2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Bare Bones Particle Swarm With Scale Mixtures Of Gaussians For Dynamic Constrained Optimization

Mauro Campos
Department of Statistics
Graduate Program in Computer Science
Federal University of Espírito Santo, UFES
Avenida Fernando Ferrari 514, 29075-910
Vitória ES, Brazil
Email: maurocm.campos@gmail.com

Renato A. Krohling
Department of Production Engineering
Graduate Program in Computer Science
Federal University of Espírito Santo, UFES
Avenida Fernando Ferrari 514, 29075-910
Vitória ES, Brazil
Email: krohling.renato@gmail.com

***Abstract*—Bare bones particle swarm optimization (BBPSO) is a well-known swarm algorithm which has shown potential for solving single-objective constrained optimization problems in static environments. In this paper, a generalized BBPSO for dynamic single-objective constrained optimization problems is proposed. An empirical study was carried out to evaluate the performance of the proposed approach. Experimental results show the suitability of the proposed algorithm in terms of effectiveness to find good solutions for all benchmark problems investigated. For comparison purposes, experimental results found by other algorithms are also presented.**

I. Introduction

Dynamic single-objective constrained optimization problems (DCOPs) form a class of problems where the objective function or the constraints can change over time. In static optimization problems, finding a global optimum is considered as the main goal. In dynamic environments, the goal is not only find an optimal solution but also track its trajectory as closely as possible over time. Thus, these problems are to be solved online [1].

Nguyen et al. [2] reported an in-depth survey of the state-of-the-art of academic research on evolutionary computation to deal with dynamic optimization problems (DOPs). Li and Yang [3] presented a review of different approaches based on particle swarm optimization (PSO) [4]–[7] for DOPs. In the literature on DOPs, many algorithms have been designed and tested mainly on dynamic single-objective unconstrained optimization problems (DUOPs) [8]–[19]. Most of these approaches have been tested on the moving peaks benchmark problem proposed by Branke [9]. However, many real-world DOPs have been identified as DCOPs. As a consequence, new algorithms specifically developed for DCOPs have been reported [20]–[24].

Bare bones PSO (BBPSO) [25]–[27] is a well known variant of swarm algorithm originally introduced by Kennedy [25]. In this paper, a generalized BBPSO for DCOPs is proposed. The algorithm is endowed with four mechanisms: 1) a generalized rule based on scale mixture of normal (SMN) distributions [28], [29] which governs the dynamics of the particles in the population; 2) a mechanism to control the population diversity based on concepts from the general framework of information theory; 3) a method to handle constraints based on sum of ranks; and 4) a mechanism to detect changes in the environment based on a fixed set of detectors.

To deal with DOPs, optimization algorithms must be able to detect changes in the environment and efficiently respond to the changed environment. Therefore, algorithms must be capable of maintaining a good balance between exploration (diversification) and exploitation (intensification). Too much stress on exploration would result in pure random search. On the other hand, too much stress on exploitation would result in pure local search. Thus, our approach consists of monitoring the population diversity throughout the search process by employing the concept of entropy as a measure of diversity and use that information to switch (whenever necessary) the search strategy between two regimes: Gaussian local search (exploitation) and global search (exploration) by employing a heavy-tailed distribution in its SMN representation. Experimental results show the suitability of the proposed approach in terms of effectiveness to find good solutions for all benchmark problems investigated.

The remainder of this paper is organized as follows. Section II presents the background information required for the subsequent development of BBPSO with SMN distributions for DCOPs. Section III describes all the ingredients of the proposed algorithm. Experimental results are presented in Section IV. Finally, conclusions are given in Section V.

II. Background

A. Dynamic Constrained Optimization

A dynamic single-objective constrained optimization problem (DCOP) with inequality, equality, upper bound, and lower bound constraints can be stated as follows [1], [2], [20]–[24]:

$$\begin{array}{lll} \text{minimize} & f(\boldsymbol{x},t) & \\ \text{subject to} & h_j(\boldsymbol{x},t)=0 & j\in J \\ & g_i(\boldsymbol{x},t)\leq 0 & i\in I \\ & l_d\leq x_d\leq u_d & d\in\{1,\ldots,D\} \end{array} \tag{1}$$

where $\boldsymbol{x}=(x_1,\ldots,x_D)'$ is an D dimensional vector of decision variables, $t\in T\subseteq\mathbb{N}:=\{0,1,2,\ldots\}$ represents the time, $f(\boldsymbol{x},t)$ is an objective function to be minimized,

$h_j(\boldsymbol{x}, t) = 0$ are $|J|$ equality constraints, and $g_i(\boldsymbol{x}, t) \leq 0$ are $|I|$ inequality constraints. The functions f, h_j, and g_i are linear or non-linear real-valued functions. The values u_d and l_d are the upper bound and the lower bound of x_d, respectively. The upper and lower bounds define the search space $\mathbb{S}$. The inequality and equality constraints define the feasible space $\mathbb{F}(t)$, such that $\mathbb{F}(t) \subseteq \mathbb{S}$ for all t. For each value of t, solutions in $\mathbb{F}(t)$ are called feasible solutions. The goal is to find an optimal feasible solution for all t.

B. Dynamic Benchmark Problems

One useful way to create dynamic benchmark problems is to combine existing static benchmark problems with dynamic rules found in dynamic constrained applications. This can be done by applying the dynamic rules to the parameters of the static problems. Formally, this idea can be described as follows. Consider a static function $f_P(\boldsymbol{x})$ with a set of parameters $P = \{p_1, p_2, \ldots\}$. It is possible generalize $f_P(\boldsymbol{x})$ to its dynamic version $f_{P_t}(\boldsymbol{x}, t)$ by replacing each static parameter in P by a time-dependent expression $p_i(t)$. The dynamic of the time-dependent problem then depends on how $p_i(t)$ varies over time [21]–[23]. Using this idea, Nguyen and Yao [21] introduced a set of six benchmark problems named by G24. This benchmark set was formulated based on a static problem (g24) proposed in [30]. The general form for each problem in the G24 set is as follows:

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}, t) \\ \text{subject to} & g_i(\boldsymbol{x}, t) \leq 0 \quad i \in I \end{array} \tag{2}$$

where $\boldsymbol{x} = (x_1, x_2) \in \mathbb{F}(t) \subseteq \mathbb{S} = [0, 3] \times [0, 4]$ and $t \in T \subset \mathbb{N}$. The objective function takes the following functional form:

