1.1. and thus $L(\hat{\theta}_k + c_k \Delta_k)$ is known *a priori*. Then naturally, $\hat{\theta}_k - c_k \Delta_k = x_a(t) - 2c_k \Delta_k$, which is the only (new) location where the (extra) fitness evaluation ($L(\hat{\theta}_k - c_k \Delta_k)$) has to be computed. Once the gradient ($\hat{g}_k(\hat{\theta}_k)$) is estimated in step 3.4.1.4, then the next (updated) location of the g_{best} will be: $x_a(t+1) = \hat{\theta}_{k+1}$. Note that the difference of this “low-cost” SPSA update is that $x_a(t+1)$ is updated (estimated) *not* from $x_a(t)$, but instead from $x_a(t) - c_k \Delta_k$.

This approach can easily be extended for MD-PSO, which is a natural extension of PSO for multi-dimensional search within a given (dimension) range, $d \in [D_{min}, D_{max}]$. The main difference is that in each dimension, there is a distinct g_{best} particle, $g_{best}(d)$. So SPSA is applied individually over the position of each $g_{best}(d)$ if it (re-) visits the dimension d (i.e. $d = x_{d,g_{best}(t)}$). Therefore, there

¹ In [28] the function evaluations are given with respect to the iteration number; however, it should have been noted that SP-PSO performs twice more evaluations than b PSO per iteration. Considering this fact, the plots therein show little or no performance improvement at all.

Table 4
Pseudo-code for the first SAD PSO approach.

<p>A1) SAD PSO Plug-in (termination criteria: $\{IterNo, \varepsilon_C, \dots, V_{max}, a, c, A, \alpha, \gamma\}$)</p> <ol style="list-style-type: none"> 1. See Line 1 in Table 1 2. See Line 2 in Table 1 3. For $\forall t \in [1, IterNo]$ do: 3.4. For $\forall a \in [1, S]$ do: <ol style="list-style-type: none"> 3.4.1. If $(a = gbest)$ then do: <ol style="list-style-type: none"> 3.4.1.1. Let $k = t, \hat{\theta}_k = x_a(t)$ and $L = f$ 3.4.1.2. Let $a_k = a/(A+k)^\alpha$ and $c_k = c/k^\gamma$ 3.4.1.3. Compute $L(\hat{\theta}_k + c_k \Delta_k)$ and $L(\hat{\theta}_k - c_k \Delta_k)$ 3.4.1.4. Compute $\hat{g}_k(\hat{\theta}_k)$ using Eq. (11) 3.4.1.5. Compute $x_a(t+1) = \hat{\theta}_{k+1}$ using Eq. (10) 3.4.2. Else do: <ol style="list-style-type: none"> 3.4.2.1. For $\forall j \in [1, N]$ do: <ol style="list-style-type: none"> 3.4.2.1.1. Compute $v_{a,j}(t+1)$ using Eq. (2) 3.4.2.1.2. If $(v_{a,j}(t+1) > V_{max})$ then clamp it to $v_{a,j}(t+1) = V_{max}$ 3.4.2.1.3. Compute $x_{a,j}(t+1)$ using Eq. (2) 3.4.2.2. End For. 3.5. End For. 4. End For.
--

can be $2(D_{max} - D_{min})$ number of function evaluations, indicating a significant cost especially if a wide dimensional range is used. However, this is a theoretical limit, which can only happen if $gbest(i) \neq gbest(j)$ for $\forall i, j \in [D_{min}, D_{max}], i \neq j$ and all particles altogether visit the particular dimensions in which they are $gbest$ (i.e. $xd_{gbest(d)}(t) = d, \forall t \in [1, iterNo]$). Especially in a wide dimensional range, note that this is highly unlikely due to the dimensional velocity, which makes particles move (jump) from one dimension to another at each iteration. It is straightforward to see that under the assumption of a uniform distribution for particles' movements over all dimensions within the dimensional range, SAD MD-PSO too, would have the same cost overhead as the SAD PSO. Experimental results indicate that the practical overhead cost is only slightly higher than this.

A2) Second SAD PSO approach: aGB formation by SPSA

The second approach replaces the FGFB operation proposed in [22] with the SPSA to create an aGB particle. SPSA is basically applied over the $pbest$ position of the $gbest$ particle. The aGB particle will then guide the swarm instead of $gbest$ if and only if it achieves a better fitness score than the (personal best position of) $gbest$. SAD PSO pseudo-code as given in Table 5 can then be plugged in between steps 3.3 and 3.4 of $bPSO$ pseudo-code.

The extension of the second approach to MD-PSO is also quite straightforward. In order to create an aGB particle, for all dimensions in the given range (i.e. $\forall d \in [D_{min}, D_{max}]$) SPSA is applied individually over the personal best position of each $gbest(d)$ particle and 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 $xx_{aGB}^d(t+1)$ and $xy_{aGB}^d(t)$. As a result, the SPSA cre-

Table 5
PSO Plug-in for the second approach.

<p>A2) SAD PSO Plug-in ($\xi, a, c, A, \alpha, \gamma$)</p> <ol style="list-style-type: none"> 1. Create a new aGB particle, $\{x_{aGB}(t+1), y_{aGB}(t+1)\}$ 2. Let $k = t, \hat{\theta}_k = \hat{y}(t)$ and $L = f$ 3. Let $a_k = a/(A+k)^\alpha$ and $c_k = c/k^\gamma$ 4. Compute $L(\hat{\theta}_k + c_k \Delta_k)$ and $L(\hat{\theta}_k - c_k \Delta_k)$ 5. Compute $\hat{g}_k(\hat{\theta}_k)$ using Eq. (11) 6. Compute $x_{aGB}(t) = \hat{\theta}_{k+1}$ using Eq. (10) 7. Compute $f(x_{aGB}(t+1)) = L(\hat{\theta}_{k+1})$ 8. If $(f(x_{aGB}(t+1)) < f(y_{aGB}(t)))$ then $y_{aGB}(t+1) = x_{aGB}(t+1)$ 9. Else $y_{aGB}(t+1) = y_{aGB}(t)$ 10. If $(f(y_{aGB}(t+1)) < f(\hat{y}(t)))$ then $\hat{y}(t) = y_{aGB}(t+1)$

Table 6
MD-PSO Plug-in for the second approach.

