Bare Bones Particle Swarm Optimization With Scale Matrix Adaptation

Mauro Campos, Renato A. Krohling, and Ivan Enriquez

***Abstract*—Bare bones particle swarm optimization (BBPSO) is a swarm algorithm that has shown potential for solving single-objective unconstrained optimization problems over continuous search spaces. However, it suffers of the premature convergence problem that means it may get trapped into a local optimum when solving multimodal problems. In order to address this drawback and improve the performance of the BBPSO, we propose a variant of this algorithm, named by us as BBPSO with scale matrix adaptation (SMA), SMA-BBPSO for short reference. In the SMA-BBPSO, the position of a particle is selected from a multivariate *t*-distribution with a rule for adaptation of its scale matrix. We use the multivariate *t*-distribution in its hierarchical form, as a scale mixtures of normal distributions. The *t*-distribution has heavier tails than those of the normal distribution, which increases the ability of the particles to escape from a local optimum. In addition, our approach includes the normal distribution as a particular case. As a consequence, the *t*-distribution can be applied during the optimization process by maintaining the proper balance between exploration and exploitation. We also propose a simple update rule to adapt the scale matrix associated with a particle. Our strategy consists of adapting the scale matrix of a particle such that the best position found by any particle in its neighborhood is sampled with maximum likelihood in the next iteration. A theoretical analysis was developed to explain how the SMA-BBPSO works, and an empirical study was carried out to evaluate the performance of the proposed algorithm. The experimental results show the suitability of the proposed approach in terms of effectiveness to find good solutions for all benchmark problems investigated. Nonparametric statistical tests indicate that SMA-BBPSO shows a statistically significant improvement compared with other swarm algorithms.**

***Index Terms*—Multivariate *t*-distribution, scale matrix adaptation (SMA), scale mixtures of normal distributions, swarm algorithms.**

I. Introduction

PARTICLE swarm optimization (PSO) is an algorithm for global optimization originally introduced in [1] for single-objective unconstrained optimization problems. Since its introduction, many researchers have expanded on the original idea. They have derived new versions, developed new applications, and published empirical and theoretical studies about the effects of the various parameters and aspects of the proposed algorithms. The interest in this topic has grown steadily and [2]–[5] provide an overview of the fundamentals of swarm algorithms. The original idea of the PSO algorithm was inspired by the social behavior of some species of animals to work as a whole in locating desirable positions in a given area. This seeking behavior was associated with that of a search for solutions to a given optimization problem [2]. Therefore, PSO is a population-based algorithm motivated by the simulation of social behavior instead of evolution as in evolutionary algorithms.

PSO is initialized with a population of particles with random positions and velocities in the search space of the problem. Each particle is associated with a potential solution, and the particles are moving through the search space obeying dynamic rules to update their positions and velocities. Each particle is capable of interacting with the environment and with other particles, particularly those particles of its neighborhood. Each particle keeps the memory of the personal best solution found during the search process, and the particles in the swarm use a neighborhood structure to exchange information between them. As a result, each particle also keeps the memory of the best solution found by any particle in its neighborhood during the search process. The particles search for solutions to a given problem learning from their own past experiences and from the experiences of their neighbors. The swarm as a whole explores the search space, first at random, and then, when better solutions are found and communicated, the swarm begins to converge refining its search until a good enough solution is found. Clerc and Kennedy [6] introduced a PSO with a constriction factor, in which limits for the parameters of the algorithm were determined for ensuring stability and local convergence. The PSO of Clerc and Kennedy is currently known as the standard PSO [2].

Bare bones PSO (BBPSO) is a well known variant of swarm algorithm originally introduced in [7]. Kennedy proposed to change the position of a particle according to a probability distribution rather than to add a velocity in the current position, as is done in PSO. The swarm in BBPSO explores the search space of a given problem by sampling of explicit probabilistic models constructed from the information associated with promising candidate solutions. The search for solutions is the result of a constructive cooperation between particles by using probabilistic models whose parameters are defined in terms

Manuscript received June 7, 2013; revised October 14, 2013; accepted October 30, 2013. Date of publication November 19, 2013; date of current version August 14, 2014. This paper was recommended by Associate Editor Y. Tan.

M. Campos and I. Enriquez are with the Department of Statistics, Federal University of Espírito Santo, Vitória ES 29075-910, Brazil (e-mail: maurocm.campos@gmail.com; ivanrobertenriquez@gmail.com).

R. A. Krohling is with the Department of Production Engineering and the Graduate Program in Computer Science, Federal University of Espírito Santo, Vitória ES 29075-910, Brazil (e-mail: krohling.renato@gmail.com).

Digital Object Identifier 10.1109/TCYB.2013.2290223

of the information that is obtained during the optimization process (the personal and neighborhood best positions).

It has been conjectured that heavy-tailed distributions can increase the chances of a particle (or individual in a population) to move away from a local optimum. The general idea is to allow exploration in regions far away from the current position. Yao *et al.* [8] introduced a fast evolutionary programming (FEP) that uses mutations based on the Cauchy distribution instead of mutations based on the Gaussian distribution (or normal distribution) which is used in the classical evolutionary programming (CEP). The experimental results show that FEP performs much better than CEP for multimodal functions with many local minima, while being comparable to CEP in performance for unimodal and multimodal functions with only a few local minima. The authors report the fact that the Cauchy mutation performs better when the current search point is far away from the global minimum, while Gaussian mutation is better at finding a local optimum in a good region.

Lee and Yao [9] introduced an evolutionary programming (EP) with mutations based on the Lévy distribution. This distribution is symmetrical with respect to zero and has two parameters α and γ (γ is the scaling factor satisfying $\gamma > 0$ and α controls the shape of the distribution satisfying $0 < \alpha \leq 2$). Its density resembles a bell-shaped curve as the Gaussian density, but with heavier tails (α can be used to control the heaviness of the tails). The Lévy distribution includes some models as special cases: the Cauchy distribution when $\alpha = 1$ and the Gaussian distribution when $\alpha = 2$, to cite two examples. An algorithm for generating Lévy random numbers was introduced in [10]. Lee and Yao reported empirical evidence that the performance of the EP using Lévy mutation was better than that of the CEP for multimodal functions with many local minima, because the Lévy mutation is more general and flexible than Cauchy and Gaussian mutations.

BBPSO suffers of the premature convergence problem which means it may get trapped into a local optimum when solving multimodal problems. Some strategies have been developed to address this drawback and improve the performance of the original algorithm. Richer and Blackwell [11] investigated the effectiveness of using the Lévy distribution in different swarm algorithms. In a series of trials, Richer and Blackwell found that the Lévy BBPSO with $\alpha = 1.4$ reproduces the standard PSO behavior, without necessity to add by hand particle bursts to produce bursts of outliers. This result supports the conjecture that a heavy-tailed distribution to update the position of a particle in swarm algorithms provides an increase in the ability of the particles to move away from a local optimum.

Krohling and Mendel [12] introduced a BBPSO with a jump strategy when no improvement on the value of the objective function is observed. The jump strategy was implemented based on the Gaussian or Cauchy distributions. The algorithm was tested on a suite of well known benchmark multimodal functions and the results were compared with those obtained by the BBPSO algorithm. Simulation results shown that the BBPSO algorithm with a jump strategy has performed well in all functions investigated. The authors also pointed that the improved performance was due to a successful number of Gaussian or Cauchy jumps, with a performance slightly better for the case of Cauchy jumps. This result is also essentially compatible with the conjecture that a distribution with heavy tails increases the chances of a particle to escape or move away from a local optimum, improving, therefore, the balance between exploration and exploitation.

Blackwell [13] formulated the dynamic update rule of the PSO as a second-order stochastic difference equation. This formulation was used to derive general expressions for search focus, search spread, and swarm stability at stagnation. The results were applied to three swarm algorithms: the standard PSO of Clerc and Kennedy, PSO with discrete recombination, and BBPSO. Blackwell proposed a generalized BBPSO such that the search focus and the search spread can each one be chosen from the global or local neighborhoods. A theoretical analysis was presented, along with empirical studies, to derive a no-collapse condition for the generalized BBPSO. Collapse is a possible swarm pathology: particles approach each other faster than the swarm as a whole approaches a local optimum. The result is that convergence toward a limit point becomes exponentially slow and the swarm stagnates [13]. The analysis conducted by Blackwell predicts a critical α_c, such that the system resists collapse if $\alpha > \alpha_c$. In addition, the fastest rate of convergence of the BBPSO occurs at α_c. The experimental results confirm that the BBPSO situated at the edge of collapse is comparable to the standard PSO and PSO with discrete recombination. Further results indicated that the performance of the proposed algorithm can be still further improved with the use of an adaptive distribution with heavy tails. In fact, Blackwell also proposed a BBPSO with jumps (BBPSOwJ), following a different strategy from that which was considered in [12]. The BBPSOwJ can be seen as the generalized BBPSO combined with a probabilistic jumping mechanism: a particle may jump uniformly in any dimension with probability p_J. This strategy can be seen as a partial reinitialization (since in general not every component undergoes a jump) or, alternatively, as a mechanism to fatten the tails of the normal distribution that generates new positions in the search space, allowing search in areas where the tails of the normal distribution are thin. This adaptive distribution allows exploration throughout the search volume at any stage of the optimization.

Hsieh and Lee [14] also introduced a modified BBPSO. Empirical studies presented in this paper considering a well known set of test functions show that the proposed algorithm can be a competitive optimizer due to its good performance and fast convergence rate. The BBPSO of Hsieh and Lee shows some similar characteristics to BBPSO algorithm proposed in [13], having the former an additional constriction parameter to the search focus.