- $f(\boldsymbol{x}, t) = -(F_{1,t} + F_{2,t})$

where $F_{j,t} = F_{j,t}(x_j, t) = p_j(t)(x_j + q_j(t))$ with $p_j(t)$ and $q_j(t)$, $j = 1, 2$, as the dynamic parameters which determine how the objective function of each benchmark problem changes over time. The constraint functions may take the following functional forms:

- $g_1(\boldsymbol{x}, t) = -2G_{1,t}^4 + 8G_{1,t}^3 - 8G_{1,t}^2 + G_{2,t} - 2$
- $g_2(\boldsymbol{x}, t) = -4G_{1,t}^4 + 32G_{1,t}^3 - 88G_{1,t}^2 + 96G_{1,t} + G_{2,t} - 36$

where $G_{j,t} = G_{j,t}(x_j, t) = r_j(t)(x_j + s_j(t))$ with $r_j(t)$ and $s_j(t)$, $j = 1, 2$, as the dynamic parameters which determine how the constraint functions of each benchmark problem changes over time. Each benchmark problem in the G24 set has a different mathematical expression for $p_j(t), q_j(t), r_j(t)$ and $s_j(t)$:

1) G24-0 (or G24-u in [23]). In this problem $I = \emptyset$, $p_1(t) = \sin(\kappa\pi t + \pi/2)$, $p_2(t) = 1$, and $q_{1,2}(t) = 0$ where $\kappa \in \{1.00, 0.50, 0.25\}$ determines the severity of objective function changes.
2) G24-1. $I = \{1, 2\}$, $p_1(t) = \sin(\kappa\pi t + \pi/2)$, $p_2(t) = 1$, $q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, and $s_{1,2}(t) = 0$.
3) G24-2. $I = \{1, 2\}$, $p_1(t) = \sin(\kappa\pi t/2 + \pi/2)$,

$$p_2(t) = \begin{cases} 0 & \text{if } t = 0 \\ p_2(t-1) & \text{if } t \bmod 2 = 0 \\ \sin(\kappa\pi(t-1)/2 + \pi/2) & \text{if } t \bmod 2 \neq 0 \end{cases}$$

$q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, and $s_{1,2}(t) = 0$.
4) G24-3. $I = \{1, 2\}$, $p_{1,2}(t) = 1$, $q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, $s_1(t) = 0$, and $s_2(t) = 2 + (\delta/S)t$ where $\delta = x_{2,\max} - x_{2,\min}$ and $S \in \{10, 20, 50\}$ determines the severity of constraint changes.
5) G24-4. $I = \{1, 2\}$, $p_1(t) = \sin(\kappa\pi t + \pi/2)$, $p_2(t) = 1$, $q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, $s_1(t) = 0$, and $s_2(t) = (\delta/S)t$.
6) G24-5. $I = \{1, 2\}$, $p_1(t) = \sin(\kappa\pi t/2 + \pi/2)$,

$$p_2(t) = \begin{cases} 0 & \text{if } t = 0 \\ p_2(t-1) & \text{if } t \bmod 2 = 0 \\ \sin(\kappa\pi(t-1)/2 + \pi/2) & \text{if } t \bmod 2 \neq 0 \end{cases}$$

$q_{1,2}(t) = 0$, $r_{1,2}(t) = 1$, $s_1(t) = 0$, and $s_2(t) = (\delta/S)t$.

Subsequently, Nguyen and Yao expanded the G24 set to a set of eighteen benchmark problems [23].

C. Short Review of BBPSO

In order to keep the description of the algorithm as simple as possible, assume that the optimization process occurs in a static environment and that the search space is also the feasible space, that is, there are no additional constraints posed on the candidate solutions. BBPSO utilizes a population of individuals during the search process. The population is referred to as swarm and its individuals are referred to as particles. The position of a particle represents a potential solution to the problem. Each particle moves in the search space, obeying dynamic rules to update its position. The particles are capable of interacting with the environment and with other particles, namely those particles in its neighborhood. Each particle search by a solution to a given problem, learning from its own past experience and from the experiences of its 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 by refining its search until a good enough solution is found.

Let $\mathbb{S} \subset \mathbb{R}^D$ be the search space of an objective function f and consider a swarm $\mathcal{S}$ with K particles. The position of a particle is denoted by an D-dimensional vector $\boldsymbol{x}_k^\tau = (x_{k1}^\tau, \ldots, x_{kD}^\tau)'$ in $\mathbb{S}$. The index k ($k = 1, \ldots, K$) labels the k-th particle and the index τ ($\tau = 1, 2, \ldots$) represents the iteration counter. Each particle has a neighborhood consisting of a set of particles which it can communicate. The neighborhood of a particle is denoted by $\mathcal{N}_k$ ($\mathcal{N}_k \subseteq \mathcal{S}$) and the neighborhood system $\mathcal{N} = \{\mathcal{N}_k | k = 1, \ldots, K\}$ represents a communication structure often thought of as a social network. There is a number of different schemes to connect the particles. Most implementations use one of two simple sociometric principles. The first, called global topology (or gbest model), 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 ($L < K$). Each particle keeps the memory of the best solution found by itself during the search process (the personal best position found up to the current iteration). The particles use the neighborhood system to exchange information between them. As a result, each particle

Require: $K, \mathcal{N}, f$
1: Initialize the swarm with random positions in $\mathbb{S}$
2: **for** each particle **do**
3: Define $\boldsymbol{p}_k$ and $\boldsymbol{n}_k$
4: **end for**
5: **repeat**
6: **for** each particle **do**
7: **for** $d \in \{1, \ldots, D\}$ **do**
8: Change the position according to Eq. (4)
9: **end for**
10: Update $\boldsymbol{p}_k$ and $\boldsymbol{n}_k$
11: **end for**
12: **until** some termination condition is met
13: $\boldsymbol{x}_* = \text{BEST}(\boldsymbol{n}_k | k = 1, \ldots, K)$; $f_* = f(\boldsymbol{x}_*)$
14: **return** $\boldsymbol{x}_*$ and f_*

Fig. 1. The canonical BBPSO($K, \mathcal{N}, f$).