<p>A2) SAD MD-PSO Plug-in ($\xi, a, c, A, \alpha, \gamma$)</p> <ol style="list-style-type: none"> 1. Create a new aGB particle, $\{xx_{aGB}^d(t+1), xy_{aGB}^d(t+1)\}$ for $\forall d \in [D_{min}, D_{max}]$ 2. For $\forall d \in [D_{min}, D_{max}]$ do: <ol style="list-style-type: none"> 2.1. Let $k = t, \hat{\theta}_k = xy^d(t)$ and $L = f$ 2.2. Let $a_k = a/(A+k)^\alpha$ and $c_k = c/k^\gamma$ 2.3. Compute $L(\hat{\theta}_k + c_k \Delta_k)$ and $L(\hat{\theta}_k - c_k \Delta_k)$ 2.4. Compute $\hat{g}_k(\hat{\theta}_k)$ using Eq. (11) 2.5. Compute $xx_{aGB}^d(t+1) = \hat{\theta}_{k+1}$ using Eq. (10) 2.6. If $(f(xx_{aGB}^d(t+1)) < f(xy_{aGB}^d(t)))$ then $xy_{aGB}^d(t+1) = xx_{aGB}^d(t+1)$ 2.7. Else $xy_{aGB}^d(t+1) = xy_{aGB}^d(t)$ 2.8. If $(f(xy_{aGB}^d(t+1)) < f(xy_{gbest(d)}^d(t)))$ then $xy_{gbest(d)}^d(t) = xy_{aGB}^d(t+1)$ 3. End For. 4. Re-assign $dbest$: $dbest = \arg \min_{d \in [D_{min}, D_{max}]} (f(xy_{gbest(d)}^d(t)))$

ates one aGB particle providing (potential) GB solutions $(xy_{aGB}^d(t+1), \forall d \in [D_{min}, D_{max}])$ for all dimensions in the given dimension range. The pseudo-code of the second approach as given in Table 6 can then be plugged in between steps 3.2 and 3.3 of the MD-PSO pseudo-code, given in Table 2.

Note that in the second SAD PSO approach, there are three extra fitness evaluations (as opposed to two in the first one) at each iteration. Yet as in the first approach, it is possible to further decrease the cost of SAD PSO by one (from three to two fitness evaluations per iteration). Let $\hat{\theta}_k + c_k \Delta_k = \hat{y}(t)$ in step 2 and thus $L(\hat{\theta}_k + c_k \Delta_k)$ is known *a priori*. Then it follows the same analogy as before and the only difference is that the aGB particle is formed not from $\hat{\theta}_k = \hat{y}(t)$ but from $\hat{\theta}_k = \hat{y}(t) - c_k \Delta_k$. However, in this approach a major difference in the computational cost may occur since in each iteration there are inevitably $3(D_{max} - D_{min})$ (or $2(D_{max} - D_{min})$ for low-cost application) fitness evaluations, which can be significant.

4. Experimental results

Two problem domains are considered in this paper over which the proposed techniques are evaluated. The first one is non-linear function minimization where several benchmark functions are used. This allows us to test the performance of SAD PSO against $bPSO$ over both uni- and multi-modal functions. 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. This problem domain can efficiently validate the performance of SAD MD-PSO regarding the convergence to the global solution in the right dimension. In this way, we can truly evaluate the contribution and the significance of both approaches (A1 and A2) especially over multi-modal optimization problems in high dimensions.

4.1. Non-linear function minimization

We used seven benchmark functions given in Table 7 to provide a good mixture of complexity and modality. They have been widely studied by several researchers, e.g. see [3,10,17,26,33,34]. *Sphere*, *De Jong* and *Rosenbrock* are uni-modal functions and the rest are multi-modal, meaning that they have many 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.

Both approaches of the proposed SAD PSO along with the “low-cost” application are tested over seven benchmark functions given in Table 7 and compared with the $bPSO$ and standalone SPSA appli-

Table 7
Benchmark functions with dimensional bias.

Function	Formula	Initial range	Dimension set: {d}
Sphere	$F_1(x, d) = \left(\sum_{i=1}^d x_i^2 \right)$	[-150, 75]	20, 50, 80
De Jong	$F_2(x, d) = \left(\sum_{i=1}^d ix_i^4 \right)$	[-50, 25]	20, 50, 80
Rosenbrock	$F_3(x, d) = \left(\sum_{i=1}^d 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right)$	[-50, 25]	20, 50, 80
Rastrigin	$F_4(x, d) = \left(\sum_{i=1}^d 10 + x_i^2 - 10 \cos(2\pi x_i) \right)$	[-500, 250]	20, 50, 80
Griewank	$F_5(x, d) = \left(\frac{1}{4000} \sum_{i=1}^d x_i^2 - \prod_{i=1}^d \cos\left(\frac{x_i}{\sqrt{ i+1}}\right) \right)$	[-500, 250]	20, 50, 80
Schwefel	$F_6(x, d) = \left(418.9829d + \sum_{i=1}^d x_i \sin\left(\sqrt{ x_i }\right) \right)$	[-500, 250]	20, 50, 80
Giunta	$F_7(x, d) = \left(\sum_{i=1}^d \sin\left(\frac{16}{15}x_i - 1\right) + \sin^2\left(\frac{16}{15}x_i - 1\right) + \frac{1}{50} \sin\left(4\left(\frac{16}{15}x_i - 1\right)\right) + \frac{268}{1000} \right)$	[-500, 250]	20, 50, 80

cation. We used the same termination criteria as the combination of the maximum number of iterations allowed (*iterNo* = 10,000) and the cut-off error ($\epsilon_C = 10^{-5}$). We used three dimensions (20, 50 and 80) for the sample functions in order to test the performance of each technique in such varying dimensions individually. For PSO (*bPSO* and *SAD PSO*) we used the swarm size, *S* = 40, and *w* is linearly decreased from 0.9 to 0.2. We used the recommended values for *A*, α and γ as 60, 0.602 and 0.101 as well as set *a* and *c* to 1 that are *fixed* for all functions. We purposefully made no parameter tuning for *SPSA* since this may not be feasible for many practical applications, particularly the ones where the underlying fitness (error) surface is unknown. In order to make a fair comparison among *SPSA*, *bPSO* and *SAD PSO*, the number of evaluations is kept equal (so *S* = 38 and *S* = 37 are used for both *SAD PSO* approaches and the number of evaluations is set to $40 \times 10,000 = 4E+5$ for *SPSA*).