In order to address the problem of premature convergence of the BBPSO and improve the performance of the algorithm, inducing exploration at any stage of the search process and enabling escape from local minima, we propose a variant of the BBPSO, named by us as BBPSO with scale matrix adaptation (SMA), SMA-BBPSO for short reference. In the SMA-BBPSO, the position of a particle is selected from a multivariate t-distribution with a rule for adaptation of its

scale matrix. We use the multivariate t-distribution in its hierarchical form, as a scale mixtures of normal distributions [15], [16]. The t-distribution has heavier tails than those of the normal distribution, which increases the ability of the particles to escape from a local optimum. In addition, our approach includes the normal distribution as a particular case. As a consequence, the t-distribution can be applied during the optimization process by maintaining the proper equilibrium between exploration and exploitation. We also propose a simple update rule to adapt the scale matrix associated with a particle. Our strategy consists of adapting the scale matrix of a particle such that the best position found by any particle in its neighborhood is sampled with maximum likelihood in the next iteration. As a consequence, each particle in the SMA-BBPSO selects new positions in the search space of the problem using a self-adaptive distribution and considering its accumulated learning until the current iteration, without discarding any information so far.

The remainder of this paper is organized as follows. Section II presents a short review of swarm algorithms. Section III presents formally the SMA-BBPSO, along with a theoretical analysis to explain how the algorithm works. Section IV presents the experimental results of an empirical study carried out to evaluate the performance of the proposed approach. Finally, conclusions and directions for future research are given in Section V.

II. Swarm Algorithms

PSO can be described as follows. Consider a swarm $\mathcal{S}$ with S particles. The position of a particle is denoted by $\boldsymbol{x}_k = (x_{kd} : d = 1, \ldots, D)^T$, which is a D-dimensional vector in the search space of the problem. The index k ($k = 1, \ldots, S$) labels the kth particle in the swarm. The velocity of a particle, denoted by $\boldsymbol{v}_k = (v_{kd} : d = 1, \ldots, D)^T$, is defined by the change in its position.

PSO explores the search space by modifying the velocity of each particle. The question that remains to be answered is how to define the velocity of the particle so that it moves in the appropriate directions and with the appropriate step size in the search space. The original inspiration suggests that particles should be influenced by their own previous experiences and the experiences of their neighbors. Therefore, it is necessary to define a neighborhood for each particle. The neighborhood of a particle is a set $\mathcal{N}_k$ of particles which it is able to communicate with ($\mathcal{N}_k \subseteq \mathcal{S}$). In the PSO, a neighborhood system $\mathcal{N} = \{\mathcal{N}_k : k = 1, \ldots, S\}$ is a communication structure (or topology) often thought of as a social network. There is a number of different schemes to connect the particles of the population. Most PSO implementations use one of two simple sociometric principles. The first, called global topology or Gbest model, conceptually connects each particle in the population with all others. The second, called local topology or Lbest model, creates a neighborhood for each particle comprising generally of the particle itself and L neighbors in the population ($L < S$).

Considering a PSO neighborhood system, we can define two vectors that are associated with each particle in the swarm and with the information communicated through its neighborhood: the best position found by a particle up to the current time (the personal best position) and the best position found by any particle of its neighborhood up to the current time (the neighborhood best position). The personal and neighborhood best positions are denoted by $\boldsymbol{p}_k$ and $\boldsymbol{n}_k$, respectively.

Now, we can present the rules that control the dynamics of the particles. PSO is initialized with a population of particles with random positions and velocities in the search space. On the initialization, the personal best position of a particle is equal to its initial position and the neighborhood best position is given by

$$\boldsymbol{n}_k \leftarrow \text{BEST}(\boldsymbol{p}_l : l \in \mathcal{N}_k) := \arg\min\{f(\boldsymbol{p}_l)\}_{l \in \mathcal{N}_k}. \quad (1)$$

The velocity of a particle is updated by the following equation:

$$\boldsymbol{v}_k^{\text{new}} \leftarrow \boldsymbol{v}_k + \boldsymbol{r}_1 \otimes (\boldsymbol{p}_k - \boldsymbol{x}_k) + \boldsymbol{r}_2 \otimes (\boldsymbol{n}_k - \boldsymbol{x}_k). \quad (2)$$

The change in its position is given by

$$\boldsymbol{x}_k^{\text{new}} \leftarrow \boldsymbol{x}_k + \boldsymbol{v}_k^{\text{new}}. \quad (3)$$

After updating the position, the personal best position is updated as follows:

$$\boldsymbol{p}_k^{\text{new}} \leftarrow \text{BEST}(\boldsymbol{x}_k^{\text{new}}, \boldsymbol{p}_k). \quad (4)$$

Finally, the neighborhood best position is given by

$$\boldsymbol{n}_k^{\text{new}} \leftarrow \text{BEST}(\boldsymbol{p}_l^{\text{new}} : l \in \mathcal{N}_k). \quad (5)$$

In (2), $\boldsymbol{r}_j$ ($j = 1, 2$) represents a vector of random numbers uniformly distributed in $(0, c_j)$ and $\otimes$ represents the component-wise vector multiplication. The parameters c_1 and c_2 determine the magnitude of the random forces in the direction of $\boldsymbol{p}_k$ and $\boldsymbol{n}_k$, respectively. In summary, each particle in PSO is associated with a potential solution and the particles are moving through the search space of the problem with dynamic rules controlled by (2) and (3).

A. Particle Swarm Optimization With Inertia Weight

In the original version of the PSO, each component of $\boldsymbol{v}_k$ is kept within the range $[-v_{\max}, v_{\max}]$. The choice of the parameter $v_{\max}$ requires some care since it appears to influence the balance between exploration and exploitation. Shi and Eberhart [17] suggested that by using a decreasing factor w, termed inertia weight, the PSO was able to improve the exploration with exploitation without using any $v_{\max}$. The result was a PSO with the following update rule for the velocity of a particle:

$$\boldsymbol{v}_k \leftarrow w\boldsymbol{v}_k + \boldsymbol{r}_1 \otimes (\boldsymbol{p}_k - \boldsymbol{x}_k) + \boldsymbol{r}_2 \otimes (\boldsymbol{n}_k - \boldsymbol{x}_k). \quad (6)$$

The parameter w can, for instance, be started at 0.9 and be linearly decreased until 0.4 [17].

B. Standard Particle Swarm Optimization

Clerc and Kennedy [6] introduced a PSO with a constriction factor (also called standard PSO). In this case, the velocity of a particle is updated as follows:

$$\boldsymbol{v}_k \leftarrow \chi[\boldsymbol{v}_k + \boldsymbol{r}_1 \otimes (\boldsymbol{p}_k - \boldsymbol{x}_k) + \boldsymbol{r}_2 \otimes (\boldsymbol{n}_k - \boldsymbol{x}_k)] \tag{7}$$

where $\varphi = c_1 + c_2 > 4$ and

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}. \tag{8}$$

The parameter φ is commonly set to 4.1, $c_1 = c_2 = 2.05$, and $\chi \approx 0.7298$ [2].

C. Fully Informed Particle Swarm

Mendes *et al.* [18] developed a fully informed particle swarm (FIPS), where each particle is influenced by all of its neighbors rather than just by the best one in its neighborhood. The FIPS algorithm was further improved in [19] and it can be considered a generalization of the standard PSO. While the standard PSO adds two terms to the velocity and divides the constant φ in half to weight each of them, the FIPS distributes the weight of φ across the entire neighborhood. In this case, the velocity of a particle is updated as follows:

$$\boldsymbol{v}_k \leftarrow \chi \left[\boldsymbol{v}_k + \sum_{l \in \mathcal{N}_k} \frac{\boldsymbol{r}_l \otimes (\boldsymbol{p}_l - \boldsymbol{x}_k)}{|\mathcal{N}_k|} \right] \tag{9}$$

where $|\mathcal{N}_k|$ is the number of neighbors that the particle k has and $\boldsymbol{r}_l$ are vectors of random numbers uniformly distributed in $(0, \varphi)$. FIPS has been applied to solve multimodal optimization problems providing very good results, and it is considered a powerful optimization algorithm, with good capacity to maintain the diversity among particles due to the strong social influence to update the velocity.

D. Bare Bones Particle Swarm Optimization

BBPSO [7] can be described as follows. Consider a swarm $\mathcal{S}$ with S particles and a neighborhood system $\mathcal{N}$. Gbest and Lbest models are possible such as in the PSO. The position of a particle is updated as follows:

$$\boldsymbol{x}_k \leftarrow \boldsymbol{\mu}_k + \boldsymbol{\sigma}_k \otimes \boldsymbol{z} = \boldsymbol{\mu}_k + \sqrt{\boldsymbol{\Sigma}_k}\boldsymbol{z} \tag{10}$$

where

$$\begin{aligned} \boldsymbol{\mu}_k &\leftarrow \frac{1}{2}(\boldsymbol{p}_k + \boldsymbol{n}_k) = \frac{1}{2}(p_{kd} + n_{kd} : d = 1, \ldots, D)^T \\ \boldsymbol{\sigma}_k &\leftarrow |\boldsymbol{p}_k - \boldsymbol{n}_k| = (|p_{kd} - n_{kd}| : d = 1, \ldots, D)^T \\ \sqrt{\boldsymbol{\Sigma}_k} &\leftarrow \text{diag}(|p_{kd} - n_{kd}| : d = 1, \ldots, D) \end{aligned}$$

and $\boldsymbol{z} \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{I}) = (N_1(0,1), \overset{idd}{\ldots}, N_D(0,1))^T$. The symbol $\sim$ indicates that $\boldsymbol{z}$ has multivariate normal distribution with mean vector $\boldsymbol{0}$ and covariance matrix $\boldsymbol{I}$ (the identity matrix). We use here the same notation presented in Appendix A. The personal best position is updated according to (4) and the neighborhood best position is updated according to (5). It is important to note that the probability density function (pdf) of $\boldsymbol{x}_k$ is given by $\boldsymbol{x}_k \sim \boldsymbol{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ for all k.