also keeps the memory of the best solution found by any particle in its neighborhood during the search process (the neighborhood best position found up to the current iteration). The personal and neighborhood best positions are respectively denoted by $\boldsymbol{p}_k^\tau = (p_{k1}^\tau, \ldots, p_{kD}^\tau)'$ and $\boldsymbol{n}_k^\tau = (n_{k1}^\tau, \ldots, n_{kD}^\tau)'$. The rules that control the dynamics of the particles can be presented as follows. The swarm is initialized with random positions. On the initialization, $\boldsymbol{p}_k^1 = \boldsymbol{x}_k^1$ for all k and $\boldsymbol{n}_k^1$ is given by:

$$\boldsymbol{n}_k^1 = \text{BEST}(\boldsymbol{p}_l^1 | l \in \mathcal{N}_k) = \arg\min\{f(\boldsymbol{p}_l^1) | l \in \mathcal{N}_k\} \quad (3)$$

for all k. The position of a particle is updated as follows:

$$x_{kd}^{\tau+1} = \mu_{kd}^\tau + \sigma_{kd}^\tau \cdot z \quad (4)$$

where $\mu_{kd}^\tau = 0.5 \cdot (p_{kd}^\tau + n_{kd}^\tau)$, $\sigma_{kd}^\tau = (|p_{kd}^\tau - n_{kd}^\tau|)$, and $z \sim N(0, 1)$ (z has a Gaussian or normal distribution with mean 0 and variance 1). Note that $x_{kd}^{\tau+1} \sim N(\mu_{kd}^\tau, (\sigma_{kd}^\tau)^2)$ for all k, d and $\tau \geq 1$. The swarm explores the search space of the 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. After updating the position, the personal best position is updated as follows:

$$\boldsymbol{p}_k^{\tau+1} = \text{BEST}(\boldsymbol{x}_k^{\tau+1}, \boldsymbol{p}_k^\tau). \quad (5)$$

Finally, the neighborhood best position is given by:

$$\boldsymbol{n}_k^{\tau+1} = \text{BEST}(\boldsymbol{p}_l^{\tau+1} | l \in \mathcal{N}_k). \quad (6)$$

The process is repeated until some stopping criterion is met. BBPSO returns the global best position $\boldsymbol{x}_*$ and the value of the objective function in $\boldsymbol{x}_*$. The essential steps of BBPSO can be summarized as the pseudo code shown in Fig. 1.

D. Scale Mixtures of Normal Distributions

Let x be a real-valued random variable with a continuous, symmetric, and unimodal distribution having a probability density function (pdf) p, location $\mu \in \mathbb{R}$, and scale $\sigma > 0$. The distribution of x is referred to as a scale mixture of normal (SMN) distributions (or scale mixture of Gaussians) [28], [29] if the pdf p can be expressed as

$$p(x|\boldsymbol{\theta}, \mu, \sigma^2) = \int_0^\infty N(x|\mu, \phi(\lambda)\sigma^2) h(\lambda|\boldsymbol{\theta}) d\lambda \quad (7)$$

where $N(x|\cdot, \cdot)$ is the normal density function, $\phi(\lambda)$ is a positive function of λ, and λ is a random variable with a pdf $h(\cdot|\boldsymbol{\theta})$ defined on $[0, \infty)$ and indexed by a parameter vector $\boldsymbol{\theta}$. The pdf h is referred to as the mixing density of this SMN representation. Note that the distribution of x can be expressed hierarchically as

$$x|\lambda \sim N(x|\mu, \phi(\lambda)\sigma^2) \quad (8)$$

$$\lambda \sim h(\lambda|\boldsymbol{\theta}). \quad (9)$$

From a suitable choice of the mixing density, a rich class of continuous, symmetric, and unimodal distribution can be described by the pdf given in Eq. (7), that can readily accommodate a thicker-than-normal process. The normal distribution can be retrieved when $\lambda = 1$ almost surely. Apart from the normal model, three different types of heavy-tailed densities are explored in this work. These are as follows:

1) THE PEARSON TYPE VII DISTRIBUTION. This model is a member of the elliptical family of distributions and its density is given by

$$P(x|\nu, \delta, \mu, \sigma^2) = \frac{B^{-1}(\frac{\nu}{2}, \frac{1}{2})}{\sqrt{\delta\sigma^2}} \left[1 + \frac{(x-\mu)^2}{\delta\sigma^2}\right]^{-\frac{\nu+1}{2}} \quad (10)$$

where $\mu \in \mathbb{R}$ and $\sigma > 0$ are the location and scale parameters, $\nu > 0$ and $\delta > 0$ are the shape parameters, and $B(a, b)$ is the beta function defined by

$$B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \quad (11)$$

with $a, b > 0$. It can be shown that

$$P(x|\nu, \delta, \mu, \sigma^2) = \int_0^\infty N(x|\mu, \frac{\sigma^2}{\lambda}) Ga(\lambda|\frac{\nu}{2}, \frac{\delta}{2}) d\lambda \quad (12)$$

where $Ga(a, b)$ is the gamma distribution with density

$$Ga(\lambda|a, b) = \frac{b^a}{\Gamma(a)} \lambda^{a-1} e^{-b\lambda} \quad (13)$$

for $a, b, \lambda > 0$. Therefore, the Pearson type VII distribution has a SMN representation. As a result the distribution of x can be expressed hierarchically as

$$x|\lambda \sim N(x|\mu, \lambda^{-1}\sigma^2) \quad \lambda \sim Ga(\lambda|\nu/2, \delta/2). \quad (14)$$

From this SMN representation, some particular models can be obtained. The t-distribution with location μ, scale σ, and ν degrees of freedom can be retrieved when $\nu = \delta$. The Cauchy distribution can be retrieved when $\nu = \delta = 1$.

2) THE VARIANCE GAMMA (VG) DISTRIBUTION. This model was first proposed by Madan and Seneta [31]

to model share market returns and its density is given by

$$VG(x|\nu,\mu,\sigma^2) \propto \left[\frac{|x-\mu|}{\sqrt{\nu\sigma^2}}\right]^{\frac{\nu-1}{2}} K_{\frac{\nu-1}{2}}\left(\frac{\nu|x-\mu|}{\sqrt{\nu\sigma^2}}\right) \quad (15)$$

where $\mu \in \mathbb{R}$ and $\sigma > 0$ are the location and scale parameters, $\nu > 0$ is the shape parameter, and $K_\eta(a)$ is the modified Bessel function of the third kind with index η defined by

$$K_\eta(a) = \frac{1}{2}\int_0^\infty z^{\eta-1} e^{-\frac{a}{2}(z+z^{-1})} dz \quad (16)$$

with $a > 0$ and $\eta \in \mathbb{R}$. It can be shown that

$$VG(x|\nu,\mu,\sigma^2) = \int_0^\infty N(x|\mu, \frac{\sigma^2}{\lambda}) IGa(\lambda|\frac{\nu}{2}, \frac{\nu}{2}) d\lambda \quad (17)$$

where $IGa(a,b)$ is the inverse gamma distribution with density

$$IGa(\lambda|a,b) = \frac{b^a}{\Gamma(a)}\lambda^{-(a+1)} e^{-b/\lambda} \quad (18)$$

for $a, b, \lambda > 0$. Therefore, the VG distribution has a SMN representation. As a result the distribution of x can be expressed hierarchically as

$$x|\lambda \sim N(x|\mu, \lambda^{-1}\sigma^2) \qquad \lambda \sim IGa(\lambda|\nu/2, \nu/2). \quad (19)$$

From this SMN representation, the Laplace distribution (also known as the double-sided exponential distribution) can be retrieved when $\nu = 2$.

