Stochastic Approximation Driven Particle Swarm Optimization

Serkan Kiranyaz*, Turker Ince** and Moncef Gabbouj*[1]
*Tampere University of Technology, Tampere, Finland
{serkan.kiranyaz, moncef.gabbouj}@tut.fi
**Izmir University of Economics, Izmir, Turkey, turker.ince@ieu.edu.tr

Abstract

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. 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 causes pre-mature 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?). Stochastic approximation (SA) is purposefully adapted into two approaches 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. We purposefully used simultaneous perturbation SA (SPSA) for its low cost and since SPSA is applied only to the gbest (not the entire swarm), both approaches have thus a negligible overhead cost over the entire PSO process. Yet we have shown over a wide range of non-linear functions that both approaches significantly improve the performance of PSO especially if the parameters of SPSA suits to the problem in hand.

1. Introduction

In the problem of finding a root θ^* (either minimum or maximum point) of the gradient equation: $g(\theta) \equiv \frac{\partial L(\theta)}{\partial \theta} = 0$ for some differentiable function $L : R^p \to R^1$, when g is present and L is a 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 in multi-modal nature, which presents many 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 [19], Kushner [13], Gelfand and Mitter [7], and Chin [4], the one and somewhat different SA application is called simultaneous perturbation SA (SPSA) proposed by Spall in 1992 [17]. 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.

The particle swarm optimization (PSO) was introduced by Kennedy and Eberhart [10] 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 [21], which basically show a natural behavior when they search for some target (e.g. food). Henceforth, PSO exhibits certain similarities with the other evolutionary algorithms (EAs) [2] such as Genetic Algorithm (GA) [8], Genetic Programming (GP) [12], Evolution Strategies (ES), [3] and Evolutionary Programming (EP), [6]. The common point of all is that EAs are in population based nature and they can avoid being trapped in a local optimum. Thus they can find the optimum solutions; however, this is never guaranteed. In a PSO process, a swarm of particles (or agents), each of which represent a potential solution to an optimization problem, navigate through the search (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 [10]) 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 [10]). 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 (personal best position of the

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

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 agent in the next iteration.

One major drawback of PSO occurs 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 [15] and it is the main cause of the premature convergence problem especially when the search space is of high dimensions [20] and the problem to be optimized is multi-modal [15]. This makes it clear that 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 the rest of the swarm due to the aforementioned direct link of information flow. This deficiency has been raised in a recent work [11] 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 this study, it has been shown that such an enhanced guidance alone is indeed sufficient to achieve an outstanding global convergence performance on multimodal functions and even on high dimensions. However, the underlying mechanism for creating the *aGB* particle, the so-called fractional GB formation (FGBF), is not generic, 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 and in a generic way, in this paper we shall propose two approaches, one of which moves *gbest* efficiently or simply put, guides it with respect to the function (or error surface). The idea behind 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 probability of getting trapped into a local optimum is further diminished and with the SA, *gbest* is moved according to (approximation of) the gradient of the function. The second approach has a similar motivation with the FGBF proposed in [11], 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. In that sense neither proposed approach is a PSO variant; rather a generic cure for the problem of the original PSO caused by poor *gbest* update. Furthermore, we shall demonstrate that they have a negligible cost, e.g. only few percent increase of the computational complexity that can be easily compensated with a slight reduction either on the swarm size or on 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.

2. The Basic PSO algorithm

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

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

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

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

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

Let f denote the fitness function to be optimized. Without loss of generality assume that the objective is to find the minimum of f in N dimensional space. Then the personal best of particle a can be updated in iteration t+1 as,

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

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

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

where w is the inertia weight, [16] 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

[16], into the velocity update equation, it is widely accepted as the basic form of PSO algorithm. Accordingly, the general pseudo-code of the *bPSO* is presented in Table 1.