The original idea of the BBPSO was inspired by a plot of the distribution of positions attained by a single particle in the PSO by moving in one dimension under the influence of the personal and neighborhood best positions, both fixed in a constant value. This empirical distribution resembled a bell curve such as a normal density. Performance comparisons between BBPSO and other swarm algorithms were presented in [7]. Further analysis pointed that what appeared to be a bell curve actually has a kurtosis which increases with the iteration, and the empirical distribution, in fact, has heavier tails than those of a normal distribution. Kennedy [20] suggested that the origin of this effect can be explained by the production of bursts of outliers, which fatten the tails of the empirical distribution. The addition by hand of particle bursts to produce bursts of outliers ameliorated the situation somewhat, indicating that the tails of a normal distribution are very light to allow the particles escape from stagnation.

Require: p_J
for $d = 1, \ldots, D$ **do**
  if $U(0,1) < p_J$ **then**
    $x_{kd} \leftarrow U(l_d, u_d)$
  else
    $x_{kd} \leftarrow \mu_{kd} + \alpha\delta_{kd}N(0,1)$
  end if
end for

Fig. 1. Jump mechanism in BBPSOwJ.

Blackwell [13] proposed a generalized BBPSO such that the position of a particle is updated as follows:

$$\boldsymbol{x}_k \leftarrow \boldsymbol{\mu}_k + \alpha\sqrt{\boldsymbol{\Sigma}_k}\boldsymbol{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{11}$$

where

$$\begin{aligned} \boldsymbol{\mu}_k &\leftarrow \text{BEST}(\boldsymbol{p}_l : l \in \mathcal{N}_k) \text{ or } \text{BEST}(\boldsymbol{p}_k : k = 1, \ldots, S) \\ \sqrt{\boldsymbol{\Sigma}_k} &\leftarrow \text{diag}(\delta_{kd} : d = 1, \ldots, D) \end{aligned}$$

and $\delta_{kd} = |p_{kd} - n_{kd}|$ or $\delta_{kd} = |p_{(k-1)d} - p_{(k+1)d}| \pmod S$. As mentioned early, the theoretical analysis presented by Blackwell predicts a no-collapse condition when $\alpha > \alpha_c = 0.65$.

Blackwell [13] also proposed a generalized BBPSO with jumps (BBPSOwJ). BBPSOwJ can be seen as the generalized BBPSO combined with a jump mechanism: a particle may jump uniformly in any dimension with probability p_J. This jump mechanism is shown in Fig. 1.

III. Bare Bones PSO With Scale Matrix Adaptation

To keep the balance between exploration and exploitation during an optimization process, different algorithms have used several search mechanisms. In particular, two mechanisms have been strongly investigated: first, probability distributions with heavy tails have been used to allow exploration during any stage of the search [8], [9], [11]–[14], and second, self-adaptive strategies have been used to allow that the algorithm itself learns its parameters from the accumulated experience during the search process [21]–[27].

In this paper, we propose a variant of the BBPSO algorithm that uses a heavy-tailed distribution to select new positions in the search space. Our strategy consists of using the multivariate t-distribution to update a position of a particle. We use the

Require: $S, \mathcal{N}, \beta, m_{\max}, f$
for each particle **do**
  $\boldsymbol{x}_k \leftarrow \underline{\boldsymbol{x}} + (\overline{\boldsymbol{x}} - \underline{\boldsymbol{x}}) \otimes \boldsymbol{U}(0,1)$
  $\boldsymbol{p}_k \leftarrow \boldsymbol{x}_k$
  $\Sigma_k \leftarrow \boldsymbol{I}$
end for
for each particle **do**
  $\boldsymbol{n}_k \leftarrow \text{BEST}(\boldsymbol{p}_l : l \in \mathcal{N}_k)$
end for
repeat
  for each particle **do**
    Update $\boldsymbol{\mu}_k$ according to (13)
    Update $\boldsymbol{\Sigma}_k$ according to (14)
    for $m \in \{0, 1, 2, 3, \ldots, m_{\max}\}$ **do**
      $\nu \leftarrow 2^m$; $\lambda \sim Ga(\frac{\nu}{2}, \frac{\nu}{2})$
      $\boldsymbol{z} \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{I})$
      $\boldsymbol{x}_k|\lambda \leftarrow \boldsymbol{\mu}_k + \lambda^{-1/2}\sqrt{\boldsymbol{\Sigma}_k}\boldsymbol{z}$
      $\boldsymbol{p}_k \leftarrow \text{BEST}(\boldsymbol{x}_k, \boldsymbol{p}_k)$
    end for
  end for
  for each particle **do**
    $\boldsymbol{n}_k \leftarrow \text{BEST}(\boldsymbol{p}_l : l \in \mathcal{N}_k)$
  end for
until some termination condition is met
$\boldsymbol{x}_{\min} \leftarrow \text{BEST}(\boldsymbol{n}_k : k = 1 \ldots S)$
return $\boldsymbol{x}_{\min}$ and $f_{\min} = f(\boldsymbol{x}_{\min})$

Fig. 2. SMA-BBPSO($S, \mathcal{N}, \beta, m_{\max}$).

multivariate t-distribution in its hierarchical form, as a scale mixtures of normal (SMN) distributions [15], [16] (for more details on SMN distributions, see Appendix B).

We also propose a rule for adaptation of the scale matrix of the multivariate t-distribution for updating the position of a particle. We conjecture that the use of a self-adaptive distribution with heavy tails can improve the balance between exploration and exploitation during an optimization process.

SMA-BBPSO can be described as follows. Consider a swarm $\mathcal{S}$ with S particles and a neighborhood system $\mathcal{N}$. Once again, Gbest and Lbest models are possible such as in the BBPSO. Each particle is characterized by a vector $(\boldsymbol{x}_k, \boldsymbol{p}_k, \boldsymbol{n}_k)^T$, where each component has the same interpretation already discussed in Section II. The position of a particle is updated as follows:

$$\boldsymbol{x}_k|\lambda \leftarrow \boldsymbol{\mu}_k + \lambda^{-1/2}\sqrt{\boldsymbol{\Sigma}_k}\boldsymbol{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{12}$$

where

$$\boldsymbol{\mu}_k \leftarrow \text{BEST}(\boldsymbol{p}_l : l \in \mathcal{N}_k) = \boldsymbol{n}_k \tag{13}$$

$$\boldsymbol{\Sigma}_k \leftarrow (1 - \beta)\boldsymbol{\Sigma}_k + \beta \boldsymbol{n}_k \boldsymbol{n}_k^T \tag{14}$$

$\lambda \sim Ga(\frac{\nu}{2}, \frac{\nu}{2})$ ($\nu > 0$), and $\beta \in (0, 1)$. On the initialization, assume that $\boldsymbol{\Sigma}_k = \boldsymbol{I}$ (the identity matrix). The personal best position is updated according to (4) and the neighborhood best position is updated according to (5). Fig. 2 outlines the SMA-BBPSO.

Rule (14) for adapting the scale matrix associated with the kth particle was inspired by the general strategy adopted in [24], where the covariance matrix is directly altered by additive updates, called rank-one update of the covariance matrix. Beyer and Sendhoff [25] introduced a covariance matrix self-adaptation evolution strategy (CMSA-ES algorithm) as a simple and efficient variant of so-called covariance matrix adaptation (CMA) strategies [21], [22]. Beyer and Finck [26] introduced a CMSA-ES for solving a mixed linear/nonlinear constrained optimization problem arising in portfolio optimization.

The following results explain the effects and consequences when rules (12), (13) and (14) are adopted to update the position of a particle in the SMA-BBPSO, clarifying thereby the proposed approach.

Result 1. If $\lambda \sim Ga(\frac{\nu}{2}, \frac{\nu}{2})$ with $\nu > 0$, then

$$\boldsymbol{x}_k|\lambda \sim \boldsymbol{N}\left(\boldsymbol{\mu}_k, \frac{1}{\lambda}\boldsymbol{\Sigma}_k\right) \tag{15}$$

and $\boldsymbol{x}_k \sim \boldsymbol{t}(\nu, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ for all k. Note that a normal-distributed SMA-BBPSO can be obtained when $\lambda = 1$ almost surely, or when $\nu \to \infty$. Therefore, the parameter ν can be used to control the heaviness of the tails of the distribution of $\boldsymbol{x}_k$.

Result 1 follows immediately from the discussion about SMN distributions, presented in Appendix B.

Since $\boldsymbol{\Sigma}_k$ is a $D \times D$ symmetric positive definite matrix, it follows from the spectral decomposition of $\boldsymbol{\Sigma}_k$ that

$$\sqrt{\boldsymbol{\Sigma}_k} = \boldsymbol{P}_k\sqrt{\boldsymbol{\Theta}_k}\boldsymbol{P}_k^T \tag{16}$$

where $\boldsymbol{P}_k$ is an orthogonal matrix whose columms are the eingenvectors of $\boldsymbol{\Sigma}_k$ and

$$\sqrt{\boldsymbol{\Theta}_k} = \text{diag}(\theta_d^{1/2} : d = 1, \ldots, D) \tag{17}$$

where $\theta_1, \ldots, \theta_D$ are the eigenvalues associated. For purposes of implementation of the SMA-BBPSO, it is important to know the following result.

Result 2. Equation (12) is equivalent to

$$\boldsymbol{x}_k|\lambda \leftarrow \boldsymbol{\mu}_k + \lambda^{-1/2}\boldsymbol{P}_k\sqrt{\boldsymbol{\Theta}_k}\boldsymbol{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{18}$$

for all k.