3) THE SLASH DISTRIBUTION. The pdf of this distribution is symmetric, unimodal, and has heavier tails than those of the normal density. The canonical slash distribution (with location 0 and scale 1) has the same tail heaviness as the Cauchy. However, it is less peaked in the center and thus more realistic in representing data. It is also useful in simulation studies where it can introduce distributional challenges in order to evaluate a statistical procedure. It can be shown that the slash density can be expressed as

$$S(x|\nu,\mu,\sigma^2) = \int_0^1 N(x|\mu, \frac{\sigma^2}{\lambda}) Be(\lambda|\nu, 1) d\lambda \quad (20)$$

where $Be(a,b)$ is the beta distribution with density

$$Be(\lambda|a,b) = B^{-1}(a,b)\lambda^{a-1}(1-\lambda)^{b-1} \quad (21)$$

for $a, b > 0$ and $\lambda \in [0,1]$. Therefore, the slash distribution has a SMN representation. As a result the distribution of x can be expressed hierarchically as

$$x|\lambda \sim N(x|\mu, \lambda^{-1}\sigma^2) \qquad \lambda \sim Be(\lambda|\nu, 1). \quad (22)$$

The slash distribution includes the normal case when $\nu \to \infty$.

In Section III, SMN representations are applied in the dynamic rule to update the position of a particle of the swarm.

E. Entropy and Kullback-Leibler Distance

Let x be a discrete random variable with alphabet χ and probability mass function $p(x)$, $x \in \chi$. The entropy of x is a measure of its uncertainty and it is defined by

$$H_b(x) = -\sum_{x\in\chi} p(x)\log_b p(x) = -E_x(\log_b p(x)). \quad (23)$$

The convention that $0\log_b 0 = 0$ is used and it is easily justified by continuity since $x\log_b x \to 0$ as $x \to 0$. Thus adding terms of zero probability does not change the entropy. If $b = 2$, then the entropy is measured in bits. If $b = e$, then the entropy is measured in nats. The entropy is a functional of the distribution of x. It does not depend on the actual values taken by x, but only on the probabilities. It follows immediately from the definition (23) that $H_b(x) \geq 0$.

The Kullback-Leibler distance (or relative entropy) between two probability mass functions $p(x)$ and $q(x)$ is defined as

$$KL(p|q) = \sum_{x\in\chi} p(x)\log_b \frac{p(x)}{q(x)} = E_x\left(\log_b \frac{p(x)}{q(x)}\right). \quad (24)$$

It is possible to show that $KL(p|q) \geq 0$ with equality if and only if $p(x) = q(x)$ for all $x \in \chi$. One important consequence of this result is that the discrete distribution with the maximum entropy is the uniform distribution. More precisely, $H_b(x) \leq \log_b|\chi|$, where $|\chi|$ denotes the number of elements (or states) in the range of x, with equality if and only if x has a uniform distribution over χ. This result can be obtained from the following equation:

$$0 \leq KL(p|u) = \log_b|\chi| - H_b(x) \quad (25)$$

where $u(x) = |\chi|^{-1}$, for all $x \in \chi$, represents the uniform distribution over χ. In Section III, the Kullback-Leibler distance is used as a measure to control the population diversity of the swarm as a whole.

III. BBPSO WITH SMN DISTRIBUTIONS FOR DCOPS

In this section, BBPSO with scale mixture of normal distributions (SMN distributions) is introduced. The proposed algorithm is explained in the following subsections.

A. The Swarm

The structure of the swarm in the proposed algorithm is equivalent to the structure of the swarm in BBPSO, as discussed in Section II-C. The swarm has K particles and a neighborhood system $\mathcal{N}$. Each particle in $\mathcal{S}$ is characterized by a vector $(\boldsymbol{x}_k, \boldsymbol{p}_k, \boldsymbol{n}_k)'$. The global best model is obtained when $\mathcal{N}_k = \mathcal{S}$ for all k and the local best model is obtained when $\mathcal{N}_k \subset \mathcal{S}$ for all k.

B. The Dynamic Rule

The position of a particle is updated as follows:

$$\boldsymbol{x}_k|\boldsymbol{\lambda} = \boldsymbol{\mu}_k + \boldsymbol{\lambda}^{-1/2} \odot \boldsymbol{\sigma}_k \odot \boldsymbol{z} \quad (26)$$

where

- $\boldsymbol{\mu}_k = (\mu_{k1}, \ldots, \mu_{kD})' = \frac{1}{2}(p_{k1} + n_{k1}, \ldots, p_{kD} + n_{kD})'$
- $\boldsymbol{\sigma}_k = (\sigma_{k1}, \ldots, \sigma_{kD})' = (|p_{k1} - n_{k1}|, \ldots, |p_{kD} - n_{kD}|)'$

- $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_D)' \sim (h(\lambda_1|\boldsymbol{\theta}), \overset{idd}{\ldots}, h(\lambda_D|\boldsymbol{\theta}))'$
- $\boldsymbol{z} = (z_1, \ldots, z_D)' \sim (N_1(0,1), \overset{idd}{\ldots}, N_D(0,1))'$

and $\boldsymbol{\lambda}^{-1/2} = (\lambda_1^{-1/2}, \ldots, \lambda_D^{-1/2})'$. Each random variable λ_d has a pdf $h(\lambda_d|\boldsymbol{\theta})$ defined on $[0, \infty)$ and independent of z_d, where $\boldsymbol{\theta}$ is a parameter vector indexing the distribution of λ_d. The distribution of x_{kd} can be expressed hierarchically as

$$x_{kd}|\lambda_d \sim N(x_{kd}|\mu_{kd}, \lambda_d^{-1}\sigma_{kd}^2) \quad (27)$$

$$\lambda_d \sim h(\lambda_d|\boldsymbol{\theta}) \quad (28)$$

and its pdf is given by

$$p(x_{kd}) = \int_0^\infty N(x_{kd}|\mu_{kd}, \lambda_d^{-1}\sigma_{kd}^2) h(\lambda_d|\boldsymbol{\theta}) d\lambda_d. \quad (29)$$

