Correspondence

Coevolutionary Particle Swarm Optimization Using Gaussian Distribution for Solving Constrained Optimization Problems

Renato A. Krohling and Leandro dos Santos Coelho

***Abstract*—In this correspondence, an approach based on coevolutionary particle swarm optimization to solve constrained optimization problems formulated as min–max problems is presented. In standard or canonical particle swarm optimization (PSO), a uniform probability distribution is used to generate random numbers for the accelerating coefficients of the local and global terms. We propose a Gaussian probability distribution to generate the accelerating coefficients of PSO. Two populations of PSO using Gaussian distribution are used on the optimization algorithm that is tested on a suite of well-known benchmark constrained optimization problems. Results have been compared with the canonical PSO (constriction factor) and with a coevolutionary genetic algorithm. Simulation results show the suitability of the proposed algorithm in terms of effectiveness and robustness.**

***Index Terms*—Constrained optimization, Gaussian distribution, min–max problem, particle swarm optimization (PSO).**

I. Introduction

Evolutionary algorithms (EAs) have shown to be a promising approach to solve complex constrained optimization problems [1]. A very important factor in constrained optimization is how to handle constraints. Some algorithms to handle constraints in EAs have been proposed in the last decade and they can be grouped as [1]: 1) preservation of the feasible individuals; 2) repair of infeasible solutions; 3) use of decoders; 4) penalty functions; and 5) hybrid algorithms. The performance of these methods depends on the problem at hand. The main question is to evaluate the fitness of infeasible individuals. In most of the cases, the approaches to handle constraints developed for EAs have been adapted to other soft computing algorithms, although it may not be always possible to. An alternative algorithm to EAs, which has demonstrated to be a promising approach for solving constrained optimization problems, is particle swarm optimization (PSO) [4]–[9].

In the last few years, several heuristics have been developed to improve the performance and set up suitable parameters for the PSO algorithm [10]–[12]. Some theoretical work to analyze the trajectory of particles has been carried out. Van den Bergh [13] studied the trajectory of particles under different inertia weights and acceleration coefficients. A constriction factor has been proposed by Clerc and Kennedy [14] to ensure convergence. Trelea [15] analyzed the trajectory of a deterministic particle in a one-dimensional search space using dynamic systems theory. However, the deterministic approach proposed in [15] may not be straightforward generalized for the analysis of PSO with random coefficients in a multidimensional search space.

In PSO, a uniform probability distribution to generate random numbers to updating the velocity is used. However, the use of other probability distributions may improve the ability to fine-tune or even to escape from local optima. In the meantime, the use of Gaussian and Cauchy probability distributions has been proposed to generate random numbers to update the velocity equation [16]–[24] inspired by studies of mutation operators in fast evolutionary programming [29], [30]. Coelho and Krohling [16], [17] proposed the use of a truncated Gaussian and Cauchy probability distribution to generate random numbers for the velocity updating equation. First, random numbers are generated using the Gaussian or the Cauchy probability distribution in the interval $[-1; 1]$, and then mapped to the interval $[0; 1]$. Secrest and Lamont [18] proposed also a rule for the Gaussian motion of the particles of the swarm. Kennedy [19] also used a Gaussian distribution in his bare bones particle. Higashi and Iba [20] and Stacey *et al.* [21] use an additional term to the velocity updating equation, which consists of a perturbation operator implemented as a mutation for generating random numbers according to the Gaussian distribution. Miranda and Fonseca [22] and Wei *et al.* [23] combine evolutionary programming with PSO. Esquivel and Coello [24] have presented an approach using Cauchy mutation to updating the velocity equation. All these approaches attempted to improve the performance of the standard PSO, but the amount of parameters of the algorithm to tune remained the same.

A different approach has been proposed by Clerc [26], [42] termed Tribes, which is a parameter-free PSO. In Tribes, the number of particles is automatically found out during the search. This approach is attractive from the user's perspective and has been applied to solve the flow shop scheduling problem [27]. An approach using a diversity of the population borrowed from EA has been proposed by Ursem [28]. Krohling [25] proposed the updating of the velocity equation based on the Gaussian distribution; the accelerating constants c_1 and c_2 are not more specified by the user, but instead of that they are generated using the absolute value of the Gaussian distribution with zero mean and unit standard deviation. PSO using Gaussian distribution was first tested on unconstrained optimization problems [25]. In this correspondence, we apply the algorithm to handle more challenging problems, i.e., min–max problems. This approach with Gaussian motion of the particle is quite different from those proposed in previous works [16]–[24]. PSO using Gaussian distribution [25] is based on the expected value, which should be generated from a probability distribution (the Gaussian one) in order to generate suitable stochastic coefficients to the velocity updating equation.

Previous studies on coevolutionary algorithms (CEAs) have demonstrated the suitability of the approach to solve constrained optimization problems [36]–[40]. A constrained optimization problem is transformed into an unconstrained optimization problem by introducing Lagrange multipliers. The optimization problem is then formulated as a min–max problem [31], a representation that arises in many areas of science and engineering, especially in game theory and robust optimal control. Min–max problems are considered difficult to solve. Hillis [32], in his pioneering work, proposed a method inspired by the coevolution of populations. Two independent genetic algorithms (GAs) were used, whereas one for sorting networks (host) and the

Manuscript received April 29, 2005; revised September 19, 2005, November 15, 2005, and January 31, 2006. This paper was recommended by Associate Editor H. Ishibuchi.

R. A. Krohling was with the Chair for Control Systems Engineering, Faculty of Electrical Engineering, University of Dortmund, 44221 Dortmund, Germany. He is now an independent consultant in engineering, CEP 29.100-021, Espírito Santo, Brazil (e-mail: krohling.renato@gmail.com).

L. dos Santos Coelho is with the Automation and Systems Laboratory, Pontifical Catholic University of Parana, Centro de Ciências Exatas e de Tecnologia (CCET) /Programa de Pós Graduação em Engenharia de Produção e Sistemas (PPGEPS), Rua Imaculada Conceição, 1 80215-901, Curitiba-PR, Brazil (e-mail: leandro.coelho@pucpr.br).

Digital Object Identifier 10.1109/TSMCB.2006.873185

other for test cases (parasites). Both GAs evolve simultaneously and are coupled through the fitness function.

In CEAs, the fitness of an individual depends not only on the individual itself but also the individuals of other EA. CEAs have shown useful results for solving complex problems. In this case, the fitness of an individual is evaluated by means of a competition with the members of the other population [32]–[35]. Inspired by the work of Hillis [32], the coevolutionary approach has been extended to solve constrained optimization problems [36]–[40]. Barbosa [36], [37] presented a method to solve min–max problems by using two independent populations of GA coupled by a common fitness function. Tahk and Sun [38] also used a coevolutionary augmented Lagrangian method to solve min–max problems by means of two populations of evolution strategies with an annealing scheme. The first population is made up of the variables vector, and the second one is made up of the Lagrange multiplier vector. Laskari *et al.* [41] have also presented a method using PSO for solving min–max problems, but not using a coevolutionary approach.

In this correspondence, based on our previous work on coevolutionary particle swarm optimization (CPSO) [39], we observed that the standard PSO presents deficiencies to find a fine-tuning of the solution. Further, in order to improve the performance of the CPSO, we have also proposed a truncated Gaussian distribution to generate the accelerating coefficients of PSO [40], and some improvement of performance has been obtained. In this correspondence, we extend our previous work on PSO using Gaussian distribution [25] and apply it for solving challenging constrained optimization problems. Two populations of independent PSO are evolved: one for the variable vector and the other for the Lagrange multiplier vector. At the end of the optimization the first PSO provides the variable vector, and the second PSO provides the Lagrange multiplier vector.