Readers interested in technical details about Result 2 are referred to Appendix C, which presents a method to generate a random sample from a multivariate normal distribution, based on spectral decomposition of its covariance matrix. Result 2 informs that the proposed algorithm can be implemented simply knowing how to simulate from univariate probability distributions, i.e., $\boldsymbol{N}(\boldsymbol{0}, \boldsymbol{I})$ can be viewed as a vector of independent random variables and normally distributed with mean 0 and variance 1, and $\lambda \sim Ga(\frac{\nu}{2}, \frac{\nu}{2})$, $\nu > 0$. The spectral decomposition of a covariance matrix is $O(D^3)$ while a method to repair a covariance matrix due to numerical errors caused by the finite precision of computation capacity is $O(D)$.

When we use a swarm algorithm to optimize a function, each particle in the swarm is subject to a sequential learning process, where $\mathcal{L}_k^{(t)} := (\boldsymbol{n}_k^{(0)}, \boldsymbol{n}_k^{(1)}, \ldots, \boldsymbol{n}_k^{(t)})$ represents the accumulated learning of a particle until the time t. If the neighborhood system chosen admits that the particle itself belongs to its neighborhood (i.e., $k \in \mathcal{N}_k$ for all k), then $\mathcal{L}_k^{(t)}$ does not exclude personal information, since $\boldsymbol{n}_k^{(s)}$ can eventually be equal to $\boldsymbol{p}_k^{(s)}$ for some $s \leq t$. In addition, $\boldsymbol{n}_k^{(s)} \leq \boldsymbol{p}_l^{(s)}$ for all $l \in \mathcal{N}_k$ and $s \leq t$. Considering these facts, Result 3 informs that for the SMA-BBPSO, the scale matrix of a particle at time $t + 1$ uses its accumulated learning until t, without discarding any information so far.

Result 3. For the SMA-BBPSO algorithm, we have that

$$\boldsymbol{\Sigma}_k^{(t+1)} = (1-\beta)^{t+1}\boldsymbol{I} + \beta\sum_{s=0}^{t}(1-\beta)^{t-s}\boldsymbol{n}_k^{(s)}\boldsymbol{n}_k^{(s),T} \tag{19}$$

for all $t \geq 0$.

We can obtain (19) using mathematical induction and the recursive equation (14), that is

$$\boldsymbol{\Sigma}_k^{(t+1)} = (1-\beta)\boldsymbol{\Sigma}_k^{(t)} + \beta\boldsymbol{n}_k^{(t)}\boldsymbol{n}_k^{(t),T}. \tag{20}$$

Note that here we introduced an index t ($t = 0, 1, 2, \ldots$) to represent time. From (20), it follows that

$$\boldsymbol{\Sigma}_k^{(1)} = (1-\beta)\boldsymbol{I} + \beta\boldsymbol{n}_k^{(0)}\boldsymbol{n}_k^{(0),T}. \tag{21}$$

Assuming that

$$\boldsymbol{\Sigma}_k^{(t)} = (1-\beta)^{t}\boldsymbol{I} + \beta\sum_{s=0}^{t-1}(1-\beta)^{t-1-s}\boldsymbol{n}_k^{(s)}\boldsymbol{n}_k^{(s),T} \tag{22}$$

and using again (20), we have Result 3.

Result 4. Given the vector $\boldsymbol{n}_k$, the probability distribution that updates the position of the kth particle is shifted toward the distribution $\boldsymbol{N}(\mathbf{0}, \boldsymbol{n}_k\boldsymbol{n}_k^T)$ which is the multivariate normal distribution that generates the vector $\boldsymbol{n}_k$ with maximum likelihood among all D-dimensional normal distributions with mean $\mathbf{0}$.

Result 4 follows from Results of 1, 3, and 6 (Result 6 is presented in Appendix A). In fact, just use Result 6 noting that Results 1 and 3 imply that the update rule of the position of a particle can be written as

$$\boldsymbol{x}_k|\lambda = \boldsymbol{\mu}_k^{(t)} + A\cdot\boldsymbol{z} + B\cdot\boldsymbol{w}_k^{(t)} \tag{23}$$

where $\lambda \sim Ga\left(\frac{\nu}{2}, \frac{\nu}{2}\right)$, $\boldsymbol{\mu}_k^{(t)} = \boldsymbol{n}_k^{(t)} = \text{BEST}(\boldsymbol{p}_l^{(t)} : l \in \mathcal{N}_k)$

$$A = \left[\frac{(1-\beta)^{t+1}}{\lambda}\right]^{1/2} \qquad B = \left(\frac{\beta}{\lambda}\right)^{1/2} \tag{24}$$

$$\boldsymbol{z} = \sum_{d=1}^{D}\boldsymbol{e}_d z_d \tag{25}$$

$$\boldsymbol{w}_k^{(t)} = \sum_{s=0}^{t}(1-\beta)^{(t-s)/2}\boldsymbol{n}_k^{(s)}z_s'. \tag{26}$$

$z_1, \ldots, z_D, z_0', \ldots, z_t' \overset{iid}{\sim} N(0,1)$, and $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_D$ represent the standard basis for $\mathbb{R}^D$. Once again, we introduced here an index t to represent time.

In summary, Results 1, 2, 3, and 4 imply that in our approach, each particle selects new positions in the search space using a self-adaptive distribution with heavy tails and considering its accumulated learning until the current iteration.

IV. Experimental Results

A. Benchmark Functions

We consider a suite of benchmark functions to investigate the performance of the proposed algorithm. The list of these functions along with their references is shown in Table I. According to their properties, these functions are divided into two classes: f_{01-04} are unimodal functions and f_{05-15} are multimodal functions with many local minima. These functions have been widely used in many works, and [8], [28]–[30] provide a detailed description of each problem. All functions used in our experiments have to be minimized. We use a nonuniform and asymmetric initialization that does not contain the global optimum.

TABLE I
Benchmark Functions Used in Our Experiments

Problem	Name	References
f_{01}	Sphere function	[8], [28], [30]
f_{02}	Schwefel's problem 1.2	[8], [30]
f_{03}	Rosenbrock's function	[8], [28], [30]
f_{04}	Schwefel's problem 2.22	[8], [30]
f_{05}	Schwefel's problem 2.26	[8], [28], [30]
f_{06}	Rastrigin's function	[8], [28], [30]
f_{07}	Ackley's function	[8], [28], [30]
f_{08}	Griewank's function	[8], [28], [30]
f_{09}	Weierstrass function	[28]
f_{10}	Rotated Rastrigin's function	[28]
f_{11}	Rotated Ackley's function	[28]
f_{12}	Rotated Griewank's function	[28]
f_{13}	Rotated Weierstrass function	[28]
f_{14}	Shifted Rastrigin's function	[29]
f_{15}	Shifted rotated Griewank's function	[29]

TABLE II
Values of β Used by the SMA-BBPSO in Our Experiments

f	f_{01}	f_{02}	f_{03}	f_{04}	f_{05}	f_{06}	f_{07}	f_{08}
β	0.30	0.20	0.05	0.20	0.05	0.05	0.10	0.05
f	f_{09}	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	-
β	0.05	0.05	0.10	0.05	0.05	0.01	0.01	-

B. Experimental Setup

We carry out a set of experiments to compare empirically nine algorithms on the 15 test functions, including the proposed algorithm. The algorithms and parameters settings are listed as follows:

1) PSO($S, \mathcal{N}, c_1, c_2$) [6]: $S = 30$, $c_1 = c_2 = 2.05$, and $\mathcal{N}$ = Lbest ring topology with $|\mathcal{N}_k| = 3$;
2) BBPSO($S, \mathcal{N}$) [7]: $S = 30$ and $\mathcal{N}$ such as PSO;
3) BBPSOwJ($S, \mathcal{N}, \alpha, p_J$) [13]: $S = 30, \alpha = 0.7, p_J = 0.01$, and $\mathcal{N}$ such as in PSO;
4) FIPS($S, \mathcal{N}, \varphi$) [18], [19]: $S = 30, \varphi = 4.1$, and $\mathcal{N}$ such as in PSO;
5) comprehensive learning PSO (CLPSO) [28];
6) Lévy BBPSO($S, \mathcal{N}, \alpha$) [11]: $S = 30, \alpha = 1.4$, and $\mathcal{N}$ such as in PSO;
7) (μ/ρ,λ)-ES [23]: $\mu = 30$, $\rho = 7$, $\lambda = \rho\mu$, $\tau_0 = (\sqrt{2D})^{-1}$, and $\tau = (\sqrt{2\sqrt{D}})^{-1}$;
8) (μ/ρ,λ)-CMSA-ES [25], [26]: μ, ρ, λ, and τ_0 such as in ES, and $\tau_c = 1 + D(D+1)/(2\mu)$;