Thus, the distribution of x_{kd} has a SMN representation. As discussed in Section II, the Eq. (26) ensures that:

1) Model I. When $\lambda_d = 1$ (with probability 1), then $x_{kd} \sim N(\mu_{kd}, \sigma_{kd}^2)$ for all d and k.
2) Model II. When $\lambda_d \sim Ga(\nu/2, \delta/2)$, then $x_{kd} \sim P(\nu, \delta, \mu_{kd}, \sigma_{kd}^2)$ for all d and k.
   - If $\nu = \delta$, then $\lambda_d \sim Ga(\nu/2, \nu/2)$ and $x_{kd} \sim t(\nu, \mu_{kd}, \sigma_{kd}^2)$ (t-distribution).
   - If $\nu = \delta = 1$, then $\lambda_d \sim Ga(1/2, 1/2)$ and $x_{kd} \sim C(\mu_{kd}, \sigma_{kd}^2)$ (Cauchy distribution).
3) Model III. When $\lambda_d \sim IGa(\nu/2, \nu/2)$, then $x_{kd} \sim VG(\nu, \mu_{kd}, \sigma_{kd}^2)$ for all d and k.
   - If $\nu = 1$, then $\lambda_d \sim IGa(1, 1)$ and $x_{kd} \sim La(\mu_{kd}, \sqrt{2}\sigma_{kd}/2)$ (Laplace distribution).
4) Model IV. If $\lambda_d \sim Be(\nu, 1)$, then $x_{kd} \sim S(\nu, \mu_{kd}, \sigma_{kd}^2)$ for all d and k.

The Model I is equivalent to the canonical BBPSO of Kennedy [25]. The Model II generalizes the dynamic rule proposed by Krohling and Mendel [32], and the Model III generalizes the dynamic rule proposed by Krohling and Coelho [33].

C. The Diversity Control

The concept of entropy has been used as a measure of diversity in swarm and evolutionary computation [34], [35]. Petalas et al. [34] introduced a memetic PSO that combines PSO with a local search method for computing periodic orbits of nonlinear mappings. Liu et al. [35] introduced an entropy-driven evolutionary approach for solving single-objective unconstrained optimization problems in static environments.

For a population P divided in M classes, the entropy of P is defined as

$$H(P) = -\sum_m P_m \log P_m \quad (30)$$

where P_m is the proportion of P that occupies the class m at a given time t. In our context, $P = \{f(\boldsymbol{p}_k)|k = 1, \ldots, K\}$ where $\boldsymbol{p}_k$ is the personal best position of the particle k. Large values of entropy are associated with small values of P_m for nearly every class, that is, the elements of P are evenly distributed over practically all classes. On the other hand, small values entropy are associated with large values of P_m for a few classes, that is, the elements of P are concentrated in few classes. Therefore, high entropy indicates high diversity in the population and low Kullback-Leibler distance (see Eqs. (24) and (25)) between P_m, $m = 1, \ldots, M$, and the uniform distribution over all classes.

An adaptive control of parameter is proposed in our algorithm by using the Kullback-Leibler distance (see Eqs. (24) and (25)). Our approach employs the following rule: when the Kullback-Leibler distance is above of an user-defined threshold, a strategy of global search is used to increase diversity and encourage exploration. Conversely, when the Kullback-Leibler distance is below of a threshold, a strategy of Gaussian local

Require: $D, K, \mathcal{N}, \boldsymbol{\theta}, d_c, f(\cdot|P_t), g_i(\cdot|P_t)(i \in I), \Delta, \tau_{\max}$
1: $\tau \leftarrow 1$; $t \leftarrow 0$
2: Establish the set of detectors in $\mathbb{S}$
3: **for** each particle **do**
4: $\quad \boldsymbol{x}_k \leftarrow \boldsymbol{x}_{\min} + (\boldsymbol{x}_{\max} - \boldsymbol{x}_{\min}) \odot \boldsymbol{U}(0,1)$
5: $\quad \boldsymbol{p}_k \leftarrow \boldsymbol{x}_k$
6: **end for**
7: **for** each particle **do**
8: $\quad$ {See SumRank() procedure in Subsection III-D}
9: $\quad \boldsymbol{n}_k \leftarrow \text{SumRank}(\boldsymbol{p}_l | l \in \mathcal{N}_k)$
10: **end for**
11: BEST $\leftarrow$ SumRank$(\boldsymbol{n}_k|k = 1, \ldots, K)$
12: $\boldsymbol{x}_* \leftarrow \emptyset$; $\boldsymbol{x}_* \leftarrow \boldsymbol{x}_* \cup$ BEST
13: $\boldsymbol{f}_* \leftarrow \emptyset$; $\boldsymbol{f}_* \leftarrow \boldsymbol{f}_* \cup f(\text{BEST}, t|P_t)$
14: **repeat**
15: $\quad$ **if** τ mod $\Delta = 0$ **then**
16: $\quad\quad t \leftarrow t + 1$
17: $\quad\quad$ Update $P_t = (p(t), q(t), r(t), s(t))'$
18: $\quad$ **end if**
19: $\quad$ DetectChange() acts on the set of detectors
20: $\quad$ **if** DetectChange() = TRUE **then**
21: $\quad\quad$ **for** each particle **do**
22: $\quad\quad\quad$ Call SumRank() to update $\boldsymbol{p}_k$ and $\boldsymbol{n}_k$
23: $\quad\quad$ **end for**
24: $\quad$ **end if**
25: $\quad$ Calculate the KL distance for the current swarm.
26: $\quad$ **if** $KL(p|u) > d_c$ **then**
27: $\quad\quad \boldsymbol{\lambda} \sim (h(\lambda_1|\boldsymbol{\theta}), \overset{idd}{\ldots}, h(\lambda_D|\boldsymbol{\theta}))'$ (Exploration)
28: $\quad$ **else**
29: $\quad\quad \boldsymbol{\lambda} = \mathbf{1} = (1, \ldots, 1)$ (Exploitation)
30: $\quad$ **end if**
31: $\quad$ **for** each particle **do**
32: $\quad\quad$ Change the position according to Eq. (26)
33: $\quad\quad$ Call SumRank() to update $\boldsymbol{p}_k$ and $\boldsymbol{n}_k$
34: $\quad$ **end for**
35: $\quad$ BEST $\leftarrow$ SumRank$(\boldsymbol{n}_k|k = 1, \ldots, K)$
36: $\quad \boldsymbol{x}_* \leftarrow \boldsymbol{x}_* \cup$ BEST
37: $\quad \boldsymbol{f}_* \leftarrow \boldsymbol{f}_* \cup f(\text{BEST}, t|P_t)$
38: $\quad \tau \leftarrow \tau + 1$
39: **until** some termination condition is met
40: **return** Convergence plot: $\{1, \ldots, \tau_{\max}\} \times f_*$

Fig. 2. BBPSO with SMN distributions for DCOPs.