The rest of the correspondence is organized as follows. In Section II, the formulation of the min–max problem is described. The standard PSO is explained in Section III. In Section IV, the PSO using Gaussian distribution is developed. In Section V, the coevolutionary particle swarm algorithm is presented to solve min–max problems. Section VI provides simulation results and comparisons for some benchmark constrained optimization problems, followed by conclusions in Section VII.

II. Problem Formulation

Many problems in various scientific areas and real-world applications can be formulated as constrained optimization problems. Generally, a constrained optimization problem is given by

$$\min_{\boldsymbol{x} \in \Re^n} f(\boldsymbol{x})$$

subject to

$$\begin{aligned} g_i(\boldsymbol{x}) &\leq 0, \qquad i = 1, \ldots, m \\ h_i(\boldsymbol{x}) &= 0, \qquad i = 1, \ldots, l \end{aligned} \tag{1}$$

where $f(\boldsymbol{x})$ is the objective function, $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}} \in \Re^n$ is the vector of variables, $g_i(\boldsymbol{x})$ is the vector of inequality constraints, and $h_i(\boldsymbol{x})$ is the vector of equality constraints.

The set $S \subseteq \Re^n$ designates the search space, which is defined by the lower and upper bounds of the variables: $\underline{x}_j \leq x_j \leq \overline{x}_j$ with $j = 1, \ldots, n$. Points in the search space, which satisfy the equality and inequality constraints, are feasible candidate solutions.

The Lagrange-based method [31] is a classical approach to formulate constrained optimization problems. By introducing the Lagrangian formulation, the dual problem associated with the primal problem (1) can be written as

$$\max_{\boldsymbol{\mu}, \boldsymbol{\lambda}} L(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\lambda})$$

subject to

$$\begin{aligned} \mu_i &\geq 0, \qquad i = 1, \ldots, m \\ \lambda_i &\geq 0, \qquad i = 1, \ldots, l \end{aligned} \tag{2}$$

where

$$L(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\mu}^{\mathrm{T}} g(\boldsymbol{x}) + \boldsymbol{\lambda}^{\mathrm{T}} h(\boldsymbol{x}) \tag{3}$$

$\boldsymbol{\mu}$ is a $m \times 1$ multiplier vector for the inequality constraints;
$\boldsymbol{\lambda}$ is a $l \times 1$ multiplier vector for the equality constraints.

If the problem (1) satisfies the convexity conditions over S, then the solution of the primal problem (1) is the vector $\boldsymbol{x}^*$ of the saddle point $\{\boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*\}$ of $L(\boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*)$ so that

$$L(\boldsymbol{x}^*, \boldsymbol{\mu}, \boldsymbol{\lambda}) \leq L(\boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*) \leq L(\boldsymbol{x}, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*).$$

The saddle point can be obtained by minimizing $L(\boldsymbol{x}^*, \boldsymbol{\mu}, \boldsymbol{\lambda})$ with the optimal Lagrange multipliers $(\boldsymbol{\mu}^*, \boldsymbol{\lambda}^*)$ as a fixed vector of parameter. In general, the optimal values of the Lagrange multipliers are unknown *a priori.* The duality theorem [31] can be used to overcome this difficulty. According to the duality theorem, the primal problem (1) subject to the inequality and equality constraints can be transformed into a dual or min–max problem.

Solving the min–max problem

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{\mu}, \boldsymbol{\lambda}} L(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) \tag{4}$$

provides the minimizer $\boldsymbol{x}^*$ as well as the Lagrange multiplier $\boldsymbol{\mu}^*, \boldsymbol{\lambda}^*$. However, for nonconvex problems, the solution of the dual problem does not coincide with that of the primal problem. In that case, a penalty term associated with equality and inequality constraints is added to the Lagrangian function. The augmented Lagrangian is given by

$$L_{\mathrm{a}}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}, r) = f(\boldsymbol{x}) + \sum_{i=1}^{m} p_i(\boldsymbol{x}, \boldsymbol{\mu}, r) + \boldsymbol{\lambda}^{\mathrm{T}} h(\boldsymbol{x}) + r \sum_{i=1}^{l} h_i^2(\boldsymbol{x}) \tag{5}$$

where the term p_i for the ith inequality constraint is given by

$$p_i(\boldsymbol{x}, \mu_i, r) = \begin{cases} \mu_i g_i(\boldsymbol{x}) + r g_i^2(\boldsymbol{x}), & \text{if } g_i(\boldsymbol{x}) - \frac{\mu_i}{2r} \\ -\frac{\mu_i^2}{4r}, & \text{if } g_i(\boldsymbol{x}) < -\frac{\mu_i}{2r} \end{cases} \tag{6}$$

and r is a penalty constant. It can be shown that the solutions of the primal problem and the augmented Lagrangian are identical. The goal is to find the saddle point $(\boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*)$. In Section V, the coevolutionary particle swarm using Gaussian distribution (CPSO-GD) will be presented to solve the min–max problem. Next, we provide some background on PSO.

III. PSO

PSO is an effective optimization method that belongs to the category of swarm intelligence methods, originally developed by Kennedy and Eberhart [2], [3]. PSO is initialized with a population of candidate solutions called particles, which have associated randomized velocities. Since particles move through the search space, each particle keeps track of its coordinates in the search space, which are associated with the best solution (fitness) it has achieved so far, $\boldsymbol{p}_{\text{best}}$. Another "best" value tracked by the global version of the particle swarm optimizer is the overall best value, gbest, and its location, obtained so far by any particle in the population. First, the velocities and then the positions of the particle are updated as follows:

$$\boldsymbol{v}_i(t+1) = w\boldsymbol{v}_i(t) + c_1 \text{rand}\,(\boldsymbol{p}_{\text{i}} - \boldsymbol{x}_{\text{i}}(\text{t})) + \text{c}_2 \text{Rand}\,\left(\boldsymbol{p}_{\text{g}} - \boldsymbol{x}_{\text{i}}(\text{t})\right) \tag{7}$$

$$\boldsymbol{x}_i(t) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1). \tag{8}$$

The vector $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^{\text{T}}$ stands for the position of the ith particle, $\boldsymbol{v}_i = [v_{i1}, v_{i2}, \ldots, v_{in}]^{\text{T}}$ stands for the velocity of the ith particle, and $\boldsymbol{p}_i = [p_{i1}, p_{i2}, \ldots, p_{in}]^{\text{T}}$ represents the best previous position (the position giving the best fitness value) of the ith particle. The index g represents the index of the best particle among all the particles, i.e., $\boldsymbol{p}_{\text{g}}$ is the global best. The variable w is the inertia weight, c_1 and c_2 are positive constants; rand and Rand are random numbers in the range [0; 1] generated according to a uniform probability distribution. The random numbers are generated anew for each dimension $i = 1, \ldots, n$ of the particle i.

The inertia weight w represents the degree of the momentum of the particles. The second part is the "cognition" part, which represents the independent behavior of the particle itself. The third part is the "social" part, which represents the collaboration among the particles. The constants c_1 and c_2 represent the weighting of the "cognition" and "social" parts that pull each particle toward $\boldsymbol{p}_{\text{best}}$ and $\boldsymbol{p}_{\text{gbest}}$. The PSO algorithm is described in [2] and [3].

Clerc and Kennedy [14] in their study on stability and convergence of PSO have introduced a constriction coefficient k. In that case, the velocity equation is updated according to

$$\boldsymbol{v}_i(t+1) = k\left[\boldsymbol{v}_i(t) + c_1\,\text{rand}\,(\boldsymbol{p}_i - \boldsymbol{x}_i(t)) + c_2\,\text{Rand}\,\left(\boldsymbol{p}_{\text{g}} - \boldsymbol{x}_i(t)\right)\right] \tag{9}$$

where

$$k = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}$$

with $\varphi = c_1 + c_2 > 4$ and k is a function of c_1 and c_2. Usually, φ is set to 4.1 ($c_1 = c_2 = 2.05$), and the constriction coefficient k is 0.729; but there are other possible choices for the constriction coefficients. A detailed theoretical analysis of the derivation of the constriction factor can be found in [14].