For each setting (for each function and each dimension), 100 runs are performed and the 1st and 2nd order statistics (mean,

μ and standard deviation, σ) of the fitness scores are reported in [Table 8](#) whilst the best statistics are highlighted. During each run, the operation terminates when the fitness score drops below the cut-off error and it is assumed that the global minimum of the function is reached, henceforth; the score is set to 0. Therefore, for any setting yielding $\mu = 0$ as the average score means that the method converges to the global minimum at every run.

As the entire statistics in the right side of [Table 8](#) indicate, either *SAD PSO* approach achieves an equal or superior average performance statistics over all functions regardless of the dimension, modality, and without any exception. In other words, *SAD PSO* performs equal or better than the best of *bPSO* and *SPSA* – even though either of them might have a quite poor performance for a particular function. Note especially that if *SPSA* performs well enough (meaning that the setting of the critical parameters, e.g. *a* and *c* is appropriate), then a significant performance improvement can be achieved by *SAD PSO*, i.e. see for instance De Jong, Rosenbrock and

Table 8
Statistical results from 100 runs over 7 benchmark functions.

Functions	d	SPSA		bPSO		SAD PSO (A2)		SAD PSO (A1)	
		μ	σ	μ	σ	μ	σ	μ	σ
Sphere	20	0	0	0	0	0	0	0	0
	50	0	0	0	0	0	0	0	0
	80	0	0	135.272	276.185	0	0	0	0
De Jong	20	0.013	0.0275	0	0	0	0	0	0
	50	0.0218	0.03	9.0445	26.9962	0.0075	0.0091	0.2189	0.6491
	80	0.418	0.267	998.737	832.1993	0.2584	0.4706	13,546.02	4305.04
Rosenbrock	20	1.14422	0.2692	1.26462	0.4382	1.29941	0.4658	0.4089	0.2130
	50	3.5942	0.7485	15.9053	5.21491	12.35141	2.67731	2.5472	0.3696
	80	5.3928	0.7961	170.9547	231.9113	28.1527	5.1699	5.2919	0.8177
Rastrigin	20	204.9169	51.2863	0.0429	0.0383	0.0383	0.0369	0.0326	0.0300
	50	513.3888	75.7015	0.0528	0.0688	0.0381	0.0436	0.0353	0.0503
	80	832.9218	102.1792	0.7943	0.9517	0.2363	0.6552	0.1240	0.1694
Griewank	20	0	0	0	0	0	0	0	0
	50	1.0631E+007	3.3726E+006	50.7317	191.1558	0	0	3074.02	13,989
	80	2.8251E+007	5.7896E+006	24,978	23,257	20,733	24,160	378,210	137,410
Schwefel	20	0.3584	0.0794	1.7474	0.3915	0.3076	0.0758	0.3991	0.0796
	50	0.8906	0.1006	10.2027	2.2145	0.8278	0.1093	0.9791	0.1232
	80	1.4352	0.1465	21.8269	5.1809	1.3633	0.1402	1.5528	0.1544
Giunta	20	42.743	667.2494	495.0777	245.1220	445.1360	264.1160	445.1356	249.5412
	50	10.724	1027.6	4257	713.1723	3938.9	626.9194	3916.2	758.3290
	80	17.283	1247.9	9873.6	1313	8838.2	1357	8454.2	1285.3

Table 9
Statistical results between full-cost and low-cost modes from 100 runs over 7 benchmark functions.

Functions	d	Full-cost mode				Low-cost mode			
		SAD PSO (A2)		SAD PSO (A1)		SAD PSO (A2)		SAD PSO (A1)	
		μ	σ	μ	σ	μ	σ	μ	σ
Sphere	20	0	0	0	0	0	0	0	0
	50	0	0	0	0	0	0	0	0
	80	0	0	0	0	0	0	0	0
De Jong	20	0	0	0	0	0	0	0	0
	50	0.0075	0.0091	0.2189	0.6491	0.0073	0.0036	23.7977	48.2012
	80	0.2584	0.4706	13,546.02	4305.04	0.0326	0.0290	14,136	4578.8
Rosenbrock	20	1.29941	0.4658	0.4089	0.2130	1.1412	0.4031	0.3124	0.2035
	50	12.35141	2.67731	2.5472	0.3696	9.2063	2.2657	2.5864	0.8232
	80	28.1527	5.1699	5.2919	0.8177	24.0142	4.9823	12.9923	2.8497
Rastrigin	20	0.0383	0.0369	0.0326	0.0300	0.0263	0.0634	0.0383	0.0327
	50	0.0381	0.0436	0.0353	0.0503	0.0066	0.0083	0.0062	0.0082
	80	0.2363	0.6552	0.1240	0.1694	0.0053	0.0086	0.0043	0.0065
Griewank	20	0	0	0	0	0	0	0	0
	50	0	0	3074.02	13,989	0.0018	0.0125	3033.1	16,588
	80	20,733	24,160	378,210	137,410	143.5794	1247.7	342,230	114,280
Schwefel	20	0.3076	0.0758	0.3991	0.0796	0.7538	0.2103	2.1327	0.5960
	50	0.8278	0.1093	0.9791	0.1232	1.2744	0.2282	6.4797	1.0569
	80	1.3633	0.1402	1.5528	0.1544	1.6965	0.2165	9.5252	1.6241
Giunta	20	445.1360	264.1160	445.1356	249.5412	375.6510	242.7301	387.8901	249.8883
	50	3938.9	626.9194	3916.2	758.3290	3282	890.3426	3688.3	908.4803
	80	8838.2	1357	8454.2	1285.3	7922.1	1320.2	12,132	1579.4

Schwefel. On the other hand, if SPSA does not perform well, even much worse than any other technique, SAD PSO still outperforms *bPSO* to a certain degree, e.g. see Giunta and particularly Griewank for $d=50$ where SAD PSO can still converge to the global optimum ($\mu=0$) whereas SPSA performance is rather bad. This basically supports our aforementioned claim, i.e. the PSO update for *gbest* is so poor that even an under-performing SPSA implementation can still improve the overall performance significantly. Note that the opposite is also true, that is, SAD PSO, which internally runs SPSA for *gbest* achieves better performance than SPSA alone, even when PSO's performance is limited.