Require: $\mathcal{P} \subseteq \mathcal{S}$
1: **for** each particle in $\mathcal{P}$ **do**
2: Calculate the properties $f(\boldsymbol{x}, t), s(\boldsymbol{x}, t)$, and $v(\boldsymbol{x}, t)$
3: **end for**
4: Rank the particles with respect to f, s, and v independently to get r_f, r_s, and r_v
5: **if** feasible particles exist in $\mathcal{P}$ **then**
6: **for** each particle **do**
7: $\phi(\boldsymbol{x}, t) \leftarrow r_f + r_s + r_v$
8: **end for**
9: **else**
10: **for** each particle **do**
11: $\phi(\boldsymbol{x}, t) \leftarrow r_s + r_v$
12: **end for**
13: **end if**
14: Sort the particles according to ϕ
15: BestParticle $\leftarrow \arg\min\{\phi(\boldsymbol{x}_l, t) : l \in \mathcal{P}\}$
16: **return** BestParticle

Fig. 3. SumRank($\mathcal{P}$) procedure.

search is used to decrease diversity and encourage exploitation. The strategy of global search is realized with a heavy-tailed distribution in its SMN representation.

D. The Method to Handle Constraints

Consider the following numerical properties associated with the position of a particle:

1) $f(\boldsymbol{x}, t)$ represents the value of objective function in the position $\boldsymbol{x}$ at time t.
2) $s(\boldsymbol{x}, t) = \sum_{i=1}^{n} \max\{0, g_i(\boldsymbol{x}, t)\}^2$ represents the sum of squares of constraint violation at time t.
3) $v(\boldsymbol{x}, t)$ represents the number of constraints violated at time t.

The first problem to deal with is the proper scaling of these three properties. To solve this problem, a simple ranking method proposed in [36] is used. Consider a population $\mathcal{P} \subseteq \mathcal{S}$ of particles. The particles in $\mathcal{P}$ are ranked with respect to each property (f, s, v) independently. This produces three new terms, denoted by r_f, r_s, and r_v, respectively. Clearly, r_f, r_s, and r_v are all of the same order of magnitude. As a result, these three terms can be easily manipulated without bias. Next, the following aggregation strategy is defined:

$$\phi(\boldsymbol{x}, t) = \begin{cases} r_s + r_v \text{ if all particles in } \mathcal{P} \text{ are infeasible} \\ r_f + r_s + r_v \text{ otherwise.} \end{cases} \tag{31}$$

Here, $\phi(\boldsymbol{x}, t)$ is the new objective function to be minimized. When all the particles in the population are infeasible, our aim is to seek the first feasible solution from the search space. The information from f then becomes unimportant and hence only r_s and r_v are used. When feasible individuals exist in the population, the algorithm should explore the search space in order to find the optimum solution. In the second case, r_s and r_v serve as a penalty function to penalize the infeasible solutions. On the other hand, r_f enables us to relate the infeasible individuals to the feasible individuals based on the f value alone. The consideration of this term allows us retain for the next generation (iteration) only those infeasible solutions with a small degree of constraint violation and a small objective function value. This strategy is necessary to maintain the diversity of the population, by exploring into the infeasible regions of the search space.

Clearly, Eq. (31) attempt to integrate the information from the objective function and constraint violation for the infeasible solutions. However, it is important to note that the Eq. (31) does not require a penalty coefficient and this is the most important feature of this strategy. By transforming the relevant numerical properties into ranking terms of the same order of magnitude, different terms can be summed directly without invoking a penalty coefficient. The essential steps of BBPSO with SMN distributions for DCOPs can be summarized as the pseudo code shown in Fig. 2. The method for handling constraints is shown in Fig. 3.

E. Detection of Changes in Environment

DCOPs differ from their static counterparts by the fact that either the objective function or the constraint functions can change over time. So, the detection of change in the environment is very important and must be implemented by a specific mechanism. There are two types of changes that can affect the search process. One, is a change in the objective function, and the other is a change in the constraint functions. Both types of changes can be efficiently detected by establishing a fixed set of detectors (or sentinels), and then evaluating their objective values (or fitness values) and constraint violations after each iteration. If the present and past objective values and constraint violations are different, then, indeed, the environment has changed. It is important to emphasize that our approach uses an entirely isolated set of detectors, differing fully from particles that constitute the swarm.

IV. Results and Discussions

A set of experiments were conducted to compare empirically the performance of BBPSO with SMN distributions for DCOPs. The G24 set, presented in Subsection II-B, was used to investigate the performance of the proposed algorithm. The benchmark problems chosen contain characteristics that are representative of real-world DCOPs. As discussed in Section III, the proposed algorithm is endowed with four mechanisms: 1) a generalized rule based on SMN distributions which governs the dynamics of the particles in the population; 2) a mechanism to control the population diversity based on concepts from the general framework of information theory; 3) a mechanism to handle constraints based on sum of ranks; and, finally, 4) a mechanism to detect changes in the environment based on a fixed set of detectors. In all experiments, the number of particles (the population size K) was set to 25 and the local best ring model was used as the neighborhood system (the social network).

TABLE I
OFFLINE ERRORS FOR DIFFERENT ALGORITHMS IN THE MEDIUM SETTINGS[†].