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 $\forall t \in [1, IterNo]$ do:
     3.1.  For $\forall a \in [1, S]$ do:
           3.1.1. Compute $y_a(t)$ using (1)
           3.1.2. If ( $f(y_a(t)) < \min\left( f(\hat{y}(t-1)), \underset{1 \le i < a}{f(y_i(t))} \right)$ )
                  then gbest = a and $\hat{y}(t) = y_a(t)$
     3.2.  End For.
     3.3.  If any termination criterion is met, then Stop.
     3.4.  For $\forall a \in [1, S]$ do:
           3.4.1. For $\forall j \in [1, N]$ do:
                  3.4.1.1. Compute $v_{a,j}(t+1)$ using Eq. (2)
                  3.4.1.2. If( $|v_{a,j}(t+1)| > V_{\max}$ ) 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.
```

3. The Proposed Technique: SAD PSO

3.1. SPSA Overview

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

$$g(\theta) \equiv \left. \frac{\partial L(\theta)}{\partial \theta} \right|_{\theta = \theta^*} = 0 \qquad (3)$$

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:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k(\hat{\theta}_k) \qquad (4)$$

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 [17]. 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,

$$\hat{g}_k(\hat{\theta}_k) = \frac{L(\hat{\theta}_k + c_k\Delta_k) - L(\hat{\theta}_k - c_k\Delta_k)}{2c_k} \begin{bmatrix} \Delta_{k1}^{-1} \\ \Delta_{k2}^{-1} \\ \cdot \\ \cdot \\ \cdot \\ \Delta_{kp}^{-1} \end{bmatrix} \qquad (5)$$

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 [17]. Spall in [17] presents conditions for convergence of SPSA (i.e. $\hat{\theta}_k \to \theta^*$) and show that under certain conditions both SPSA and FDSA have the same convergence ability –yet SPSA needs only 2 measurements whereas FDSA needs 2*p*. This makes SPSA our natural choice for driving *gbest* in both approaches.Table 2 presents the general pseudo-code of the SPSA technique.

Table 2: Pseudo-code for *SPSA* technique

```
SPSA (IterNo, a, c, A, $\alpha$ , $\gamma$ )
1.   Initialize $\hat{\theta}_1$
2.   For $\forall k \in [1, IterNo]$ do:
     2.1.  Generate zero-mean, p-dimensional
           perturbation vector: $\Delta_k$
     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 (5)
     2.5.  Compute $\hat{\theta}_{k+1}$ using (4)
3.   End For.
```

SPSA has 5 parameters as given in Table 2. Spall in [18] 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 [14] varied them with respect to the problem whilst keeping the other three (*A*, α, and γ) as recommended.

3.2. SAD PSO

As mentioned earlier, in this work two distinct SAD PSO approaches are proposed, each of which is only applied on the *gbest* 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 to 2.5 in Table 2 are inserted accordingly into the PSO process. The following sub-sections will detail each approach.

A1) First SAD PSO approach: gbest update by SPSA

In this approach, at each iteration *gbest* 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. Due to space limitations, we have to skip the pseudo code details of this approach. Note that such a "plug-in" approach will not change the internal PSO structure and only affects the *gbest* 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 *gbest* 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 $x_a(t)$, instead from $x_a(t) - c_k\Delta_k$.

A2) Second SAD PSO approach: aGB formation by SPSA

The second approach replaces the FGBF operation proposed in [11] 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 3 can then be plugged in between steps 3.3 and 3.4 of *bPSO*'s pseudo-code. Note that in this 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 diminish 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 $\hat{\theta}_k = \hat{y}(t) - c_k\Delta_k$.

Table 3: PSO Plug-in for the second approach