Based on the results in Table 8, we can perform comparative evaluations with some of the promising PSO variants such as [2,10,32,33] where similar experiments are performed over some or all of these benchmark functions. For example in [2], a tournament selection mechanism is formed among particles and the method is applied over four functions (*Sphere*, *Rosenbrock*, *Rastrigin*

and *Griewank*). Although the method is applied over a reduced positional range, ± 15 , and at low dimensions (10, 20 and 30), they got varying average scores in the range {0.3, 1194}. As a result, they reported both better and worse performances than *bPSO*, depending on the function. In [10], *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 [32], 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 experimental results have shown that none of the three methods converged to the global minimum except *ARPSO* over (only) Rastrigin at dimension 20. *ARPSO* performed better than *bPSO* and *GA* over Rastrigin

Table 10
t-Test results for statistical significance analysis.

Functions	d	Pair of competing methods			
		<i>bPSO</i> – (A2)	<i>bPSO</i> – (A1)	SPSA – (A2)	SPSA – (A1)
Sphere	20	0	0	0	0
	50	0	0	0	0
	80	4.90	4.90	0	0
De Jong	20	0	0	4.73	4.73
	50	3.35	3.27	4.56	3.03
	80	12.00	28.62	2.95	31.46
Rosenbrock	20	0.54	17.56	2.88	21.42
	50	6.06	25.55	31.50	12.54
	80	6.16	7.14	43.51	0.88 (*)
Rastrigin	20	0.86	2.12	39.95	39.95
	50	1.80	2.05	67.81	67.81
	80	4.83	6.93	81.49	81.50
Griewank	20	0	0	0	0
	50	2.65	2.16	31.52	31.51
	80	1.27 (*)	25.35	48.76	48.13
Schwefel	20	36.11	33.75	4.63	3.62
	50	42.28	41.59	4.23	5.56
	80	39.48	39.12	3.55	5.53
Giunta	20	1.39	1.43	589.42	593.75
	50	3.35	3.27	56.37	53.31
	80	5.48	7.73	45.81	49.28

and Rosenbrock but worse for Griewank. The CPSO proposed in [41] was applied over five functions including Sphere, Rastrigin, Rosenbrock and Griewank. The dimension of all functions is fixed to 30 and in this dimension, CPSO performed better than *b*PSO in 80% of the experiments. Finally in [33] 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 six 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 combinations converged to the global minimum of multi-modal functions except only for some dimensions of the Griewank function.

The statistical comparison between low-cost mode and the original (full-cost) are reported in Table 9. The statistics in the table indicate that both modes within both approaches usually obtain a similar performance but occasionally a significant gap is visible. For instance, low-cost mode achieves a significantly better performance within the second SAD PSO approach for De Jong and Griewank functions at $d=80$. The opposite is true for Schwefel particularly at $d=20$.

In order to verify if the results are statistically significant, we shall now apply statistical significance test between each SAD PSO approach and each technique (*b*PSO and SPSA) using the statistical data given in Table 9. Let H_0 be the null hypothesis, which states that there is no difference between the proposed and competing techniques (i.e. the statistical results occur by chance). We shall then define two common threshold values for P , 5% and 1%. If the P value, which is the probability of observing such a large difference (or larger) between the statistics, is less than either threshold, then we can reject H_0 with that confidence level. To accomplish this, we performed the standard t -test and compute the t values between the pair of competing methods. Recall that the formula for the t -test is as follows:

$$t = \frac{\mu_1 - \mu_2}{\sqrt{((n_1 - 1)\sigma_1^2 + (n_2 - 1)\sigma_2^2) / ((n_1 + n_2 - 2)((n_1 + n_2) / n_1 n_2)}} \tag{12}$$

where $n_1 = n_2 = 100$ is the number of runs. Using the first and second order statistics presented in Table 8, the overall t -test values are computed and enlisted in Table 10. In those entries with 0 value, both methods have a zero mean and zero variance, indicating convergence to the global optimum. In such cases, H_0 cannot be rejected. In those non-zero entries, t -test values corresponding to the best approach are highlighted. In those t -

Table 11
 t -Table presenting degrees of freedom vs. probability.

Degrees of freedom	P : probability			
	0.1	0.05	0.01	0.001
100	1.29	1.66	2.364	3.174
∞	1.282	1.645	2.325	3.090

tests the *degrees of freedom* is simply, $n_1 + n_2 - 2 = 198$. Table 11 presents two corresponding entries of t -test values required to reject H_0 at several levels of confidence (one-tailed test). Accordingly, H_0 can be rejected and hence all results are statistically significant beyond the confidence level of 0.01 except the two entries shown with a (*) in Table 10. Note that the majority of the results are statistically significant beyond the 0.001 level of confidence (e.g. the likelihood to occur by chance is less than 1 in 1000 times).

4.2. Data clustering

In order to test each approach of the proposed SAD MD-PSO technique over (data) clustering, we created 8 synthetic data spaces as shown in Fig. 2 where white dots (pixels) represent data points. 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 approaches 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 convergence accuracy to the true (solution space) dimension. As a result, significantly varying complexity levels are established among all data spaces to perform a general-purpose evaluation of each approach.

Unless stated otherwise, the maximum number of iterations is set to 10,000 as before; 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 4.1 are also used in all experiments except the positional range, since it can now be set simply as the natural boundaries of the 2D data space. For MD-PSO, we used the swarm size, $S=200$ and for both SAD MD-PSO approaches, a reduced number is used in order to ensure the same number of evaluation among all competing techniques. w is linearly decreased from 0.75 to 0.2 and we again used the recommended values for A , α and γ as 60, 0.602 and 0.101, whereas a and c are set to 0.4 and 10, respectively. For each dataset, 20 clustering runs are performed and the 1st and 2nd order

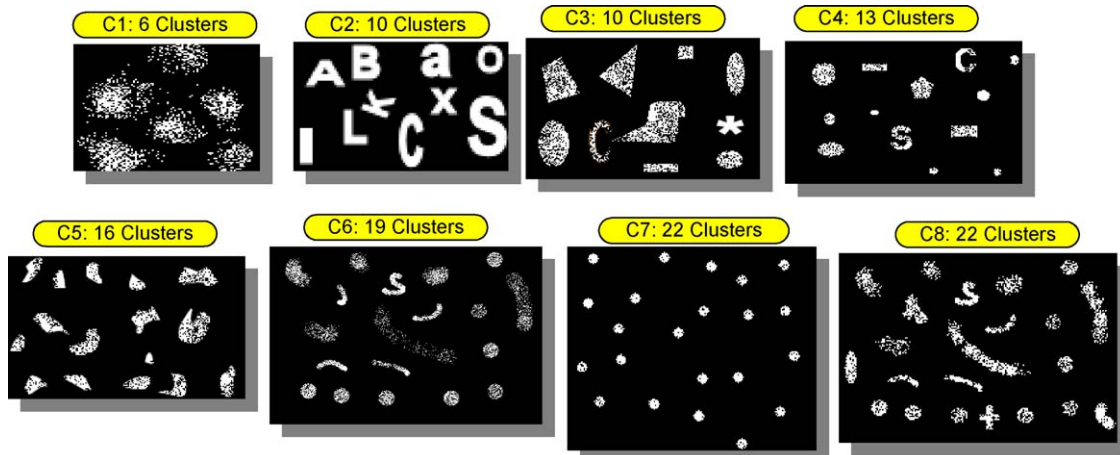


Fig. 2. 2D synthetic data spaces carrying different clustering schemes.