-	Problems →	G24-0(dF,noC)		G24-1(dF,fC)		G24-2(dF,fC)	
Algorithms ↓	References ↓	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
GA-noElit	[22, p. 181]	0.361	0.064	0.803	0.041	0.429	0.061
RIGA-noElit	[22, p. 181]	0.234	0.019	0.633	0.046	0.351	0.060
HyperM-noElit	[22, p. 181]	0.249	0.034	0.450	0.094	0.304	0.025
GA-elit	[22, p. 181]	0.214	0.037	0.587	0.085	0.329	0.074
RIGA-elit	[22, p. 181]	0.131	0.034	0.401	0.046	0.283	0.021
HyperM-elit	[22, p. 181]	0.173	0.042	0.475	0.060	0.376	0.055
GA+Repair	[22, p. 181], [24]	0.468	0.059	0.226	0.024	0.281	0.036
GSA+Repair	[24]	0.049	0.004	0.132	0.015	0.182	0.019
dRepairGA	[22, p. 181]	0.362	0.063	0.101	0.022	0.198	0.030
dRepairRIGA	[22, p. 181]	0.254	0.048	0.082	0.015	0.162	0.021
dRepairHyperM	[22, p. 181]	0.319	0.034	0.093	0.023	0.171	0.026
Genocop	[22, p. 181]	0.120	0.028	0.099	0.034	0.177	0.031
dGenocop	[22, p. 181]	0.091	0.035	0.085	0.024	0.099	0.028
BBPSO+SMNd+SumRanks[a]	-	0.028	0.017	0.122	0.166	**0.071**	0.031
BBPSO+SMNd+SumRanks[b]	-	0.062	0.029	0.132	0.148	0.091	0.056
BBPSO+SMNd+SumRanks[c]	-	**0.017**	0.013	**0.078**	0.112	0.072	0.027
-	Problems →	G24-3(fF,dC)		G24-4(dF,dC)		G24-5(dF,dC)	
Algorithms ↓	References ↓	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
GA-noElit	[22, p. 181]	0.884	0.093	0.718	0.095	0.465	0.061
RIGA-noElit	[22, p. 181]	0.742	0.100	0.612	0.052	0.401	0.048
HyperM-noElit	[22, p. 181]	0.692	0.094	0.562	0.062	0.347	0.068
GA-elit	[22, p. 181]	0.384	0.092	0.627	0.045	0.373	0.031
RIGA-elit	[22, p. 181]	0.340	0.045	0.492	0.071	0.259	0.031
HyperM-elit	[22, p. 181]	0.561	0.104	0.494	0.039	0.297	0.047
GA+Repair	[22, p. 181], [24]	0.156	0.008	0.211	0.015	0.236	0.024
GSA+Repair	[24]	0.028	0.004	0.073	0.012	0.153	0.013
dRepairGA	[22, p. 181]	0.034	0.005	0.170	0.026	0.181	0.032
dRepairRIGA	[22, p. 181]	0.029	0.004	0.140	0.028	0.152	0.017
dRepairHyperM	[22, p. 181]	0.027	0.005	0.059	0.010	0.131	0.019
Genocop	[22, p. 181]	0.099	0.034	0.177	0.031	**0.059**	0.039
dGenocop	[22, p. 181]	0.028	0.007	0.140	0.043	0.114	0.025
BBPSO+SMNd+SumRanks[a]	-	**0.013**	0.006	0.062	0.035	0.074	0.051
BBPSO+SMNd+SumRanks[b]	-	0.039	0.021	0.083	0.052	0.148	0.114
BBPSO+SMNd+SumRanks[c]	-	**0.013**	0.008	**0.051**	0.026	0.061	0.019

[†]Experimental setup: $K = 25$, change frequency $= 1000$ evaluations ($\Delta = 40$), $\kappa = 1/2$, and $S = 20$.

Here dF=dynamic function, noC=no constraint, fC=fixed constraint, fF=fixed function, and dC=dynamic constraint.

[a]For this algorithm $\lambda_d \sim h(\lambda_d|\boldsymbol{\theta}) = Ga(\lambda_d|1/2, 1/2)$ for all d.

[b]For this algorithm $\lambda_d \sim h(\lambda_d|\boldsymbol{\theta}) = IGa(\lambda_d|1, 1)$ for all d.

[c]For this algorithm $\lambda_d \sim h(\lambda_d|\boldsymbol{\theta}) = Be(\lambda_d|1, 1)$ for all d.

For each experiment, summary statistics were calculated based on 50 runs, each one with the following experimental setup: change frequency $= 1000$ evaluations ($\Delta = 40$), $\kappa = 1/2$ (the severity of objective function changes), and $S = 20$ (the severity of constraint changes). The experimental results are shown in Table I. For comparison purposes, results found by other algorithms established in the literature are also presented in Table I. The results show that the proposed algorithm is comparable in performance with other algorithms. In particular, the proposed approach which uses a dynamic rule based on the slash distribution (with $\lambda_d \sim Be(\lambda_d|\nu, 1)$ for all d, see Section III) outperforms other algorithms (GA+Repair, GSA+Repair, dRepairGA, dRepairRIGA, dRepairHyperM, Genocop, and dGenocop) for most of the problems investigated.

V. CONCLUSION

In this article, a generalized BBPSO was proposed for DCOPs, which are optimization problems to be solved online over time. The proposed algorithm is endowed with four mechanisms for: 1) monitoring the population diversity throughout the search process by employing the concept of

entropy as a measure of diversity; 2) increasing diversity when necessary by employing a heavy-tailed distribution in its SMN representation; 3) handling constraints; and 4) detecting changes in the environment. The combined effect of these mechanisms endows the proposed algorithm with the ability to maintain a good balance between exploration (diversification) and exploitation (intensification) 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. The proposed algorithm is simple and easy to implement like other swarm algorithms. In future works, new methods to handle constraints and detect changes in environments will be investigated.

REFERENCES

[1] R. Morrinson, *Designing Evolutionary Algorithms for Dynamic Environments*. Heidelberg: Springer-Verlag, 2004.

[2] T. Nguyen, S. Yang, and J. Branke, *Evolutionary dynamic optimization: a survey of the state of the art*. Swarm and Evolutionary Computation, vol. 6, pp. 1-24, 2012.

[3] C. Li and S. Yang, *A comparative study on particle swarm optimization in dynamic environments*, in S. Yang and X. Yao (Eds.), Evolutionary Computation for DOPs, Studies in Computational Intelligence 490, pp. 109-136. Heidelberg: Springer-Verlag, 2013.

[4] J. Kennedy and R. Eberhart, *Particle swarm optimization*, in Proc. IEEE International Conference on Neural Networks, 1995, pp. 1941-1948.

[5] Y. Shi and R. Eberhart, *A modified particle swarm optimizer*, in Proc. IEEE Congress on Evolutionary Computation, 1998, pp. 69-73.