For a simplified analysis assuming the independence of the local and global terms, the stochastic coefficient of the terms $(\boldsymbol{p}_i - \boldsymbol{x}_i)$ and $(\boldsymbol{p}_{\text{g}} - \boldsymbol{x}_i)$ is calculated by multiplying 0.729 by 2.05 resulting 1.494, which multiplied by 0.5 (mean value of a uniform probability distribution $U(0,1)$ gives a constriction coefficient 0.729. Trelea [15] also presented some analysis of convergence for a one-dimensional particle. The parameter settings used in his simulations were $w = 0.6$ and $c_1 = c_2 = 1.7$. The mean value of the stochastic coefficients for the terms $(\boldsymbol{p}_i - \boldsymbol{x}_i)$ and $(\boldsymbol{p}_{\text{g}} - \boldsymbol{x}_i)$ is calculated by multiplying $c_1 = c_2 = 1.7$ by 0.5 (mean value of $U(0,1)$, which gives 0.85. As can be seen from (9), the two differences are averaged; each with its own uniformly distributed random numbers. In the Appendix, we rewrite (9) over two time steps, and one concludes that a possible good choice for the stochastic coefficients lies in the interval [0.72; 0.86].

IV. PSO Using Gaussian Distribution

Since the mean value of the two stochastic coefficients for the local term $(\boldsymbol{p}_i - \boldsymbol{x}_i)$ and the global term $(\boldsymbol{p}_{\text{g}} - \boldsymbol{x}_i)$ lies in the interval [0.72; 0.86], then a candidate for the probability distribution that generates random numbers is the absolute value of the Gaussian probability distribution with zero mean and unit variance, i.e., $\text{abs}(N(0,1))$. The probability density function of $\text{abs}(N(0,1))$ is given by

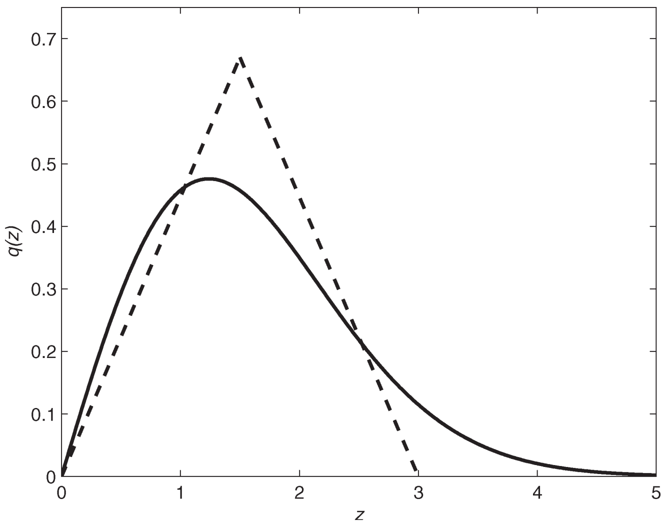

Fig. 1. Sum of two random uniform probability density functions with support set $[0; kc] = [0; 1.49]$ (dashed line) and sum of two absolute Gaussian probability density functions $\text{abs}(N(0,1))$ (solid line).

$$q(x) = \frac{2}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}, \qquad x \geq 0. \tag{10}$$

The mean or expected value $E(x)$ of $\text{abs}(N(0,1))$ is calculated by

$$E(x) = \frac{2}{\sqrt{2\pi}} \int_0^\infty x e^{-\frac{x^2}{2}} dx = 0.798.$$

The variance of $\text{abs}(N(0,1))$ is given by

$$V(x) = \sigma^2 = \frac{2}{\sqrt{2\pi}} \int_0^\infty (x - \mu)^2 e^{-\frac{x^2}{2}} dx$$

$$\sigma^2 = \frac{2}{\sqrt{2\pi}} \int_0^\infty (x - \mu)^2 2e^{-\frac{x^2}{2}} dx = 0.36$$

which results in a standard deviation $\sigma = 0.60$. The objective consists of generating the stochastic coefficients for the terms $(\boldsymbol{p}_i - \boldsymbol{x}_i)$ and $(\boldsymbol{p}_{\text{g}} - \boldsymbol{x}_i)$. The velocity equation now is updated according to

$$\boldsymbol{v}_i(t+1) = |\text{rand}n|\,(\boldsymbol{p}_i - \boldsymbol{x}_i(t)) + |\text{Rand}n|\,\left(\boldsymbol{p}_{\text{g}} - \boldsymbol{x}_i(t)\right) \tag{11}$$

where $|\text{rand}n|$ and $|\text{Rand}n|$ are positive random numbers generated using $\text{abs}(N(0,1))$.

Let us consider z a random variable, which results from the sum of two random variables with probability density function $q(x)$. Then, the probability distribution function of the sum $q(z)$ is given by

$$q(z) = \frac{2}{\sqrt{\pi}} e^{-\frac{z^2}{4}} \cdot \text{erf}(0.5z)$$

where $\text{erf}(z) = (1/\sqrt{\pi}) \int_0^z e^{-x^2} dx$.

Fig. 1 shows the difference between the sum of two uniform random variables $U(0,1)$ multiplied by 1.49 and the sum of two $\text{abs}(N(0,1))$. The use of $\text{abs}(N(0,1))$ for generating the stochastic coefficients of

PSO seems to provide a good compromise between the probability of having a large number of small amplitudes around the current points (fine-tuning) and a small probability of having higher amplitudes, which may allow particles to move away from the current point and escape from local minima. Since the velocity is calculated from an average of two differences between previous best and current position, Kennedy [45] has demonstrated that the probability density is constant on a large area around the center of the box (hyperrectangle).

Preliminary results for unconstrained optimization have shown the suitability of the PSO algorithm using $\mathrm{abs}(N(0,1))$ [25]. In the following, we describe the coevolutionary PSO algorithm to handle constrained optimization problems.

V. CPSO-GD

Two populations of PSOs are involved in the CPSO for solving the min–max problem formulated according to (5). The first PSO focuses on evolving the variable vector $\boldsymbol{x}$ while the vector of Lagrangian multiplier $\boldsymbol{\theta} = [\boldsymbol{\mu}, \boldsymbol{\lambda}]$ is maintained, “frozen.” Only the variable vector $\boldsymbol{x}$ is represented in the population P_1. The second PSO focuses on evolving the Lagrangian multiplier vector $\boldsymbol{\theta}$ while the Population P_1 is maintained “frozen.” Only the multiplier vector $\boldsymbol{\theta}$ is represented in the population P_2. The two PSOs interact with each other through a common fitness evaluation. For the first PSO, the problem is a minimization problem and the fitness value of each particle $\boldsymbol{x}$ is evaluated according to

$$f_1(\boldsymbol{x}) = \max_{\boldsymbol{\theta} \in P_2} L_{\mathrm{a}}(\boldsymbol{x}, \boldsymbol{\theta}). \tag{12}$$

Each particle $\boldsymbol{x}_i$ of P_1 is evaluated against all particle $\boldsymbol{\theta}_j$ of P_2. The fitness for the particle $\boldsymbol{x}_i$ is the highest value of $f_1(\boldsymbol{x}_i)$. This process is repeated for all N particles of P_1.