Table 12
Statistical results from 20 runs over 8 2D data spaces.

Clusters	No.	<i>d</i>	MD-PSO				SAD MD-PSO (A2)				SAD MD-PSO (A1)			
			Score		<i>dbest</i>		Score		<i>dbest</i>		Score		<i>dbest</i>	
			μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
C1	6	6	1456.5	108.07	6.4	0.78	1455.2	103.43	6.3	0.67	1473.8	109	6.2	1.15
C2	10	12	1243.2	72.12	10.95	2.28	1158.3	44.13	12.65	2.08	1170.8	64.88	11.65	1.56
C3	10	11	3833.7	215.48	10.4	3.23	3799.7	163.5	11.3	2.57	3884.8	194.03	11.55	2.66
C4	13	14	1894.5	321.3	20.2	3.55	1649.8	243.38	19.75	2.88	1676.2	295.8	19.6	2.32
C5	16	17	5756	1439.8	19	7.96	5120.4	1076.3	22.85	4.17	4118.3	330.31	21.8	2.87
C6	19	28	21,533	4220.8	19.95	10.16	18,323	1687.6	26.45	2.41	20,016	3382	22.3	6.97
C7	22	22	3243	1133.3	21.95	2.8	2748.2	871.1	23	2.51	2380.5	1059.2	22.55	2.8
C8	22	25	6508.85	1014	17.25	10.44	6045.1	412.78	26.45	3.01	5870.25	788.6	23.5	5.55

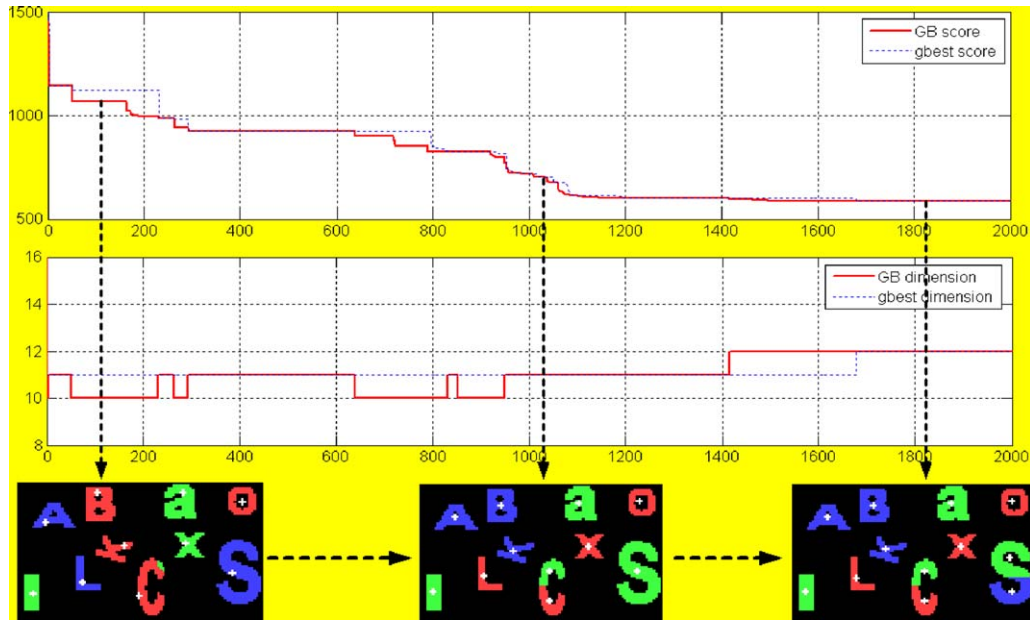


Fig. 3. Fitness score (top) and dimension (bottom) plots vs. iteration number for a clustering operation over C2. Three clustering snapshots at iterations 105, 1050 and 1850, are presented below.

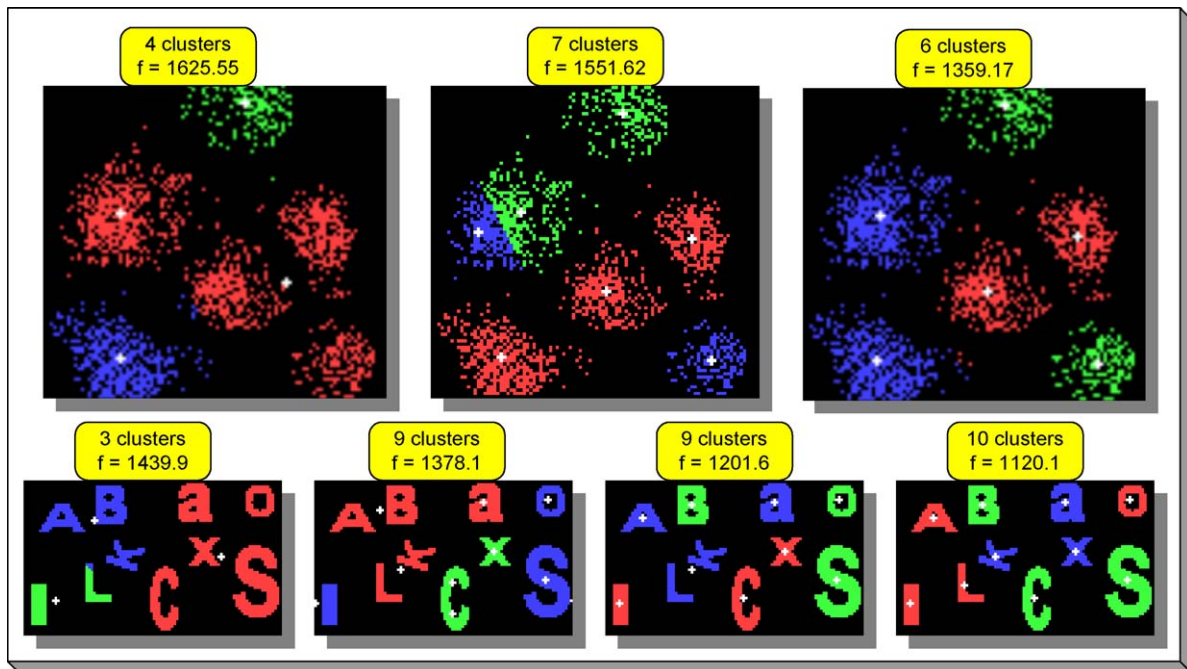


Fig. 4. Some clustering runs with the corresponding fitness scores (*f*).