TABLE III
COMPARISONS BETWEEN PSO, BBPSO, BBPSOWJ, FIPS, AND SMA-BBPSO

	PSO [6]		BBPSO [7]		BBPSOwJ [13]		FIPS [18], [19]		SMA-BBPSO	
Problem	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
f_{01}	1.13E-10	7.94E-11	1.58E-06	1.75E-06	4.34E-03	2.37E-02	1.03E-04	6.95E-05	**2.42E-154**	2.71E-154
f_{02}	1.41E+03	7.93E+02	5.11E+03	1.90E+03	2.14E+03	2.62E+03	1.44E+03	1.32E+03	**5.24E-153**	5.13E-153
f_{03}	5.26E+01	4.20E+01	9.47E+01	8.55E+01	6.50E+01	4.22E+01	4.29E+01	4.01E+01	**2.87E+01**	1.37E-02
f_{04}	4.04E-04	1.33E-04	9.67E-04	4.44E-04	4.25E-03	8.31E-03	2.67E-04	1.08E-04	**2.32E-84**	2.17E-84
f_{05}	3.19E+03	1.18E+02	2.51E+03	2.88E+02	**3.81E+01**	5.79E+01	3.36E+03	2.48E+02	1.61E+02	4.14E+01
f_{06}	8.36E+01	1.98E+01	8.28E+01	1.83E+01	1.11E+01	3.45E+00	7.78E+01	1.63E+01	**0.00E+00**	0.00E+00
f_{07}	1.90E+01	3.59E+00	5.08E-01	8.70E-01	1.54E-01	3.55E-01	6.86E-01	1.04E+00	**2.22E-15**	1.81E-15
f_{08}	2.10E-03	4.16E-03	5.23E-03	7.96E-03	3.05E-02	2.84E-02	5.53E-02	8.15E-02	**0.00E+00**	0.00E+00
f_{09}	3.19E-03	4.95E-04	5.23E-03	3.18E-03	5.21E-01	1.22E-01	8.01E-02	7.05E-02	**4.80E-12**	2.39E-12
f_{10}	1.60E+02	4.31E+01	1.88E+02	3.44E+01	1.61E+02	3.71E+01	1.10E+02	4.61E+01	**0.00E+00**	0.00E+00
f_{11}	4.24E+00	6.99E+00	1.91E+00	4.86E-01	4.15E+00	9.56E-01	1.64E+00	8.12E-01	**1.98E-15**	1.79E-15
f_{12}	4.20E-03	6.84E-03	5.69E-03	7.01E-03	1.97E-02	1.88E-02	6.99E-02	9.03E-02	**0.00E+00**	0.00E+00
f_{13}	2.30E+01	1.48E+00	2.61E+01	1.88E+00	2.86E+01	2.34E+00	1.79E+01	2.97E+00	**7.28E-12**	2.16E-12
f_{14}	6.70E+01	1.54E+01	6.11E+01	1.45E+01	**1.02E+01**	2.86E+00	1.26E+02	3.19E+01	4.64E+01	1.31E+01
f_{15}	3.21E-02	1.06E-01	6.62E-03	5.20E-03	1.69E-02	1.94E-02	1.15E-01	1.00E-01	**5.19E-03**	7.63E-03
$W/L/T$	15/0/0		15/0/0		13/2/0		15/0/0		-	

TABLE IV
COMPARISONS BETWEEN CLPSO, LÉVY BBPSO, ES, CMSA-ES, AND SMA-BBPSO

	†CLPSO [28]		Lévy BBPSO [11]		ES [23]		CMSA-ES [25], [26]		SMA-BBPSO	
Problem	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
f_{01}	4.46E-14	1.73E-14	2.23E-10	4.59E-10	2.49E-104	7.62E-104	**6.15E-156**	7.33E-156	2.42E-154	2.71E-154
f_{02}	NA	NA	1.04E+00	7.75E-01	1.62E+03	5.12E+02	5.13E-147	1.16E-146	**5.24E-153**	5.13E-153
f_{03}	2.10E+01	2.98E+00	6.01E+02	2.29E+02	1.98E+01	9.76E+00	**9.59E+00**	1.26E+00	2.87E+01	1.37E-02
f_{04}	NA	NA	5.98E-02	2.58E-02	3.88E+00	2.87E+00	7.68E-70	2.94E-70	**2.32E-84**	2.17E-84
f_{05}	**0.00E+00**	8.79E-13	1.46e+03	1.78E+02	8.87E+03	3.28E+02	1.03E+04	4.21E+02	1.61E+02	4.14E+01
f_{06}	4.85E-10	3.63E-10	3.19E+01	7.71E+00	3.64E+01	7.96E+00	9.37E+00	2.19E+00	**0.00E+00**	0.00E+00
f_{07}	**0.00E+00**	0.00E+00	1.27E+00	5.43E-01	4.00E-15	0.00E+00	4.00E-15	0.00E+00	2.22E-15	1.81E-15
f_{08}	3.14E-10	4.64E-10	6.12E-01	1.72E-01	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f_{09}	3.45E-07	1.94E-07	4.93E-01	1.01E-01	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	4.80E-12	2.39E-12
f_{10}	3.46E+01	4.59E+00	1.80E+02	1.05E+01	9.63E+01	1.96E+01	8.16E+00	2.34E+00	**0.00E+00**	0.00E+00
f_{11}	3.43E-04	1.91E-04	3.10E+00	4.73E-01	4.00E-15	0.00E+00	4.00E-15	0.00E+00	**1.98E-15**	1.79E-15
f_{12}	7.04E-10	1.25E-11	6.57E-01	1.33E-01	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
f_{13}	3.07E+00	1.61E+00	1.95E+01	3.17E+00	1.27E-01	3.46E-01	6.86E-04	1.90E-03	**7.28E-12**	2.16E-12
f_{14}	NA	NA	3.82E+01	6.18E+00	1.21E+02	2.34E+01	**2.04E+01**	5.53E+00	4.64E+01	1.31E+01
f_{15}	NA	NA	7.63E-01	1.35E-01	1.06E+00	5.74E+00	**4.93E-04**	1.88E-03	5.19E-03	7.63E-03
$W/L/T$	8/3/0		14/1/0		11/2/2		8/5/2		-	

† For this algorithm, we report here the mean and the standard deviation available in the original paper [28]. NA means not available.

9) SMA-BBPSO($S, \mathcal{N}, \beta, m_{\max}$): $S = 30$, $\mathcal{N}$ such as in PSO, and $m_{\max} = 5$. Table II shows the value of β for each test function used in our experiments.

For each pair test function/algorithm, a sequence of empirical errors was obtained from 30 independent runs of 1500 iterations, where in each run the algorithm was used to optimize the test function of dimension $D = 30$. The empirical error is defined by $|f(\boldsymbol{x}_{best}) - f(\boldsymbol{x}_{\min})|$, where f is the test function, $\boldsymbol{x}_{best}$ is the best solution obtained by the algorithm into a particular run, and $\boldsymbol{x}_{\min}$ is the global optimum of the test function. Summary statistics were calculated to each sequence of empirical errors.

C. Results and Discussions

Tables III and IV show the mean and the standard deviation of the errors of each algorithm on 15 test functions considered in our experiments. The best results are shown in boldface. It is important to observe that for the specific case of the CLPSO algorithm, we report in Table IV the mean and the standard deviation available in the original paper [28], where the algorithm was presented. All the other results in Tables III and IV were derived from computational simulations realized by the authors considering the experimental setup described in Section IV-B and the test functions listed in Table I.

A quick and easy test to compare the performance of two algorithms in a multiproblem analysis is to count the number

TABLE V
RESULTS OF THE WILCOXON SIGNED-RANKS TEST FOR PAIRWISE COMPARISONS OF ALGORITHMS ON EACH BENCHMARK FUNCTION

	Comparison: SMA-BBPSO versus						
Problem	PSO	BBPSO	BBPSOwJ	FIPS	Lévy BBPSO	ES	CMSA-ES
f_{01}	+	+	+	+	+	+	−
f_{02}	+	+	+	+	+	+	+
f_{03}	+	+	+	=	+	−	−
f_{04}	+	+	+	+	+	+	+
f_{05}	+	+	−	+	+	+	+
f_{06}	+	+	+	+	+	+	+
f_{07}	+	+	+	+	+	+	+
f_{08}	+	+	+	+	+	=	=
f_{09}	+	+	+	+	+	−	−
f_{10}	+	+	+	+	+	+	+
f_{11}	+	+	+	+	+	+	+
f_{12}	+	+	+	+	+	=	=
f_{13}	+	+	+	+	+	+	+
f_{14}	+	+	−	+	−	+	−
f_{15}	+	=	+	+	+	+	−

(+) means that SMA-BBPSO is significantly better than its competitor algorithm.
(=) means that SMA-BBPSO does not differ significantly from its competitor algorithm.
(−) means that the competitor algorithm is significantly better than SMA-BBPSO.

TABLE VI
RESULTS OF WILCOXON SIGNED-RANKS TEST FOR PAIRWISE COMPARISONS OF ALGORITHMS ON THE SET OF BENCHMARK FUNCTIONS

Comparison	V	p-value	Conclusion
SMA-BBPSO versus PSO	0	6.104E-06	SMA-BBPSO shows a significant improvement over PSO
SMA-BBPSO versus BBPSO	0	6.104E-06	SMA-BBPSO shows a significant improvement over BBPSO
SMA-BBPSO versus BBPSOwJ	24	0.04126	SMA-BBPSO shows a significant improvement over BBPSOwJ
SMA-BBPSO versus FIPS	0	6.104E-06	SMA-BBPSO shows a significant improvement over FIPS
SMA-BBPSO versus Lévy BBPSO	10	0.002625	SMA-BBPSO shows a significant improvement over Lévy BBPSO
SMA-BBPSO versus ES	12	0.01709	SMA-BBPSO shows a significant improvement over ES
SMA-BBPSO versus CMSA-ES	38	0.6355	there is no significant difference

An algorithm is significantly better than another if the p-value is less than $\alpha = 5\%$ (α is the significance level of the test).

of cases on which an algorithm is the winner. If two algorithms are equivalent (as assumed under the null hypotheses), then each algorithm should win on approximately $N/2$ out of N problems investigated. Since tied matches support the null hypothesis, they should not be discounted when applying this test, but equally divided between the two algorithms. If there is an odd number of tied matches, one of them should be ignored. This procedure is known as the Sign test [31]–[33]. The Sign test can provide a first snapshot about the pairwise statistical comparisons of algorithms. Comparisons between SMA-BBPSO and other algorithms are summarized as $W/L/T$ in the last row of Tables III and IV, which means that SMA-BBPSO wins in W problems, loses in L problems, and ties in T problems. In our experimental framework, the critical value for the two-tailed Sign test (with a significance level of 5%) for paired comparisons of algorithms is 12, because we use 15 test problems. It means that an algorithm is significantly better than another if it performs better on at least 12 out of 15 problems. Our experimental results show that SMA-BBPSO is significantly better than PSO, BBPSO, BBPSOwJ, FIPS, Lévy BBPSO, and ES with a level of significance of 5%. However, based on the Sign test, there is no experimental evidence to assert that SMA-BBPSO presents a statistically significant improvement compared to CMSA-ES, that is, there is no significant difference between these algorithms.