The second problem consists in a maximization problem and the fitness value of each particle $\boldsymbol{\theta}$ is evaluated according to

$$f_2(\theta) = \min_{\boldsymbol{x} \in P_1} L_{\mathrm{a}}(\boldsymbol{x}, \boldsymbol{\theta}). \tag{13}$$

Each particle $\boldsymbol{\theta}_j$ of P_2 is evaluated against all particle $\boldsymbol{x}_i$ of P_1. The fitness for particle $\boldsymbol{\theta}_j$ is the smallest value of $f_2(\boldsymbol{\theta}_j)$. This process is repeated for all M particles of P_2.

In the PSO algorithm, all particles are transferred into the next generation (no selection mechanism). The cooperation among particles is established through the “history” variables $\boldsymbol{p}_{\mathrm{best}}$ and $\boldsymbol{p}_{\mathrm{gbest}}$, which are updated if better fitness values are obtained. The CPSO-GD algorithm is described in Fig. 2. Within each generation g, the first PSO, named PSO1, is run for C cycles; then the second PSO, named PSO2, is run for C cycles. This process is repeated until the maximum number of generations has been elapsed. The global best in the population P_1, i.e., $\boldsymbol{p}_{\mathrm{gbest1}}$ is the solution for the variable vector $\boldsymbol{x}$, and the global best in the population P_2, i.e., $\boldsymbol{p}_{\mathrm{gbest2}}$ is the solution for the Lagrangian multiplier vector $\boldsymbol{\theta}$. In the CPSO-GD algorithm, when one PSO is running, the other serves as its environment, so each PSO has a changing environment from generation to generation.

VI. Simulation Results

A. Benchmark Problems

Four benchmark problems of constrained optimization [38] have been used to investigate the performance of the proposed algorithm.

The problem G1 consists of minimizing

$$f(\boldsymbol{x}) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

subject to

$$\begin{aligned}
g_1(\boldsymbol{x}) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0\\
g_2(\boldsymbol{x}) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0\\
g_3(\boldsymbol{x}) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0\\
g_4(\boldsymbol{x}) &= -8x_1 + x_{10} \le 0\\
g_5(\boldsymbol{x}) &= -8x_2 + x_{11} \le 0\\
g_6(\boldsymbol{x}) &= -8x_3 + x_{12} \le 0\\
g_7(\boldsymbol{x}) &= -2x_4 - x_5 + x_{10} \le 0\\
g_8(\boldsymbol{x}) &= -2x_6 - x_7 + x_{11} \le 0\\
g_9(\boldsymbol{x}) &= -2x_8 - x_9 + x_{12} \le 0
\end{aligned}$$

with

$$\begin{aligned}
&0 \le x_i \le 1, && i = 1, \ldots, 9\\
&0 \le x_i \le 100, && i = 10, 11, 12\\
&0 \le x_i, \le 1, && i = 13.
\end{aligned}$$

The global minimum is known to be $\boldsymbol{x}^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$ with $f(\boldsymbol{x}^*) = -15$.

The problem G7 consists of minimizing

$$\begin{aligned}
f(\boldsymbol{x}) = {} & x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2\\
&+ 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2\\
&+ 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45
\end{aligned}$$

subject to

$$\begin{aligned}
g_1(\boldsymbol{x}) &= -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \le 0\\
g_2(\boldsymbol{x}) &= 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \le 0\\
g_3(\boldsymbol{x}) &= 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0\\
g_4(\boldsymbol{x}) &= x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \le 0\\
g_5(\boldsymbol{x}) &= -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \le 0\\
g_6(\boldsymbol{x}) &= 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \le 0\\
g_7(\boldsymbol{x}) &= -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \le 0\\
g_8(\boldsymbol{x}) &= 0.5(x_1 - 8)^2 + 2(x_2 - 4) + 3x_5^2 - x_6 - 30 \le 0
\end{aligned}$$

with

$$-10 \le x_i \le 10, \qquad i = 1, \ldots, 10.$$

The global minimum is $\boldsymbol{x}^* = (2.1719, 2.3636, 8.7739, 5.0959, 0.9906, 1.4305, 1.3216 , 9.8287, 8.2800, 8.3759)$ with $f(\boldsymbol{x}^*) = 24.306$.

The problem G9 consists of minimizing

$$\begin{aligned}
f(\boldsymbol{x}) = {} & (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2\\
&+ 10x_5^6 + 7x_6^2 - 4x_6x_7 - 10x_6 - 8x_7
\end{aligned}$$

subject to

$$\begin{aligned}
g_1(\boldsymbol{x}) &= -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \le 0\\
g_2(\boldsymbol{x}) &= -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \le 0\\
g_3(\boldsymbol{x}) &= -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \le 0\\
g_4(\boldsymbol{x}) &= 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0
\end{aligned}$$

```
CPSO - GD Algorithm :
Input parameters : Swarm size, and the penalty parameter.
Initialize the two populations $P_1$ and $P_2$ with size $N$, and $M$, respectively using a uniform probability distribution.
Population $P_1$ consists of the variable vector $\boldsymbol{x}$ and the population $P_2$ consists of the Lagrange multiplier vector
$\boldsymbol{\theta} = [\boldsymbol{\mu}, \boldsymbol{\lambda}]$.
for $g$ =1 to $G_{\max}$    // $G_{\max}$ is the maximum numbers of generations
    for $t$ =1 to $C$    // $C$ is the numbers of cycles running PSO1
        for $i$=1 to $N$    // evolving population $P_1$
            for $j$=1 to $M$    // population $P_2$ remains frozen
                $\boldsymbol{\theta}_j = \boldsymbol{p}_{best2}$
                $f(\boldsymbol{x}_i) = L_a(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$    // fitness evaluation
            end for
            $\boldsymbol{x}_i = \arg\max_{\boldsymbol{\theta}_j \in P_2} L_a(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$
            if $f(\boldsymbol{x}_i) < f(\boldsymbol{p}_{best1})$ then
                $\boldsymbol{p}_{best1} = \boldsymbol{x}_i$    // updating the personal best of $P_1$
            if $f(\boldsymbol{x}_i) < f(\boldsymbol{p}_{gbest1})$ then
                $\boldsymbol{p}_{gbest1} = \arg\min\{f(\boldsymbol{p}_{best1})\}$    //updating the global best of $P_1$
        end for
        $\boldsymbol{v}_i(t+1) = |randn|(\boldsymbol{p}_{best1} - \boldsymbol{x}_i(t)) + |Randn|(\boldsymbol{p}_{gbest1} - \boldsymbol{x}_i(t))$  // velocity updating for $P_1$
        $\boldsymbol{x}_i(t) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1)$    // variable vector updating for $P_1$
    end for //    end of the number of cycles running PSO1
    for $t$ = 1 to $C$    // $C$ is the numbers of cycles running PSO2
        for $j$=1 to $M$    // evolving population $P_2$
            for $i$=1 to $N$    // population $P_1$ remains frozen
                $\boldsymbol{x}_i = \boldsymbol{p}_{best1}$
                $f(\boldsymbol{\theta}_j) = L_a(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$    // fitness evaluation
            end for
            $\boldsymbol{\theta}_j = \arg\min_{\boldsymbol{x}_i \in P_1} L_a(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$
            if $f(\boldsymbol{\theta}_j) > f(\boldsymbol{p}_{best2})$ then
                $\boldsymbol{p}_{best2} = \boldsymbol{\theta}_j$    // updating the personal best of $P_2$
            if $f(\boldsymbol{\theta}_j) > f(\boldsymbol{p}_{gbest2})$ then
                $\boldsymbol{p}_{gbest2} = \arg\max\{f(\boldsymbol{p}_{best2})\}$    // updating the global best of $P_2$
        end for
        $\boldsymbol{v}_j(t+1) = |randn|(\boldsymbol{p}_{best2} - \boldsymbol{\theta}_j(t)) + |Randn|(\boldsymbol{p}_{gbest2} - \boldsymbol{\theta}_j(t))$  // velocity updating for $P_2$
        $\boldsymbol{\theta}_j(t) = \boldsymbol{\theta}_j(t) + \boldsymbol{v}_j(t+1)$    // Lagrange multiplier vector updating for $P_2$
    end for    //end of the number of cycles running PSO2
end for    // end of the generation number
Output: $f(\boldsymbol{x}^*)$, $\boldsymbol{p}_{gbest1} = \boldsymbol{x}^*$, and $\boldsymbol{p}_{gbest2} = \boldsymbol{\theta}^* = [\boldsymbol{\mu}^*, \boldsymbol{\lambda}^*]$.
```

Fig. 2. Pseudocode of the CPSO-GD algorithm.

with

$$-10 \le x_i \le 10, \qquad i = 1, \ldots, 7.$$

The global minimum is $\boldsymbol{x}^* = (2.3304, 1.9513, -0.4775, 4.3657, -0.6244, 1.0381, 1.5942)$ with $f(\boldsymbol{x}^*) = 680.63$.