[6] M. Clerc and J. Kennedy, *The particle swarm - explosion, stability, and convergence in a multidimensional complex space*. IEEE Transactions on Evolutionary Computation, vol. 6, no. 1, pp. 58-73, February 2002.

[7] D. Bratton and J. Kennedy, *Defining a standard for particle swarm optimization*, in Proc. IEEE Swarm Intelligence Symposium, 2007, pp. 120-127.

[8] T. Back, *On the behavior of evolutionary algorithms in dynamic environments*, in Proc. IEEE Congress on Evolutionary Computation, 1998, pp. 446-451.

[9] J. Branke, *Memory enhanced evolutionary algorithms for changing optimization problems*, in Proc. IEEE Congress on Evolutionary Computation, 1999, pp. 1875-1882.

[10] R. Eberhart and Y. Shi, *Tracking and optimizing dynamic systems with particle swarms*, in Proc. IEEE Congress on Evolutionary Computation, 2001, pp. 94-97.

[11] Y. Jin and J. Branke, *Evolutionary optimization in uncertain environments a survey*. IEEE Transactions on Evolutionary Computation, vol. 9, no. 3, pp. 303-317, June 2005.

[12] T. Blackwell and J. Branke, *Multiswarms, exclusion, and anti convergence in dynamic environments*. IEEE Transactions on Evolutionary Computation, vol. 10, no. 4, pp. 459-472, August 2006.

[13] R. Lung and D. Dumitrescu, *A collaborative model for tracking optima in dynamic environments*, in Proc. IEEE Congress on Evolutionary Computation, 2007, pp. 564-567.

[14] W. Du and B. Lin, *Multi-strategy ensemble particle swarm optimization for dynamic optimization*. Information Sciences, vol. 178, pp. 3096-3109, 2008.

[15] H. Richter, *Detecting change in dynamic fitness landscapes*, in Proc. IEEE Congress on Evolutionary Computation, 2009, pp. 1613-1620.

[16] R. Lung and D. Dumitrescu, *Evolutionary swarm cooperative optimization in dynamic environments*. Natural Computing, vol. 9, pp. 83-94, 2010.

[17] L. Liu, S. Yang, and D. Wang, *Particle swarm optimization with composite particles in dynamic environments*. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 40, no. 6, pp. 1634-1648, December 2010.

[18] C. Li and S. Yang, *A general framework of multipopulation methods with clustering in undetectable dynamics environments*. IEEE Transactions on Evolutionary Computation, vol. 16, no. 4, pp. 556-577, August 2012.

[19] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, and M. Meybodi, *A novel multi-swarm algorithm for optimization in dynamic optimization based on particle swarm optimization*. Applied Soft Computing, vol. 13, pp. 2144-2158, 2013.

[20] C.-A. Liu, *New dynamic constrained optimization PSO algorithm*, in Proc. Fourth International Conference on Natural Computation, 2008, pp. 650-653.

[21] T. Nguyen and X. Yao, *Benchmarking and solving dynamic constrained optimization*, in Proc. IEEE Congress on Evolutionary Computation, 2009, pp. 690-697.

[22] T. Nguyen, *Continuous dynamic optimisation using evolutionary algorithms*. Ph.D. thesis, The University of Birmingham, Birmingham, UK. [Online]. Available: http://etheses.bham.ac.uk/1296, October 2010.

[23] T. Nguyen and X. Yao, *Continuous dynamic constrained optimization the challenges*. IEEE Transactions on Evolutionary Computation, vol. 16, no. 6, pp. 769-786, December 2012.

[24] K. Pal, C. Saha, S. Das, and C. Coello, *Dynamic constrained optimization with offspring repair based gravitational search algorithm*, in Proc. IEEE Congress on Evolutionary Computation, 2013, pp. 2414-2421.

[25] J. Kennedy, *Bare bones particle swarms*, in Proc. IEEE Swarm Intelligence Symposium, 2003, pp. 80-87.

[26] T. Blackwell, *A study of collapse in bare bones particle swarm optimization*. IEEE Transactions on Evolutionary Computation, vol. 16, no. 3, pp. 354-372, June 2012.

[27] M. Campos, R. Krohling, and I. Enriquez, *Bare bones particle swarm optimization with scale matrix adaptation*. IEEE Transactions on Cybernetics. [Online]. Available: http://ieeexplore.ieee.org, November 2013, DOI:10.1109/TCYB.2013.2290223.

[28] D. Andrews and C. Mallows, *Scale mixtures of normal distributions*. Journal of the Royal Statistical Society. Series B (Methodological), vol. 36, no. 1, pp. 99-102, 1974.

[29] S. Choy and J. Chan, *Scale mixtures of distributions in statistical modelling*. Australian and New Zealand Journal of Statistics, vol. 50(2), pp. 135-146, 2008.

[30] J. Liang, T. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C. Coello, and K. Deb, *Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization*, Nanyang Technological University, Singapure, Science Institute, University of Iceland, Iceland, Evolutionary Computation Group, CINVESTAV-IPN, Mexico, France Télécon, France, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology, Kanpur, India, Technical Report. [Online]. Available: http://www3.ntu.edu.sg/home/EPNSugan, September 2006, pp. 1-24.

[31] D. Madan and E. Seneta, *The variance gamma (V.G.) model for share market returns*. The Journal of Business, vol. 63, no. 4, pp. 511-524, October 1990.

[32] R. Krohling and E. Mendel, *Bare bones particle swarm optimization with Gaussian or Cauchy jumps*, in Proc. IEEE Congress on Evolutionary Computation, 2009, pp. 3285-3291.

[33] R. Krohling and L. Coelho, *PSO-E: particle swarm with exponential distribution*, in Proc. IEEE Congress on Evolutionary Computation, 2006, pp. 5577-5582.

[34] Y. Petalas, K. Parsopoulos, and M. Vrahatis, *Entropy-based memetic particle swarm optimization for computing periodic orbits of nonlinear mappings*, in Proc. IEEE Congress on Evolutionary Computation, 2007, pp. 2040-2047.

[35] S.-H. Liu, M. Mernik, and B. Bryant, *To explore or to exploit: an entropy-driven approach for evolutionary algorithms*. International Journal of Knowledge-based and Intelligent Engineering Systems, vol. 13, pp. 185-206, 2009.

[36] P. Ho and K. Shimizu, *Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme*, Information Sciences, vol. 177, pp. 2985-3004, 2007.