We also develop a single-problem analysis. A single-problem analysis deals with results obtained over several runs of the algorithms over a given problem. The results of this analysis are presented in Table V. We use the Wilcoxon signed-rank test [31]–[33] for pairwise comparisons of algorithms on each test function. We tested whether the results obtained by SMA-BBPSO are statistically different from those obtained by its competitor algorithm on each test function. The symbol (+) denotes that SMA-BBPSO is significantly better than its competitor algorithm. The symbol (=) denotes that SMA-BBPSO does not differ significantly from its competitor algorithm. Finally, the symbol (−) denotes that the competitor algorithm is significantly better than SMA-BBPSO; this means

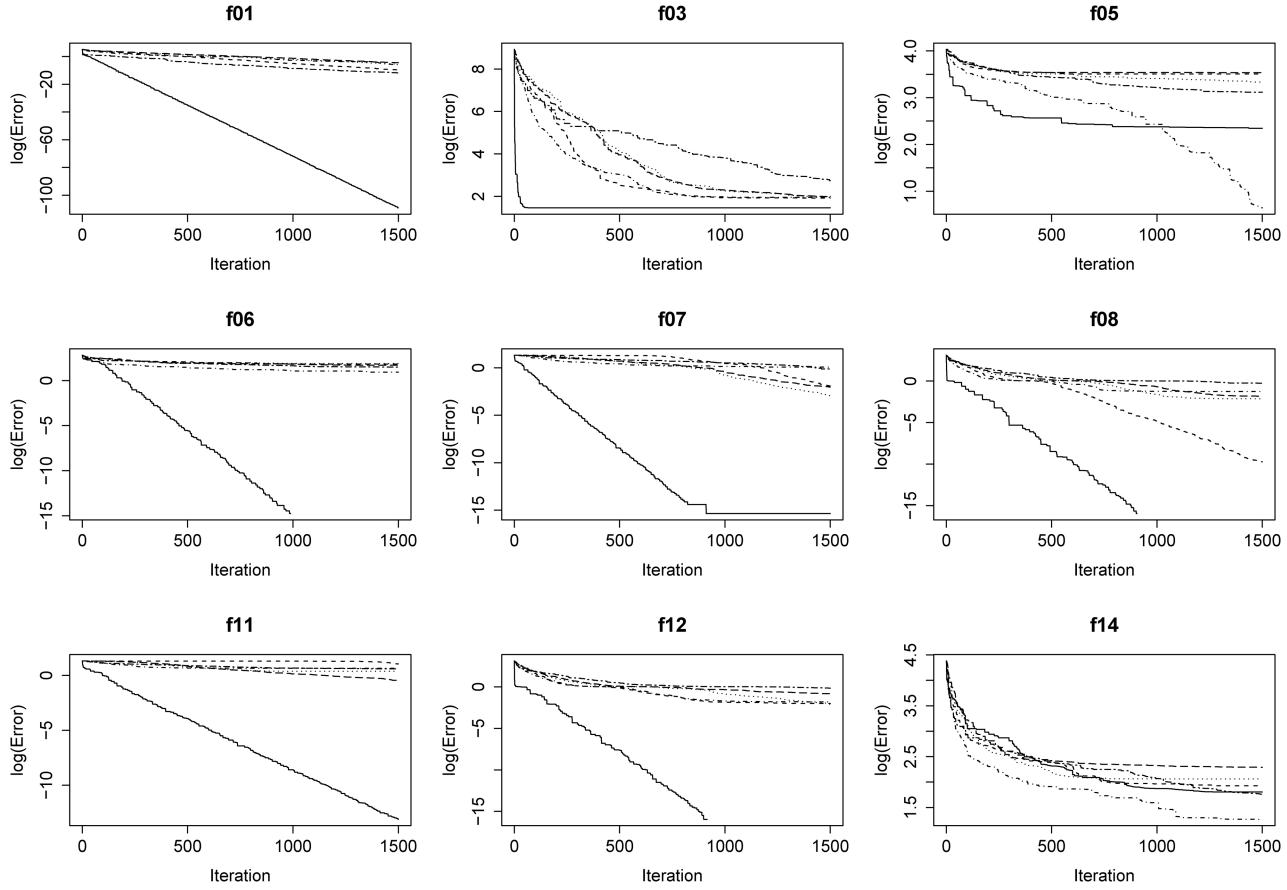

Fig. 3. Convergence plots for SMA-BBPSO (solid line), PSO (dashed line), BBPSO (dotted line), BBPSOwJ (dot-dash line), FIPS (long-dash line), and Lévy BBPSO (two-dash line) over nine test functions. Error = $|f(\boldsymbol{x}_{best}) - f(\boldsymbol{x}_{\min})|$ and log(Error) = $\log_{10}$(Error).

that the mean error of the SMA-BBPSO is greater than the mean error of its competitor. In all comparisons, we use a level of significance of 5%.

Once again, a multiproblem analysis was developed using the Wilcoxon signed-rank test for pairwise comparisons of algorithms on the set of test functions. The results of this analysis are summarized in Table VI. The results of the Wilcoxon signed-rank test are in agreement with the results of the Sign test previously obtained. SMA-BBPSO shows a significant improvement over PSO, BBPSO, BBPSOwJ, FIPS, Lévy BBPSO, and ES with a level of significance of 5%. However, based on Wilcoxon signed-rank test, there is no significant difference between SMA-BBPSO and CMSA-ES.

Fig. 3 shows convergence plots for SMA-BBPSO, PSO, BBPSO, BBPSOwJ, FIPS, and Lévy BBPSO over nine test functions.

V. Conclusion

In this paper, a variant of the BBPSO was introduced and named by us as BBPSO with scale matrix adaptation, SMA-BBPSO for short. A theoretical analysis was presented to explain how the SMA-BBPSO works, emphasizing the differences between the proposed approach and those of other swarm algorithms. Each particle in SMA-BBPSO selects new positions in the search space using a self-adaptive distribution with heavy tails and considering its accumulated learning until the current iteration. This strategy induces exploration and exploitation at any stage of the search process.

The experimental results obtained from an empirical study revealed the suitability of the proposed approach in terms of effectiveness to find good solutions for all benchmark problems investigated. Nonparametric statistical tests for pairwise comparisons of algorithms on a set of benchmark functions indicated that SMA-BBPSO shows a statistically significant improvement compared with other swarm algorithms.

SMA-BBPSO is simple and easy to implement like other swarm algorithms. Another feature of the SMA-BBPSO is that it is also very flexible, allowing including other distributions with heavy tails besides the t-distribution. From a suitable choice of the mixing density, a rich class of continuous, symmetric and unimodal distributions can be described. Furthermore, the particular case of the normal distribution can be recovered when $\lambda = 1$ almost surely, or when $\nu \to \infty$ if the mixing density is given by $Ga(\frac{\nu}{2}, \frac{\nu}{2})$. Finally, our approach can also be extended for solving constrained optimization problems with many potential applications in science and engineering.

APPENDIX A
PROBABILITY DENSITIES AND USEFUL RESULTS

This list includes the probability densities that were used in this paper.

1) A random variable x has a gamma distribution with parameters α and β, denoted by $x \sim Ga(\alpha, \beta)$, if its pdf is given by

$$Ga(x|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp(-\beta x) \tag{27}$$

where $x > 0$, $\alpha, \beta > 0$, and $\Gamma(\cdot)$ is the well known gamma function. This distribution has mean α/β and variance α/β^2.

2) A random variable $\boldsymbol{x} = (x_d : d = 1, \ldots, D)^T$ has a multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and variance-covariance matrix $\boldsymbol{\Sigma}$, denoted by $\boldsymbol{x} \sim \boldsymbol{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, if its pdf is given by

$$\boldsymbol{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}\Delta^2\right) \tag{28}$$

where $\boldsymbol{x} \in \mathbb{R}^D$, $\boldsymbol{\mu} \in \mathbb{R}^D$, $\boldsymbol{\Sigma}$ is a $D \times D$ symmetric positive definite matrix, and Δ^2 is the squared Mahalanobis distance given by $\Delta^2 = (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})$. When $D = 1$, we have the well known univariate normal distribution $N(\mu, \sigma^2)$ with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 > 0$. When $\mu = 0$ and $\sigma^2 = 1$, the distribution is referred to as standard normal distribution.

3) A random variable $\boldsymbol{x}$ has a multivariate t-distribution with location $\boldsymbol{\mu}$, scale parameter $\boldsymbol{\Sigma}$ and ν degrees of freedom, denoted by $\boldsymbol{x} \sim \boldsymbol{t}(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$, if its pdf is given by

$$\boldsymbol{t}(\boldsymbol{x}|\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{C_{\nu,D}}{|\boldsymbol{\Sigma}|^{1/2}} \left(1 + \frac{1}{\nu}\Delta^2\right)^{-\frac{(\nu+D)}{2}} \tag{29}$$

where $\boldsymbol{x} \in \mathbb{R}^D$, $\boldsymbol{\mu} \in \mathbb{R}^D$, $\boldsymbol{\Sigma}$ is a $D \times D$ symmetric positive definite matrix, and

$$C_{\nu,D} = \frac{\Gamma((\nu + D)/2)}{\Gamma(\nu/2)(\nu\pi)^{D/2}}. \tag{30}$$

This distribution has mean $\boldsymbol{\mu}$, when $\nu > 1$, and variance-covariance matrix $[\nu/(\nu - 2)]\boldsymbol{\Sigma}$, when $\nu > 2$.