The problem G10 consists of minimizing

$$f(\boldsymbol{x}) = x_1 + x_2 + x_3$$

subject to

$$g_1(\boldsymbol{x}) = -1 + 0.0025(x_4 + x_6) \le 0$$

$$g_2(\boldsymbol{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \le 0$$

$$g_3(\boldsymbol{x}) = -1 + 0.01(x_8 - x_5) \le 0$$

$$g_4(\boldsymbol{x}) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83\,333.333 \le 0$$

$$g_5(\boldsymbol{x}) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \le 0$$

$$g_6(\boldsymbol{x}) = -x_3x_8 + 1\,250\,000 + x_3x_5 - 2500x_5 \le 0$$

TABLE I
RESULTS USING CPSO-GD AND CPSO (SWARM SIZE $P_1 = P_2 = 30$)

Problem	Method	best	median	mean	std. dev.	worst
G1	CPSO	-10.997	-6.000	-6.266	1.436	-5.0
optimal solution (-15.0)	CPSO-GD	**-15.0**	-14.996	-14.997	0.0017	-14.994
G7	CPSO	24.7881	36.504	72.427	65.889	199.6038
optimal solution (24.306)	CPSO-GD	**24.481**	25.641	25.778	0.7767	27.597
G9	CPSO	680.632	680.644	680.653	0.0196	680.711
optimal solution (680.63)	CPSO-GD	680.739	680.887	680.929	0.1843	681.590
G10	CPSO	7610.0	11100	11347	3484.9	21000
optimal solution (7049.331)	CPSO-GD	**7064.3**	7987.2	8015.3	767.47	10265

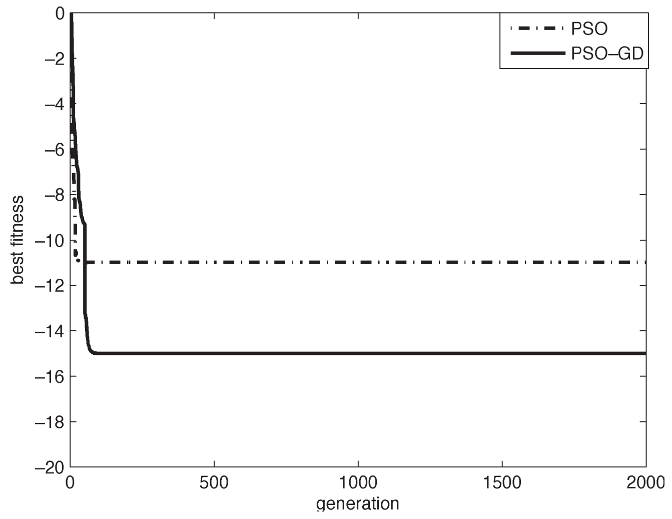

Fig. 3. Fitness of the best particle (best run) for problem G1.

with

$$\begin{aligned} &100 \le x_i \le 10\,000, && i = 1 \\ &1000 \le x_i \le 10\,000, && i = 2, 3 \\ &10 \le x_i \le 1000, && i = 4, \ldots, 8. \end{aligned}$$

The global minimum is $\boldsymbol{x}^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.979, 286.4162, 395.5979)$ with $f(\boldsymbol{x}^*) = 7049.33$.

B. Experimental Settings

The maximum numbers of generations was set to 2000 and the number of cycles to 2. The penalty constant r introduced by the augmented Lagrange formulation was set to 100 as recommended in a previous work [38]. This value will be used in all experiments if not stated otherwise. For the benchmark problems studied, the population size was initially set to $P_1 = P_2 = 30$. Later on, we will also present results for another different swarm size. The particles are randomly initialized within the boundaries for each run according to a uniform probability distribution.

For the standard PSO (with constriction factor) [14] the parameters were set to $c_1 = c_2 = 2.05$, and $k = 0.729$. For the PSO using Gaussian distribution, the only parameter of the algorithm to be specified by the user is the number of particles (population size). Each experiment is run 30 times. Each run is terminated only when the maximum number of generations has been elapsed.

C. Discussions of the Results

The simulation results using the standard CPSO (with constriction factor) and CPSO-GD (with the absolute Gaussian distribution) are shown in Table I.

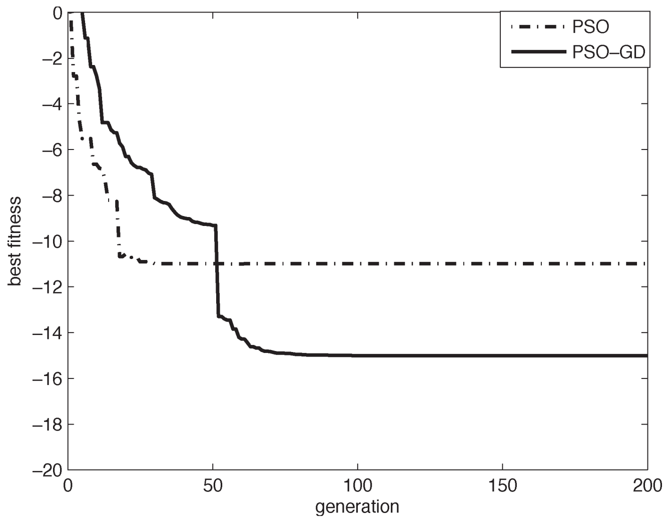

Fig. 4. Fitness of the best particle (best run) for problem G1 showing details of the first 200 generations.

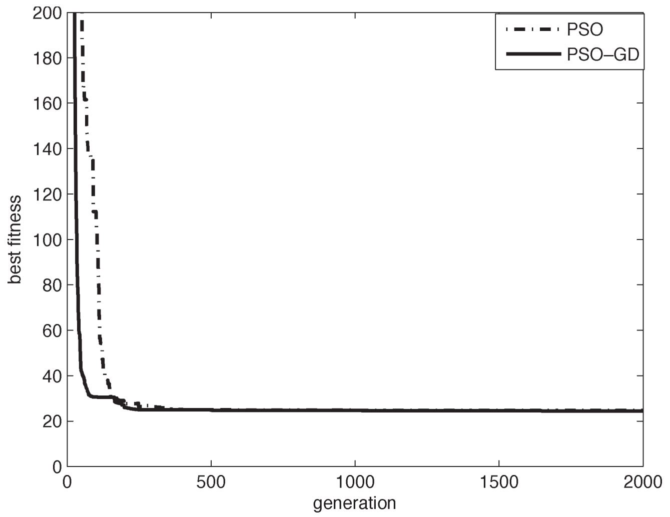

Fig. 5. Fitness of the best particle (best run) for problem G7.