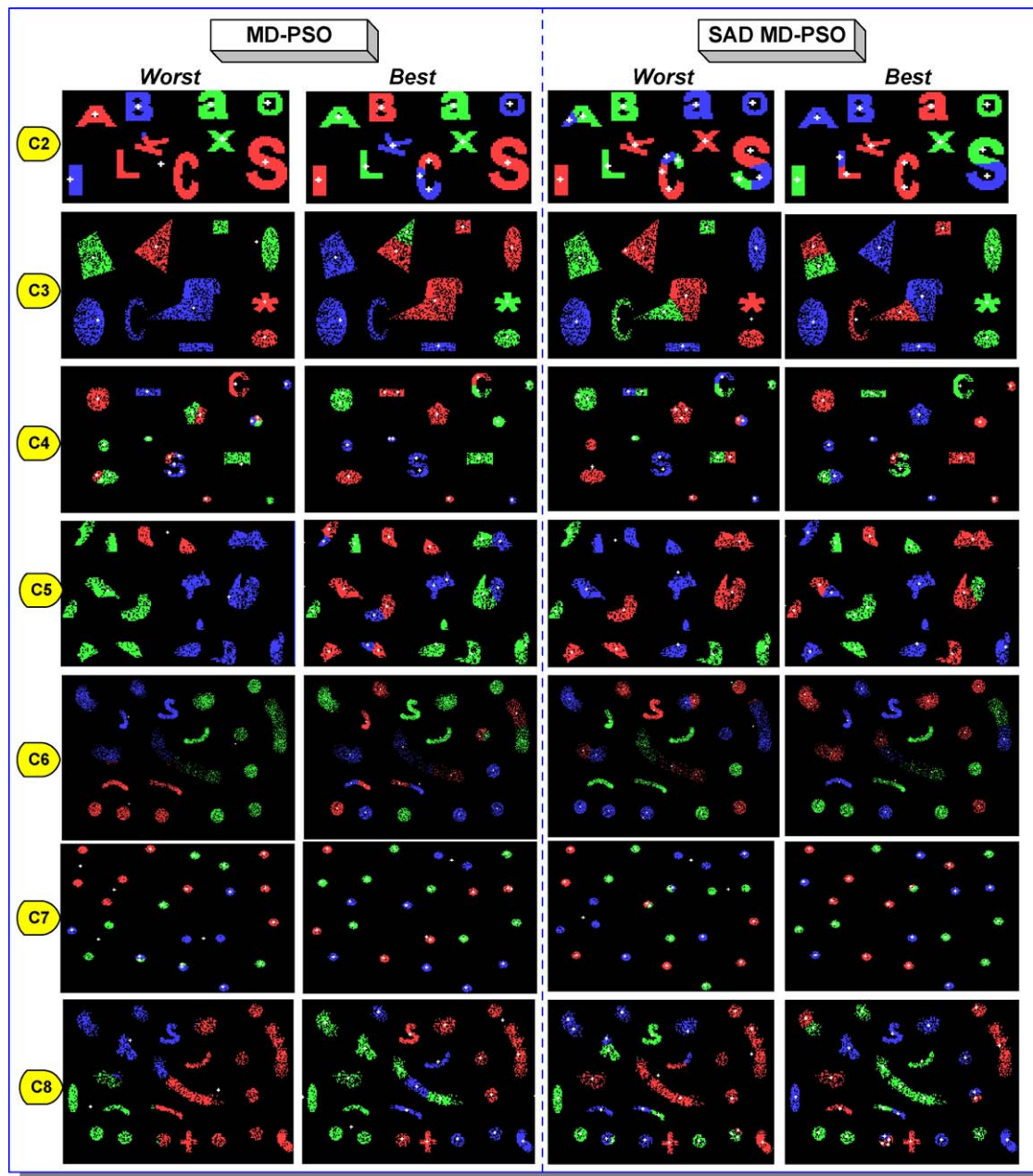


Fig. 5. The worst and the best clustering results using standalone (left) and SAD (right) MD-PSO.

statistics (mean, μ and standard deviation, σ) of the fitness scores and $dbest$ values converged are presented in Table 12 whilst the best statistics are highlighted.

According to the statistics in Table 12, similar comments can be made as in the PSO application on non-linear function minimization, i.e. either SAD MD-PSO approach achieves a superior performance over all data spaces regardless of the number of clusters and cluster complexity (modality) without any exception. The superiority hereby is visible on the average fitness scores achieved as well as the proximity of the average $dbest$ statistics to the optimal dimension. Note that d in the table is the optimal dimension, which may be different than the true number of clusters due to the validity index function used. A sample clustering operation (over data space C2) illustrating this case is shown in Fig. 3 where in the bottom part, 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 '+'. 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 10 clusters (~ 1100) remains higher than the one with 11 (~ 710) and than the final (and optimal) outcome with 12 clusters (~ 570) too. The main reason for this is that the validity index in Eq. (8) 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. C3–C6 and C8), the proposed method yields a slight over-clustering but never under-clustering. Improving the validity index or adopting a more sophisticated one such as Dunn's index [8] or any other, might improve the clustering accuracy; however, this is beyond the scope of this paper. Note that under-clustering, if it occurs, is a major error in a clustering operation, which means that the optimization method got trapped in a local optimum during the early stages.