Result 5 (see [21] and [22]): Let z be a random variable with the standard normal distribution and let us consider a vector $\boldsymbol{y}$ in $\mathbb{R}^D$. Then $\boldsymbol{u} = \boldsymbol{y}z \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{y}\boldsymbol{y}^T)$. In addition, the distribution of $\boldsymbol{u}$ generates $\boldsymbol{y}$ with maximum likelihood among all D-dimensional normal distributions with mean $\boldsymbol{0}$.

Result 6 (see [21] and [22]): Let $z_1, \ldots, z_t$ be a sequence of random variables independent and normally distributed with mean 0 and variance 1. If the vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t \in \mathbb{R}^D$ span $\mathbb{R}^D$ ($t \geq D$), then $\boldsymbol{u}_1 = \boldsymbol{y}_1 z_1, \ldots, \boldsymbol{u}_t = \boldsymbol{y}_t z_t$ are independent random variables such that

$$\boldsymbol{w}_t = \sum_{s=1}^{t} \boldsymbol{u}_s = \sum_{s=1}^{t} \boldsymbol{y}_s z_s \sim \boldsymbol{N}\left(\boldsymbol{0}, \sum_{s=1}^{t} \boldsymbol{y}_s \boldsymbol{y}_s^T\right). \tag{31}$$

Given the vector $\boldsymbol{y}_t$, the distribution of $\boldsymbol{w}_{t-1}$ is shifted toward the distribution $\boldsymbol{N}(\boldsymbol{0}, \boldsymbol{y}_t \boldsymbol{y}_t^T)$, which is the multivariate normal distribution that generates the vector $\boldsymbol{y}_t$ with maximum likelihood among all D-dimensional normal distributions with mean $\boldsymbol{0}$.

To complete Appendix A, we want to show that

$$\mathcal{I}(\boldsymbol{x}) = \int_0^\infty \boldsymbol{N}(\boldsymbol{x}|\boldsymbol{\mu}, \lambda^{-1}\boldsymbol{\Sigma}) Ga(\lambda|\alpha, \beta) d\lambda \tag{32}$$

$$= \boldsymbol{t}(\boldsymbol{x}|2\alpha, \boldsymbol{\mu}, \beta\alpha^{-1}\boldsymbol{\Sigma}) \tag{33}$$

where $\alpha, \beta > 0$, $\boldsymbol{\mu} \in \mathbb{R}^D$, and $\boldsymbol{\Sigma}$ is a $D \times D$ symmetric positive definite matrix (this result will be used in Appendix B). Observe that

$$\mathcal{I}(\boldsymbol{x}) = \frac{\beta^\alpha |\boldsymbol{\Sigma}|^{-1/2}}{\Gamma(\alpha) 2\pi^{D/2}} \int_0^\infty \lambda^{(\alpha + \frac{D}{2}) - 1} e^{-(\beta + \frac{\Delta^2}{2})\lambda} d\lambda. \tag{34}$$

Defining $u = B\lambda$, where $B = \beta + (\Delta^2/2)$, we see that

$$\mathcal{I}(\boldsymbol{x}) = \frac{\beta^\alpha |\boldsymbol{\Sigma}|^{-1/2}}{\Gamma(\alpha) 2\pi^{D/2} B^{\alpha + (D/2)}} \int_0^\infty u^{(\alpha + \frac{D}{2}) - 1} e^{-u} du. \tag{35}$$

As $\Gamma(z) = \int_0^\infty u^{z-1} e^{-u} du$, it follows that

$$\begin{aligned} \mathcal{I}(\boldsymbol{x}) &= \frac{\Gamma((2\alpha + D)/2)\beta^\alpha}{\Gamma(2\alpha/2) 2\pi^{D/2} |\boldsymbol{\Sigma}|^{1/2}} \cdot B^{-\frac{(2\alpha+D)}{2}} \\ &= \frac{C_{2\alpha,D}}{|\beta\alpha^{-1}\boldsymbol{\Sigma}|^{1/2}} \left(1 + \frac{\Delta^2}{2\alpha}\right)^{-\frac{(2\alpha+D)}{2}} \\ &= \boldsymbol{t}(\boldsymbol{x}|2\alpha, \boldsymbol{\mu}, \beta\alpha^{-1}\boldsymbol{\Sigma}). \end{aligned} \tag{36}$$

We conclude that a distribution $\boldsymbol{t}(2\alpha, \boldsymbol{\mu}, \beta\alpha^{-1}\boldsymbol{\Sigma})$ is equivalent to the following hierarchical form with two levels:

$$\boldsymbol{x}|\lambda \sim \boldsymbol{N}\left(\boldsymbol{\mu}, \frac{1}{\lambda}\boldsymbol{\Sigma}\right) \tag{37}$$

$$\lambda \sim Ga(\alpha, \beta). \tag{38}$$

APPENDIX B
SCALE MIXTURES OF NORMAL DISTRIBUTIONS

Scale mixtures of normal distributions represent a very important role in statistical modelling [15], [16]. They are derived by mixing a normally distributed random variable $\boldsymbol{y}$ with a nonnegative random variable λ as follows:

$$\boldsymbol{x}|\lambda = \boldsymbol{\mu} + \kappa^{1/2}(\lambda)\boldsymbol{y} \tag{39}$$

where $\boldsymbol{\mu}$ is a location parameter, $\kappa(\cdot)$ is a positive weight function, and $\boldsymbol{y} \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$. The random variable λ has a pdf $h(\lambda|\boldsymbol{\theta})$, independent of $\boldsymbol{y}$, where $\boldsymbol{\theta}$ is a parameter vector indexing the distribution of λ. In this paper, we restrict our attention to the case in that $\kappa(\lambda) = \lambda^{-1}$. Thus, given λ

$$\boldsymbol{x}|\lambda \sim \boldsymbol{N}(\boldsymbol{\mu}, \lambda^{-1}\boldsymbol{\Sigma}) \tag{40}$$

and the pdf of $\boldsymbol{x}$ is given by

$$p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \int_0^\infty \boldsymbol{N}(\boldsymbol{x}|\boldsymbol{\mu}, \lambda^{-1}\boldsymbol{\Sigma}) h(\lambda|\boldsymbol{\theta}) d\lambda. \tag{41}$$

From a suitable choice of the mixing density $h(\cdot|\boldsymbol{\theta})$, a rich class of continuous symmetric and unimodal distribution can be described by the density given in (41) that can readily accommodate a thicker-than-normal process. Note that we retrieve the normal distribution when $\lambda = 1$ almost surely.

When $h(\lambda|\boldsymbol{\theta}) = Ga(\lambda|\nu/2, \nu/2)$, we have that $\boldsymbol{x} \sim \boldsymbol{t}(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$. Thus, $\boldsymbol{x}$ has a multivariate t-distribution with location $\boldsymbol{\mu}$, scale parameter $\boldsymbol{\Sigma}$ and ν degrees of freedom, and this distribution is equivalent to the following hierarchical form:

$$\boldsymbol{x}|\lambda \sim \boldsymbol{N}\left(\boldsymbol{\mu}, \frac{1}{\lambda}\boldsymbol{\Sigma}\right) \qquad \lambda \sim Ga\left(\frac{\nu}{2}, \frac{\nu}{2}\right). \tag{42}$$

This result is a particular case of integral (32) when $\alpha = \beta = \nu/2$.

APPENDIX C
METHOD FOR GENERATING $\boldsymbol{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ SAMPLES

Let $\boldsymbol{A}$ be a $D \times D$ symmetric matrix. Then $\boldsymbol{A}$ has D pairs of eigenvalues and eigenvectors, namely $(\theta_1, \boldsymbol{v}_1), \ldots, (\theta_D, \boldsymbol{v}_D)$. The eigenvectors can be chosen to satisfy $\boldsymbol{v}_d^T \boldsymbol{v}_d = 1$ for $d = 1, \ldots, D$, and be mutually perpendicular. The eigenvectors are unique unless two or more eigenvalues are equal. The eigenvalues are real.

If $\boldsymbol{A}$ is a $D \times D$ symmetric matrix with eigenvalues $\theta_1, \ldots, \theta_D$ and normalized eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_D$, then $\boldsymbol{A}$ can be expressed as

$$\boldsymbol{A} = \sum_{d=1}^{D} \theta_d \boldsymbol{v}_d \boldsymbol{v}_d^T = \boldsymbol{P}\boldsymbol{\Theta}\boldsymbol{P}^T \tag{43}$$

where $\boldsymbol{\Theta} = \text{diag}(\theta_d : d = 1, \ldots, D)$ and $\boldsymbol{P}$ is an orthogonal matrix whose columms are the eigenvectors of $\boldsymbol{A}$. Thus, $\boldsymbol{P}\boldsymbol{P}^T = \boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{I}$ (the identity matrix) or $\boldsymbol{P}^T = \boldsymbol{P}^{-1}$.

The result in (43) is often called the spectral decomposition of $\boldsymbol{A}$. Analysis of eigenvalues and eigenvectors is widely used in the field of multivariate analysis; in particular, it also plays a crucial role in principal components analysis, which is used in the field of machine learning for the purpose of feature extraction and reducing the dimensionality of data without significant loss of information.