For problem G1, the optimal solution is −15. The fitness value for the best particle is shown in Fig. 3 and in more detail in Fig. 4. From Table I, the median and mean of fitness found by CPSO-GD is much better than the results obtained by CPSO. It can be seen that CPSO-GD finds the optimal solution, while CPSO does not. Using CPSO-GD, the optimal solution is found in approximately 100 generations, while CPSO gets stuck in local minima and are not more able to escape. It can be seen that the solution found presents a very small standard deviation (0.0017), which demonstrates the robustness of the algorithm.

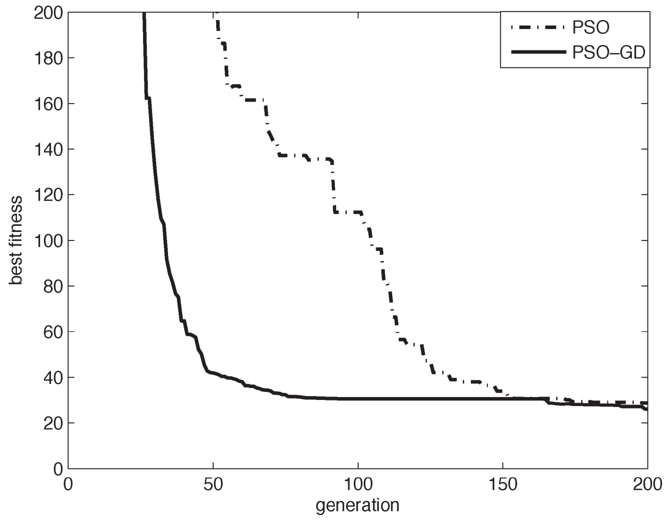

Fig. 6. Fitness of the best particle (best run) for problem G7 showing details of the first 200 generations.

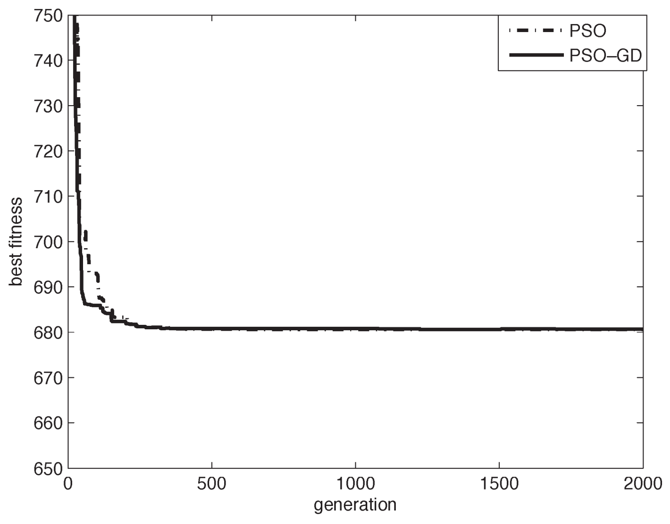

Fig. 7. Fitness of the best particle (best run) for problem G9.

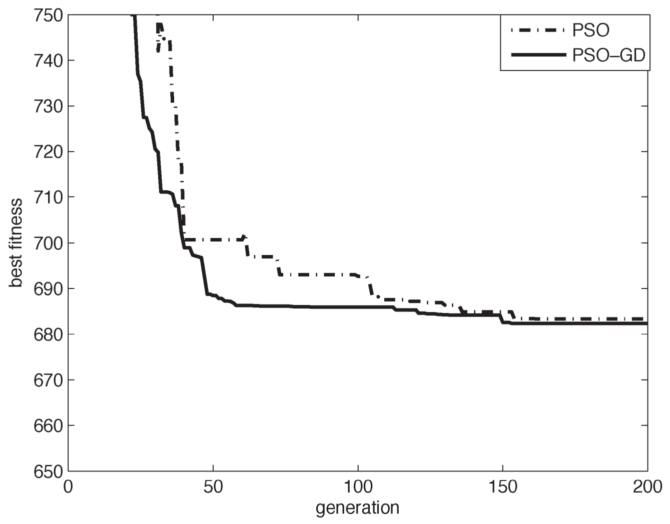

Fig. 8. Fitness of the best particle (best run) for problem G9 showing details of the first 200 generations.

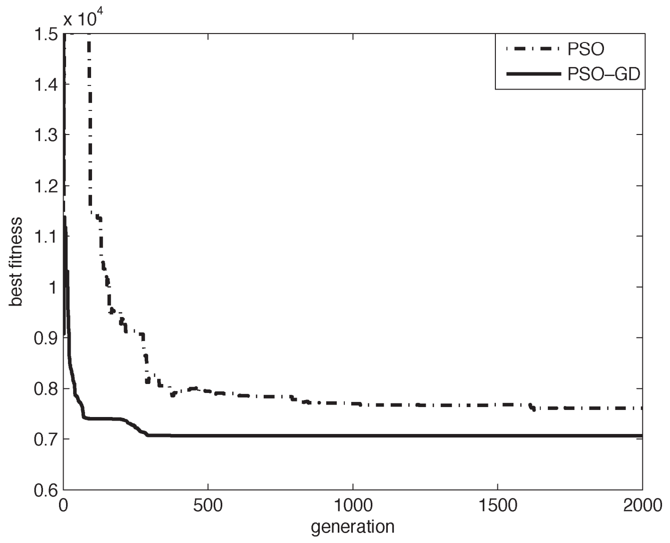

Fig. 9. Fitness of the best particle (best run) for problem G10.

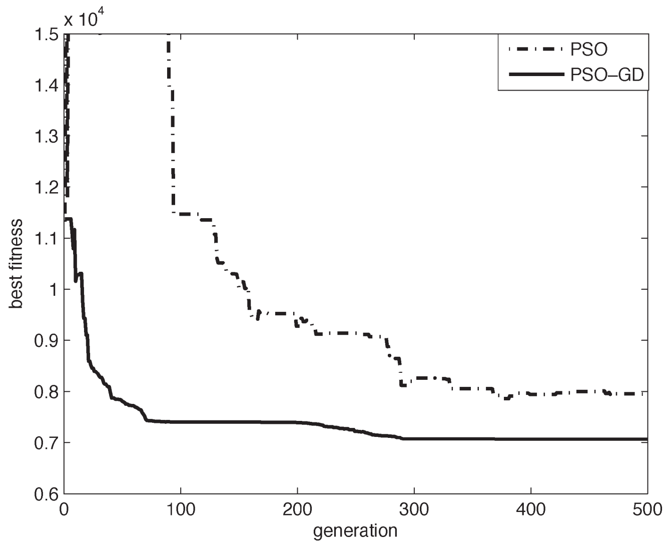

Fig. 10. Fitness of the best particle (best run) for problem G10 showing details of the first 500 generations.

For problem G7, the optimal solution is 24.306. The fitness value for the best particle is shown in Fig. 5 and in more detail in Fig. 6. From Table I, it can be observed that CPSO-GD finds a solution very close to the optimal solution, which is much better than the solution found using CPSO. In this case, CPSO-GD performed significantly better than CPSO in terms of the median as well as the mean. According to the results obtained by CPSO-GD, the solution found presents a relatively low standard deviation (0.7767).

For problem G9, the optimal solution is 680.630. The fitness value for the best particle is shown in Fig. 7 and in more detail in Fig. 8. In this case, according to Table I, one observes that CPSO finds the optimal solution and that CPSO-GD finds a very close value to the optimal. The solution found by CPSO presents a smaller standard deviation (0.0196) than the results obtained by CPSO-GD (0.1843).