```
A2) SAD PSO Plug-in ( $\xi$ , a, c, A, $\alpha$ , $\gamma$ )
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. (5)
6.  Compute $x_{aGB}(t) = \hat{\theta}_{k+1}$ using Eq. (4)
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 $\hat{y}(t) = y_{aGB}(t+1)$
```

4. Experimental Results

We used 5 benchmark functions, which are given in Table 4 to provide a good mixture of complexity and modality and have been widely studied by several researchers, e.g. see [1], [5], and [16]. *Sphere* and *Rosenbrock* are the uni-modal functions and the rest are multi-modal, meaning that they have many deceiving local minima. On the macroscopic level *Griewank* demonstrates certain similarities with uni-modal functions especially when the dimensionality is above 20; however, in low dimensions it bears a significant noise, which creates many local minima due to the second multiplication term with *cosine* components.

Both approaches of the proposed SAD PSO along with the "low cost" application are tested over 5 benchmark functions given in Table 4 and compared with the *bPSO* and stand-alone SPSA application. We used the same termination criteria as the combination of the maximum number of iterations allowed (*iterNo* = 10000) and the cut-off error ($\varepsilon_C = 10^{-4}$). 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 40x10000 = 4e+5 for SPSA).

Table 4: 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
Rosenbrock	$F_2(x,d) = \left(\sum_{i=1}^{d} 100\,(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$	[-50, 25]	20, 50, 80
Rastrigin	$F_3(x,d) = \left(\sum_{i=1}^{d} 10 + x_i^2 - 10Cos(2\pi x_i)\right)$	[-500, 250]	20, 50, 80
Griewank	$F_4(x,d) = \left(\frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} Cos\left(\frac{x_i}{\sqrt{i+1}}\right)\right)$	[-500, 250]	20, 50, 80
Giunta	$F_5(x,d) = \left(\sum_{i=1}^{d} Sin\left(\frac{16}{15}x_i - 1\right) + Sin^2\left(\frac{16}{15}x_i - 1\right) + \frac{1}{50}Sin\left(4\left(\frac{16}{15}x_i - 1\right)\right) + \frac{268}{1000}\right)$	[-500, 250]	20, 50, 80

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 presented in Table 5 whilst the best statistics are highlighted. During each run, the operation terminates when the fitness score drops below the cut-off error ($\varepsilon_C = 10^{-4}$) and it is assumed that the global minimum of the function is reached, henceforth; the score is set to 0. Therefore, for any setting yielding μ=0 as the average score means that the method converges to the global minimum at every run.

Table 5: Statistical results from 100 runs over 5 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**
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	13989
	80	2.8251e+007	5.7896e+006	24978	**23257**	**20733**	24160	378210	137410
Giunta	**20**	42743	667.2494	495.0777	245.1220	445.1360	264.1160	**445.1356**	**249.5412**
	50	10724	1027.6	4257	713.1723	3938.9	626.9194	**3916.2**	**758.3290**
	80	17283	1247.9	9873.6	1313	8838.2	1357	**8454.2**	**1285.3**

As the entire statistics in the right side of Table 5 indicate, either SAD PSO approach achieves a superior performance over all functions regardless of the dimension, modality, and without any exception. In other words, SAD PSO always works better than the best of bPSO and SPSA even though either of them does not perform well 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 Schwefel. On the other hand, if SPSA does not perform well, even much worse than any other technique, SAD PSO still surpasses bPSO up to some certain degrees, i.e. see Giunta and particularly Griewank for d=50 where SAD PSO can still converge to the global optimum (μ=0) although 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 vice versa is also true, that is, SAD PSO, which internally runs SPSA for gbest achieves better performance than the SPSA alone, even though when PSO's performance is limited.

5. Conclusions

In this paper we have basically proposed two SAD PSO approaches that 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 surpasses *bPSO* by some certain magnitudes. This basically 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 evaluation 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 contrary, it occasionally may perform even better.

5. References

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

[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] D. C. Chin, "A More Efficient Global Optimization Algorithm Based on Styblinski and Tang," Neural Networks, pp. 573-574, 1994.

[5] S. C. Esquivel and C. A. Coello Coello, "On the Use of Particle Swarm Optimization with Multimodal Functions", In IEEE Trans. on Evolutionary Computation, vol. 2, pp. 1130-1136, 2003.

[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] S. B. Gelfand and S. K. Mitter, "Recursive stochastic algorithms for global optimization", in Rd, SIAM Journal on Control and Optimization, vol. 29, no.5, pp. 999-1018, Sept. 1991.

[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] J. Kennedy, R Eberhart., "Particle swarm optimization", in Proc. of IEEE Int. Conf. On Neural Networks, vol. 4, pp. 1942–1948, Perth, Australia, 1995.

[11] S. Kiranyaz, T. Ince, A. Yildirim and M. Gabbouj, "Fractional Particle Swarm Optimization in Multi-Dimensional Search Space", IEEE Transactions on Systems, Man, and Cybernetics, 2009. In Print.

[12] J. Koza, Genetic Programming: On the Programming of Computers by means of Natural Selection, MIT Press, Cambridge, Massachussetts, 1992.

[13] H. J. Kushner, and G. G. Yin, Stochastic Approximation Algorithms and Applications, Springer, New York, 1997.

[14] , J. L. Maryak and D. C. Chin, "Global random optimization by simultaneous perturbation stochastic approximation", In Proc. of the 33rd Conf. on Winter Simulation, Washington, DC, pp. 307-312, Dec. 09 - 12, 2001.

[15] J. Riget and J. S. Vesterstrom, "A Diversity-Guided Particle Swarm Optimizer - The ARPSO", Technical report, Department of Computer Science, University of Aarhus, 2002.

[16] Y. Shi and R.C. Eberhart, "A Modified Particle Swarm Optimizer", In Proc. of the IEEE Congress on Evolutionary Computation, pp. 69-73, 1998.

[17] J. C. Spall, "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation", IEEE Transactions on Automatic Control, vol. 37, pp. 332-341, 1992.

[18] J. C. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization", IEEE Trans. on Aerospace and Electronic Systems, vol. 34, pp. 817-823, July 1998.

[19] M. A. Styblinski , T. -S. Tang, "Experiments in nonconvex optimization: stochastic approximation with function smoothing and simulated annealing", Neural Networks, vol. 3 no. 4, pp. 467-483, 1990.

[20] F. Van den Bergh, "An Analysis of Particle Swarm Optimizers", PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[21] E.O. Wilson, Sociobiology: The new synthesis, Cambridge, MA: Belknap Press, 1975.