Let $\boldsymbol{A}$ be a $D \times D$ symmetric matrix. $\boldsymbol{A}$ is positive definite if $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} > 0$ for all vectors $\boldsymbol{x} \neq \boldsymbol{0}$. Using the spectral decomposition, it is possible to show that a $D \times D$ symmetric matrix $\boldsymbol{A}$ is a positive definite matrix if and only if every eigenvalue of $\boldsymbol{A}$ is positive. Finally, if $\boldsymbol{A}$ is a $D \times D$ symmetric positive definite matrix, then

$$\sqrt{\boldsymbol{A}} = \boldsymbol{P}\sqrt{\boldsymbol{\Theta}}\boldsymbol{P}^T \tag{44}$$

where $\sqrt{\boldsymbol{\Theta}} = \text{diag}(\theta_d^{1/2} : d = 1, \ldots, D)$. $\sqrt{\boldsymbol{A}}$ represents the square-root matrix of $\boldsymbol{A}$. Observe that $\sqrt{\boldsymbol{A}}$ is symmetric and $\boldsymbol{A} = \sqrt{\boldsymbol{A}}\sqrt{\boldsymbol{A}} = \sqrt{\boldsymbol{A}}\sqrt{\boldsymbol{A}}^T$.

If $\boldsymbol{B}$ is a $D \times D$ matrix, $\boldsymbol{\mu}$ is a D-dimensional vector and $\boldsymbol{y} \sim \boldsymbol{N}(\boldsymbol{m}, \boldsymbol{C})$ where $\boldsymbol{C}$ is a $D \times D$ symmetric positive definite matrix, then

$$\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{B}\boldsymbol{y} \sim \boldsymbol{N}(\boldsymbol{\mu} + \boldsymbol{B}\boldsymbol{m}, \boldsymbol{B}\boldsymbol{C}\boldsymbol{B}^T). \tag{45}$$

In particular, if $\boldsymbol{y} \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{I})$ and $\boldsymbol{B} = \sqrt{\boldsymbol{\Sigma}}$ where $\boldsymbol{\Sigma}$ is a $D \times D$ symmetric positive definite matrix, then

$$\boldsymbol{x} = \boldsymbol{\mu} + \sqrt{\boldsymbol{\Sigma}}\boldsymbol{y} \sim \boldsymbol{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{46}$$

Thus, to generate a random sample from the $\boldsymbol{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ distribution, take the following steps.

1) Generate $\boldsymbol{y} = (y_1, \ldots, y_D)^T \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{I})$ where $y_1, \ldots, y_D \overset{idd}{\sim} N(0, 1)$.
2) Compute $\sqrt{\boldsymbol{\Sigma}} = \boldsymbol{P}\sqrt{\boldsymbol{\Theta}}\boldsymbol{P}^T$.
3) Apply transformation (46)

$$\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{P}\sqrt{\boldsymbol{\Theta}}\boldsymbol{P}^T\boldsymbol{y} = \boldsymbol{\mu} + \boldsymbol{P}\sqrt{\boldsymbol{\Theta}}\boldsymbol{y}. \tag{47}$$

4) Deliver the vector $\boldsymbol{x}$ that is distributed as $\boldsymbol{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1941–1948.
[2] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proc. IEEE Swarm Intell. Symp.*, 2007, pp. 120–127.
[3] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, 2007.
[4] L. N. de Castro, "Fundamentals of natural computing: An overview," *Phys. Life Rev.*, vol. 4, no. 1, pp. 1–36, 2007.
[5] R. Eberhart and Y. Shi, *Computational Intelligence: Concepts to Implementations*. Burlington, MA, USA: Morgan Kaufmann, 2007.
[6] M. Clerc and J. Kennedy, "The particle swarm: Explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.
[7] J. Kennedy, "Bare bones particle swarms," in *Proc. IEEE Swarm Intell. Symp.*, 2003, pp. 80–87.
[8] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.
[9] C.-Y. Lee and X. Yao, "Evolutionary programming using mutations based on the Lévy probability distribution," *IEEE Trans. Evol. Comput.*, vol. 8, no. 1, pp. 1–13, Feb. 2004.
[10] R. Mantegna, "Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes," *Phys. Rev. E*, vol. 49, no. 5, pp. 4677–4683, 1994.
[11] T. Richer and T. Blackwell, "The Lévy particle swarm," in *Proc. IEEE Congr. Evol. Comput.*, 2006, pp. 3150–3157.
[12] R. Krohling and E. Mendel, "Bare bones particle swarm optimization with Gaussian or Cauchy jumps," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 3285–3291.
[13] T. Blackwell, "A study of collapse in bare bones particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 3, pp. 354–372, Jun. 2012.
[14] H.-I. Hsieh and T.-S. Lee, "A modified algorithm of bare bones particle swarm optimization," *Int. J. Comput. Sci. Issues*, vol. 7, no. 6, pp. 12–17, Nov. 2010.
[15] D. Andrews and C. Mallows, "Scale mixtures of normal distributions," *J. Roy. Statist. Soc. B (Methodol.)*, vol. 36, no. 1, pp. 99–102, 1974.
[16] S. Choy and J. Chan, "Scale mixtures of distributions in statistical modelling," *Australian New Zealand J. Statist.*, vol. 50, no. 2, pp. 135–146, 2008.
[17] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Congr. Evol. Comput.*, 1998, pp. 69–73.
[18] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 204–210, Jun. 2004.
[19] J. Kennedy and R. Mendes, "Neighborhood topologies in fully informed and best-of-neighborhood particle swarms," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 36, no. 4, pp. 515–519, Jul. 2006.
[20] J. Kennedy, "Probability and dynamics in the particle swarm," in *Proc. IEEE Congr. Evol. Comput.*, 2004, pp. 340–347.
[21] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," in *Proc. IEEE Congr. Evol. Comput.*, 1996, pp. 312–317.
[22] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.
[23] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies: A comprehensive introduction," *Natural Comput.*, vol. 1, no. 1, pp. 3–52, May 2002.
[24] C. Igel, T. Suttorp, and N. Hansen, "A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies," in *Proc. GECCO*, 2006, pp. 453–460.
[25] H.-G. Beyer and B. Sendhoff, "Covariance matrix adaptation revisited: The CMSA evolution strategy," in *Proc. PPSN X*, 2008, pp. 123–132.

[26] H.-G. Beyer and S. Finck, "On the design of constraint covariance matrix self-adaptation evolution strategies including a cardinality constraint," *IEEE Trans. Evol. Comput.*, vol. 16, no. 4, pp. 578–596, Aug. 2012.
[27] S. Ghosh, S. Das, S. Roy, S. Islan, and P. Suganthan, "Differential covariance matrix adaptation evolutionary algorithm," *Inform. Sci.*, vol. 182, no. 1, pp. 199–219, Jan. 2012.
[28] J. Liang, A. Qin, P. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.
[29] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y.-P. Chen, A. Auger, *et al.*, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Nanyang Technological Univ., Singapore, and KanGAL IIT Kanpur, Kanpur, India, Tech. Rep. 2005005. [Online]. Available: http://www3.ntu.edu.sg/home/EPNSugan, May 2005, pp. 1–49.
[30] H. Wang, S. Rahnamayan, H. Sun, and M. Omran, "Gaussian bare-bones differential evolution," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 634–647, Apr. 2013.
[31] M. Hollander and D. Wolfe, *Nonparametric Statistical Methods*, 2nd ed. New York, NY, USA: Wiley, 1999.
[32] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learning Res.*, vol. 7, pp. 1–30, Jan. 2006.
[33] J. Derrac, S. Garcia, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.

Mauro Campos received the B.Sc. and M.Sc. degrees in physics from the Federal University of Espírito Santo (UFES), Vitória, Brazil. He is currently pursuing the Ph.D. degree at the Graduate Program in Computer Science, UFES.

He is an Adjunct Professor with the Department of Statistics, UFES. His current research interests include natural computing and machine learning, with an emphasis on swarm and evolutionary computation, neural computation, and kernel methods for regression.

Renato A. Krohling received the B.Sc. (Eng.) and M.Sc. degrees in electrical engineering from the Federal University of Espírito Santo (UFES), Vitória, Brazil, in 1990 and 1994, respectively, and the Ph.D. (Dr.-Ing.) degree in electrical engineering from the University of Saarland, Saarbrücken, Germany, in 1999.

From 2000 to 2001, he was a Post-Doctoral Research Associate with the Laboratory for Intelligent Automation, Electrical Engineering Department, UFES. In 2002, he was a Research Associate with the Bio-Inspired Computation Laboratory, Department of Electronics, University of York, York, U.K. From 2003 to 2006, he was a Research Associate with the Chair for Control Systems Engineering, University Dortmund, Dortmund, Germany. Since 2007, he has been with the Graduate Program in Computer Science, and since 2012, he has also been with the Department of Production Engineering, UFES. He is the co-author with M. Jamshidi, L. dos S. Coelho, and P. Fleming of the book entitled *Robust Control Systems With Genetic Algorithms* (CRC Press, Boca Raton, FL, USA, 2002). His current research interests include bioinspired computation, especially swarm intelligence, evolutionary algorithms, neural networks, optimization, intuitionistic fuzzy systems, decision making, and time series prediction.

Dr. Krohling serves as a reviewer for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. Since 2002, he has been a referee for several international conferences, including the IEEE Congress on Evolutionary Computation.

Ivan Enriquez received the B.Sc. degree in statistics from the National University of Engineering, Lima, Peru, the M.Sc. degree in statistics from the University of São Paulo (USP), São Paulo, Brazil, and the Ph.D. degree in statistics from USP.

He is currently with the Department of Statistics, Federal University of Espírito Santo, Vitória, Brazil. His current research interests include computational statistics, stochastics volatility, and time series analysis.