For problem G10, the optimal solution is 7049.331. In this case, the penalty constant $r = 100$ does not provide a feasible solution due the tradeoff between the function value and the penalty. The small penalty value used was not enough to force particles to the feasible search space. Therefore, we carried out experiments increasing the penalty parameter and set it to $r = 400\,000$. The fitness value for the

TABLE II
RESULTS USING CPSO-GD AND CPSO (SWARM SIZE $P_1 = 50$, $P_2 = 30$)

Problem	Method	best	median	mean	std. dev.	worst
G1	CPSO	-12.003	-6.000	-6.733	1.856	-5.0
Optimal solution (−15.0)	CPSO-GD	**-15.0**	-14.997	-14.997	0.0012	-14.994
G7	CPSO	24.986	36.4960	110.261	161.5097	614.638
optimal solution (24.306)	CPSO-GD	**24.711**	25.623	25.709	0.672	27.166
G9	CPSO	680.627	680.635	680.637	0.0063	680.649
optimal solution (680.63)	CPSO-GD	680.678	680.834	680.8710	0.1484	681.371
G10	CPSO	7262.5	8927.7	10846	4472.3	23481
optimal solution (7049.331)	CPSO-GD	**7055.6**	8216.6	8464.2	1256.7	11458

TABLE III
STATISTIC COMPARISON BETWEEN CPSO-GD AND CPSO ($P_1 = P_2 = 30$) USING THE WILCOXON RANK SUM TEST

Problem	*p*	*h*	*zval*	*ranksum*
G1	1.8515e-11	1	6.7173	1365
G7	2.4833e-06	1	4.70	1234
G9	3.3342e-11	1	-6.631	466
G10	9.1404e-07	1	4.9093	1247

TABLE IV
STATISTIC COMPARISON BETWEEN CPSO-GD AND CPSO ($P_1 = 50$, $P_2 = 30$) USING THE WILCOXON RANK SUM TEST

Problem	*P*	*h*	*zval*	*ranksum*
G1	1.4344e-11	1	6.7544	1365
G7	3.9648e-08	1	5.4924	1287
G9	3.0161e-11	1	-6.645	465
G10	0.0183	1	2.3593	1075

best particle is shown in Fig. 9 and in more detail in Fig. 10. From Table I, we observe that increasing the penalty CPSO-GD finds a value of 7064.3 for the fitness of the best particle, which is very close to the optimal solution. The CPSO algorithm finds a value of 7610.0 for the best particle, which is considerably inferior. For this problem most of the algorithms proposed in the literature present difficulties due the nature of the constraints (all are nonlinear).

For the problems G1, G7, G9, and G10, we provide more details on the convergence behavior in Figs. 4, 6, 8, and 10, respectively. We can observe that for all these problems CPSO-GD converges much faster than CPSO, and obtains solutions closer to the optimal and presents a smaller standard deviation with exception of the problem G9 where CPSO provides slightly better results.

The same benchmarks problems have been simulated for a different population size in order to evaluate the influence of the population size. The first population is the main population since it is made up of the variable vector $\boldsymbol{x}$. Therefore, we increase the size of the first population to $P_1 = 50$ and have maintained the size of the second population $P_2 = 30$, since this population is made up of the Lagrange multipliers vector $\boldsymbol{\mu}$. The new simulation results for the benchmark problems are shown in Table II. From the results, we observe that for some problems a small improvement has occurred.

For the validation of significance of results and comparison between the CPSO and CPSO-GD algorithms, we carried out the Wilcoxon rank sum test[1] using the statistical toolbox provided in Matlab [44]. The test was performed using the results of 30 runs. For the hypothesis test, the h value indicates that the results are statistically significant at the 95% confidence level.

The statistic test of the results given in Tables I and II are provided in Tables III and IV, respectively. The Wilcoxon rank sum test indicates that the medians of CPSO-GD and CPSO are significantly different at the 95% confidence level, since h is equal to 1 in Tables III and IV. The Wilcoxon rank sum test confirms that CPSO-GD performs better than CPSO except for problem G9.

In order to compare our results, we have simulated the benchmark problems using a coevolutionary genetic algorithm (CGA). The population size was set to $P_1 = 50$ and $P_2 = 30$. Crossover probability was set to 0.35 and mutation probability to 0.02. The maximal number of generation was set to 2000. More details on the GA used can be found in [46]. The results using CGA are shown in Table V and the statistics in Table VI. From Table V, we observe that the best solution found by CPSO-GD and CGA are similar for problems G1 and G9, but CPSO-GD found better solutions for problems G7 and G10. The convergence of CGA occurred in the first 500 generations similar to CPSO-GD. Increasing the populations size of CGA and changing the probabilities of crossover and mutation did not seem to improve the quality of the final solution.

The performance of any method of evolutionary computation for constrained optimization problems depends on the EAs as well as the constraint-handling technique. Throughout our study, we focus on the optimization method PSO-GD, and have used the standard Lagrangian formulation. However, it would also be possible the application of other constraint-handling techniques.

VII. CONCLUSION

In this correspondence, a CPSO using the absolute value of the Gaussian distribution (CPSO-GD) has been presented for solving constrained optimization problems. The use of $\mathrm{abs}(N(0,1))$ for generating the stochastic coefficients of PSO seems to provide a good compromise between the probability of having a large number of small amplitudes around the current points (fine-tuning) and a small probability of having larger amplitudes, which may allow particles to move away from the current point and escape from local minima.

The CPSO-GD algorithm was compared with the standard CPSO (with constriction factor) and with a CGA on the benchmark constrained optimization problems. The simulation results showed that the algorithm CPSO-GD outperforms the standard CPSO and CGA as well. The CPSO-GD algorithm presents faster convergence and obtains solutions closer to the optimal. As part of future work, the CPSO-GD algorithm could be used with other more powerful methods to handle constraints, e.g., stochastic ranking.

[1]The Wilcoxon rank sum test performs a two-sided rank sum test of the hypothesis that two independent samples of data come from distributions with equal medians, and returns the p-value from the test. p is the probability of observing the given result, or one more extreme, by chance if the null hypothesis is true, i.e., the medians are equal. Small values of p cast doubt on the validity of the null hypothesis. ranksum contains the value of the rank sum statistic. If the sample size is large, then p is calculated using a normal approximation and zval contains the value of the normal statistics.

TABLE V
RESULTS USING CGA (POPULATION SIZE $P_1 = 50$, $P_2 = 30$)

Problem	best	median	mean	std. dev.	worst
G1 optimal solution (-15.0)	**-15.0**	-14.9982	-14.5216	1.006	-11.5411
G7 optimal solution (24.306)	25.3454	25.9360	25.9208	0.3248	27.2169
G9 optimal solution (680.63)	680.6706	681.4297	681.7100	1.0370	685.8794
G10 optimal solution (7049.331)	7095.5	7321.3	7350.4	182.131	7831.8

TABLE VI
STATISTIC COMPARISON BETWEEN CPSO-GD AND CGA USING THE WILCOXON RANK SUM TEST

Problem	*p*	*h*	*zval*	*ranksum*
G1	0.0351	1	2.2251	1058
G7	0.0824	0	-1.7372	1033
G9	8.1946e-007	1	4.9307	1249
G10	2.8389e-004	1	3.6296	1161

APPENDIX

Let us consider the velocity updating equation described in (9). Without loss of generality, we omit the indexes and consider a one-dimensional particle as follows:

$$\begin{aligned} v(t+1) = kv(t) + cU(0,1)\,[\boldsymbol{p}_{\text{best}} - x(t)] \\ + cU(0,1)\left[(\boldsymbol{p}_{\text{gbest}} - x(t)\right] \end{aligned} \quad (14)$$

where t stands for the time step. The stochastic coefficients for the local and global terms $cU(0,1)$ are a realization of a uniform random variable with support set in $c = [0; c]$. Then, for the first step, the velocity and position equation can be written, respectively, as

$$v(t+1) = kv(t) + c_1^{(t)}[\boldsymbol{p}_{\text{best}} - x(t)] + c_2^{(t)}\left[(\boldsymbol{p}_{\text{gbest}} - x(t)\right] \quad (15)$$

$$x(t+1) = x(t) + v(t+1) \quad (16)$$

and for the second step as

$$\begin{aligned} v(t+2) &= kv(t+1) + c_3^{(t+1)}\,[\boldsymbol{p}_{\text{best}} - x(t+1)] \\ &\quad + c_4^{(t+1)}\left[(\boldsymbol{p}_{\text{gbest}} - x(t+1)\right] \quad (17) \\ x(t+2) &= x(t+1) + v(t+2). \quad (18) \end{aligned}$$