Some further important conclusions can be drawn from the statistical results in Table 12. First of all, the performance gap tends to increase as the cluster number (dimension of the solution space) rises. For instance all methods have fitness scores in a close vicinity for the data space C1 whilst both SAD MD-PSO approaches perform significantly better for C7. Note, however, that the performance gap for C8 is not as high as in C7, indicating SPSA parameters are not too appropriate for C8 (as a consequence of fixed SPSA parameter setting). On the other hand, in some particular clustering runs, the difference in the average fitness scores in Table 12 does not really correspond to the actual improvement in the clustering quality. Take for instance the two clustering runs over C1 and C2 in Fig. 4 where some clustering instances with the corresponding fitness scores are shown. The first (left-most) instances in both rows are from severely erroneous clustering operation although only a mere difference in fitness scores occurs with the instances in the second column, which have significantly less clustering errors. On the other hand the proximity of the average *dbest* statistics to the optimal dimension may be another alternative for the evaluation of the clustering performance; however, it is fairly probable that two runs, one with severely under- and another with over-clustering, may have an average *dbest* that is quite close to the optimal dimension. Therefore, the standard deviation should play an important role in the evaluation and in this aspect; one can see from the statistical results in Table 12 that the second SAD MD-PSO approach (A2) in particular achieves the best performance (i.e. converging to the true number of clusters and correct localization of the centroids) whilst the performance of the standalone MD-PSO is the poorest.

For visual evaluation, Fig. 5 presents the *worst* and the *best* clustering results of the two competing techniques, standalone vs. SAD MD-PSO, based on the highest (worst) and lowest (best) fitness scores achieved among the 20 runs. The clustering results of the best performing SAD MD-PSO approach, as highlighted in Table 12, are shown whilst excluding C1 since results of all techniques are quite close for this data space due to its simplicity. Note first of all that the results of the (standalone) MD-PSO deteriorate severely as the complexity and/or the number of clusters increases. Particularly in the *worst* results, the critical errors such as under-clustering often occur with dislocated cluster centroids. For instance 4 out of 20 runs for C6 results in severe under-clustering with 3 clusters, similar to the one shown in the figure whereas this goes up to 10 out of 20 runs for C8. Although the clusters are the simplest in shape and in density for C7, due to the high solution space dimension (e.g. number of clusters = 22), even the *best* MD-PSO run is not immune to under-clustering errors. In some of the *worst* SAD MD-PSO runs too, one or few under-clusterings do occur; however, they are minority cases in general and definitely not as severe as in MD-PSO runs. It is quite evident from the *worst* and the *best* results in the figure that SAD MD-PSO achieves a significantly superior clustering quality and usually converges to a close vicinity of the global optimum solution.

5. Conclusions

In this paper, we draw the focus on a major drawback of the PSO algorithm: the poor *gbest* update. This can be a severe problem, which may cause premature convergence to local optima since *gbest* as the common term in the update equation of all particles, is the primary *guide* of the swarm. Therefore, we basically seek a solution for the social problem in PSO, i.e. “Who will guide the guide?” which resembles the rhetoric question posed by Plato in his famous work on government: “Who will guard the guards?” (*Quis custodiet ipsos custodes?*). SA is purposefully adopted to guide (or drive) the *gbest* particle (with simultaneous perturbation) towards the “right” direction with the gradient estimate of the underlying surface (or function) whilst avoiding local traps due to its stochastic nature. In

that, the proposed SAD PSO is not a new PSO variant or extension, rather a “guided PSO” algorithm, which has an identical process with the basic PSO as guidance is only provided to *gbest* particle – of the whole swarm.

In SAD PSO, we have proposed two approaches where SPSA is explicitly used. The first approach replaces the PSO update of *gbest* with the SPSA whereas the second one forms an alternative (or artificial) GB particle (the *aGB*), which can replace *gbest* if it proves its superiority. Both SAD PSO approaches are tested over seven non-linear functions and the experimental results demonstrated that they achieved a superior performance over all functions regardless of the dimension, modality, and without any exception. Especially if the setting of the critical parameters, e.g. *a* and *c* is appropriate, then a significant performance gain can be achieved by SAD PSO. If not, SAD PSO still outperforms *bPSO*. This shows that SPSA, even without proper parameter setting still performs *better* than the PSO’s native *gbest* update. The complexity overhead in SAD PSO is negligible, i.e. only two (or three in the second approach) extra fitness evaluations per iteration and with the proposed low-cost mode, it is further reduced by one. The experimental results show that the low-cost mode does not cause a noticeable performance loss; on the contrary, it occasionally may perform even better.

Both approaches are also integrated into MD-PSO, which defines a new particle formation and integrates the ability of dimensional navigation into the core of the PSO process. Recall that 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. SAD MD-PSO is then applied to the unsupervised clustering problem within which the (clustering) complexity can be thought of as synonymous to (function) modality and tested over eight synthetic data spaces in 2D with ground truth clusters. The statistical results obtained from the clustering runs approve the superiority of SAD MD-PSO in terms of global convergence. As in SAD PSO application for non-linear function minimization, we have applied a *fixed* set of SPSA parameters and hence we can make the same conclusion as before about the effect of the SPSA parameters over the performance. Furthermore, we have noticed that the performance gap widens especially when the clustering complexity increases since the performance of the standalone MD-PSO operation, without any proper guidance, severely deteriorates. One observation worth mentioning is that the second approach on SAD MD-PSO has a significant overhead cost, which is anyway balanced by using a reduced number of particles in the experiments; therefore, the low-cost mode should be used with a limited dimensional range for those applications with high computational complexity.

Encouraged by the results, current plans for future work include the application of SAD (MD-)PSO to other problem domains. Since the SPSA only “estimates” the gradient of the error surface without imposing any continuity, the proposed technique can be used for discrete (and even binary) problems. Particularly, we can foresee that those problems where a proper guidance mechanism such as FGBF is needed but cannot be applied due to infeasibility problems (e.g. evolutionary ANN applications by the standalone MD-PSO as in [18] and [23]), SAD MD-PSO would be a promising solution to further improve the performance. For those multi-objective problems (MOPs), the application of SAD PSO can be crucial since the goal is to search for a set of Pareto-optimal solutions and hence, the particles must follow (a set of) best guide(s) that will lead them toward the optimal solutions. This will thus be the subject of our future research.

References

- [1] A. Abraham, S. Das, S. Roy, Swarm intelligence algorithms for data clustering, in: *Soft Computing for Knowledge Discovery and Data Mining Book*, 2007, Part IV, pp. 279–313, October 25.

- [2] P.J. Angeline, Using selection to improve particle swarm optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE Press, 1998, pp. 84–89.
- [3] P.I. Angeline, Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences, in: Evolutionary Programming VII, Conference EP'98, Springer Verlag, Lecture Notes in Computer Science No. 1447, California, USA, March 1998, pp. 601–410.
- [4] T. Back, H.P. Schwefel, An overview of evolutionary algorithm for parameter optimization, *Evolutionary Computation* 1 (1993) 1–23.
- [5] T. Back, F. Kursawe, Evolutionary algorithms for fuzzy logic: a brief overview, in: Fuzzy Logic and Soft Computing, World Scientific, Singapore, 1995, pp. 3–10.
- [6] D.C. Chin, A more efficient global optimization algorithm based on Styblinski and Tang, *Neural Networks* 7 (1994) 573–574.
- [7] D.L. Davies, D.W. Bouldin, A cluster separation measure, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1 (1979) 224–227.
- [8] J.C. Dunn, Well separated clusters and optimal fuzzy partitions, *Journal of Cybernetics* 4 (1974) 95–104.
- [9] R. Eberhart, P. Simpson, R. Dobbins, *Computational Intelligence. PC Tools*, Academic Press, Inc., Boston, MA, USA, 1996.
- [10] S.C. Esquivel, C.A. Coello Coello, On the use of particle swarm optimization with multimodal functions, *IEEE Transactions on Evolutionary Computation* 2 (2003) 1130–1136.
- [11] U.M. Fayyad, G.P. Shapire, P. Smyth, R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*, MIT Press, Cambridge, MA, 1996.
- [12] H. Frigui, R. Krishnapuram, Clustering by competitive agglomeration, *Pattern Recognition* 30 (1997) 1109–1119.
- [13] S.B. Gelfand, S.K. Mitter, Recursive stochastic algorithms for global optimization, *SIAM Journal on Control and Optimization* 29 (September(5)) (1991) 999–1018.
- [14] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989, pp. 1–25.
- [15] M. Gunther, V. Nissen, a comparison of neighbourhood topologies for staff scheduling with particle swarm optimisation, *Advances in Artificial Intelligence, Lecture Notes in Computer Science* 5803 (2009) 185–192.
- [16] M. Halkidi, Y. Batistakis, M. Vazirgiannis, On cluster validation techniques, *Journal of Intelligent Information Systems* 17 (2–3) (2001) 107–145.
- [17] H. Higashi, H. Iba, Particle swarm optimization with Gaussian mutation, in: Proceedings of the IEEE Swarm Intelligence Symposium, 2003, pp. 72–79.
- [18] T. Ince, S. Kiranyaz, M. Gabbouj, A generic and robust system for automated patient-specific classification of electrocardiogram signals, *IEEE Transactions on Biomedical Engineering* 56 (May(5)) (2009) 1415–1426.
- [19] A.K. Jain, M.N. Murthy, P.J. Flynn, Data clustering: a review, *ACM Computing Reviews* 31 (1999) 264–323.
- [20] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, vol. 4, Perth, Australia, 1995, pp. 1942–1948.
- [21] J. Kennedy, R. Eberhart, Y. Shi, *Swarm Intelligence*, Kaufmann, San Francisco, 2001.
- [22] S. Kiranyaz, T. Ince, A. Yildirim, M. Gabbouj, Fractional particle swarm optimization in multi-dimensional search space, *IEEE Transactions on Systems, Man, and Cybernetics – Part B* 40 (2010) 298–319.
- [23] S. Kiranyaz, T. Ince, A. Yildirim, M. Gabbouj, Evolutionary artificial neural networks by multi-dimensional particle swarm optimization, *Neural Networks* 22 (2009) 1448–1462.
- [24] J. Koza, *Genetic Programming: On the Programming of Computers by means of Natural Selection*, MIT Press, Cambridge, MA, 1992.
- [25] H.J. Kushner, G.G. Yin, *Stochastic Approximation Algorithms and Applications*, Springer, New York, 1997.
- [26] M. Lovberg, T. Krink, Extending particle swarm optimisers with self-organized criticality, in: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 2, 2002, pp. 1588–1593.
- [27] J.L. Maryak, D.C. Chin, Global random optimization by simultaneous perturbation stochastic approximation, in: Proceedings of the 33rd Conference on Winter Simulation, Washington, DC, December 09–12, 2001, pp. 307–312.
- [28] Y. Maeda, T. Kuratani, Simultaneous perturbation particle swarm optimization, in: IEEE Congress on Evolutionary Computation, CEC'06, 2006, pp. 672–676.
- [29] M. Omran, A. Salman, A.P. Engelbrecht, Image classification using particle swarm optimization, in: Conference on Simulated Evolution and Learning, vol. 1, 2002, pp. 370–374.
- [30] M.G. Omran, A. Salman, A.P. Engelbrecht, Dynamic clustering using particle swarm optimization with application in image segmentation, *Pattern Analysis and Applications* 8 (2006) 332–344.
- [31] N.R. Pal, J. Biswas, Cluster validation using graph theoretic concepts, *Pattern Recognition* 30 (1997) 847–857.
- [32] J. Riget, J.S. Vesterstrom, A Diversity-Guided Particle Swarm Optimizer – The ARPSO, Technical Report, Department of Computer Science, University of Aarhus, 2002.
- [33] M. Richards, D. Ventura, Dynamic sociometry in particle swarm optimization, in: Proceedings of the Sixth International Conference on Computational Intelligence and Natural Computing, North Carolina, September, 2003, pp. 1557–1560.
- [34] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of the IEEE Congress on Evolutionary Computation, 1998, pp. 69–73.
- [35] J.C. Spall, Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, *IEEE Transactions on Automatic Control* 37 (1992) 332–341.
- [36] J.C. Spall, Implementation of the simultaneous perturbation algorithm for stochastic optimization, *IEEE Transactions on Aerospace and Electronic Systems* 34 (July) (1998) 817–823.
- [37] M.A. Styblinski, T.-S. Tang, Experiments in nonconvex optimization: stochastic approximation with function smoothing and simulated annealing, *Neural Networks* 3 (4) (1990) 467–483.
- [38] P.N. Suganthan, Particle swarm optimiser with neighborhood operator, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE Press, 1999, pp. 1958–1962.
- [39] R.H. Turi, Clustering-based colour image segmentation, PhD Thesis, Monash University, Australia, 2001.
- [40] F. Van den Bergh, An analysis of particle swarm optimizers, PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.
- [41] F. van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, *IEEE Transactions on Evolutionary Computation* 3 (June) (2004) 225–239.
- [42] E.O. Wilson, *Sociobiology: The New Synthesis*, Belknap Press, Cambridge, MA, 1975.