The quantities $c_1^{(t)}$, $c_2^{(t)}$, $c_3^{(t+1)}$, $c_4^{(t+1)}$ are uniform random variables with support set in $c = [0; c]$. Inserting (15) and (16) into (17) we obtain for the velocity equation

$$\begin{aligned} v(t+2) = &\left[k - \left(c_3^{(t+1)} + c_4^{(t+1)}\right) + \frac{\left(c_3^{(t+1)} + c_4^{(t+1)}\right)}{\left(c_1^{(t)} + c_2^{(t)}\right)}\right] \\ &\times v(t+1) - k\frac{\left(c_3^{(t+1)} + c_4^{(t+1)}\right)}{\left(c_1^{(t)} + c_2^{(t)}\right)}v(t) \\ &+ \left[c_3^{(t+1)} - c_1^{(t)} \cdot \frac{\left(c_3^{(t+1)} + c_4^{(t+1)}\right)}{\left(c_1^{(t)} + c_2^{(t)}\right)}\right](\boldsymbol{p}_{\text{best}} - \boldsymbol{p}_{\text{gbest}}). \quad (19) \end{aligned}$$

Rearranging the terms in (19) and shifting the index, the difference equation for the velocity reads [43]

$$v(t+2) = Zv(t+1) - kQv(t) + W(\boldsymbol{p}_{\text{best}} - \boldsymbol{p}_{\text{gbest}}) \quad (20)$$

with

$$z = k - \left(c_3^{(t+1)} + c_4^{(t+1)}\right) + \frac{\left(c_3^{(t+1)} + c_4^{(t+1)}\right)}{\left(c_1^{(t)} + c_2^{(t)}\right)}$$

$$q = \frac{\left(c_3^{(t+1)} + c_4^{(t+1)}\right)}{\left(c_1^{(t)} + c_2^{(t)}\right)}$$

$$w = c_3^{(t+1)} - c_1^{(t)} . \frac{(c_3^{(t+1)} + c_4^{(t+1)})}{\left(c_1^{(t)} + c_2^{(t)}\right)}.$$

Furthermore, Z can be rewritten as $Z = (k - s + q)$ with $S = c_3^{(t+1)} + c_4^{(t+1)}$. The quantities z, q, and w are realizations of the random variables Z, Q, and W, respectively. Next, we calculate the mean value for these random variables. The following analysis is based on Clerc's work [43]. The calculation of the probability densities for the variables S and Q is quite easy, but for Z and W is quite complicated. Since the random variable S consists of a sum of two random uniform variables $s = c_3^{(t+1)} + c_4^{(t+1)}$, then its mean value is given by mean$(S) = c$.

For the random variable Q, the mean can be calculated considering that $Q = S_1/S_2$, which can be written as a product of two independent random variables, i.e., $Q = S_1(1/S_2)$, with $1/S_2 = 1/(U_1 + U_2)$. The probability density function for S_2 is given by

$$f(u) = \begin{cases} \frac{2}{u^2} - \frac{1}{u^3}, & \text{if } u \in [0.5;\ 1] \\ \frac{1}{u^3}, & \text{if } u > 1 \end{cases}$$

and the mean of S_2 can be computed as

$$\text{mean}(S_2) = \int_{0.5}^{1} \left(\frac{2}{u} - \frac{1}{u^2}\right) du + \int_{1}^{\infty} \frac{1}{u^2} = 2\ln(2).$$

Therefore, mean$(Q) = 2\ln(2) = 1.386$.

The random variable $Z = k - S + Q$ can be written as

$$Z = k - \left(U_3^{(t+1)} + U_4^{(t+1)}\right) + \left(c - \frac{1}{U_1^{(t)} + U_2^{(t)}}\right).$$

The main difficulty is that the variables S and Q are not independent. The minimum value assumed by the random variable Z is $k + 1 - 2c$. In order to allow negative value of z, a possible choice for c is the relationship [43]

$$c = \frac{k+1}{2}.$$

The mean value of Z is computed as mean$(Z) = k - c + 2\ln 2$.

In PSO two differences are averaged, each with its own uniformly distributed random numbers. According to the mean values calculated for Q and Z, a possible choice for the constriction coefficients is given as follows:

$$k = \frac{1}{2 \ln 2} = 0.72$$

$$c = \frac{k+1}{2} = 0.86.$$

Therefore, the interval [0.72; 0.86] contains possible values for the constriction coefficients. Details can be found in [43].

Acknowledgment

The authors would like to thank D. Cook for the review process. They are deeply grateful to the anonymous referees for the valuable comments which improved the initial manuscript very much. They also thank M. Clerc for information on Tribes.

References

[1] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evol. Comput.*, vol. 4, no. 1, pp. 1–32, 1996.

[2] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, Australia, 1995, pp. 1941–1948.

[3] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*. San Francisco, CA: Morgan Kaufmann, 2001.

[4] G. Coath and S. K. Halgamuge, "A comparison of constraint handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems," in *Proc. IEEE CEC*, Canberra, Australia, Dec. 2003, vol. 4, pp. 2419–2425.

[5] A. I. El-Gallad, M. E. El-Hawary, and A. A. Sallam, "Swarming of intelligent particles for solving the nonlinear constrained optimization problem," *Eng. Intell. Syst. Electr. Eng. Commun.*, vol. 9, no. 3, pp. 155–163, Sep. 2001.

[6] X. Hu, R. C. Eberhart, and Y. Shi, "Engineering optimization with particle swarm," in *Proc. IEEE Swarm Intell. Symp.*, Indianapolis, IN, Apr. 2003, pp. 53–57.

[7] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method for constrained optimization problems," in *Intelligent Technologies—Theory and Applications: New Trends in Intelligent Technologies*, vol. 76, P. Sincak, J. Vascak, V. Kvasnicka, and J. Pospicha, Eds. Amsterdam, The Netherlands: IOS Press, 2002, pp. 214–220.

[8] G. Toscano-Pulido and C. A. Coello Coello, "A constraint handling mechanism for particle swarm optimization," in *Proc. IEEE CEC*, Portland, OR, Jun. 2004, vol. 2, pp. 1396–1403.

[9] W.-J. Zhang, X.-F. Xie, and D.-C. Bi, "Handling boundary constraints for numerical optimization by particle swarm flying in periodic search space," in *Proc. IEEE CEC*, Portland, OR, Jun. 2004, vol. 2, pp. 2307–2311.

[10] Y. Shi and R. C. Eberhart, "Parameter selection in PSO optimization," in *Proc. 7th Annu. Conf. Evol. Program.*, San Diego, CA, 1998, pp. 25–27.

[11] ——, "Empirical study of particle swarm optimization," in *Proc. IEEE CEC*, Washington, DC, 1999, vol. 3, pp. 1945–1950.

[12] ——, "A modified particle swarm optimizer," in *Proc. IEEE CEC*, Anchorage, AK, 1998, pp. 69–73.

[13] F. van den Bergh, "An analysis of particle swarm optimizers," Ph.D. dissertation, Dept. Comput. Sci., Univ. Pretoria, Pretoria, South Africa, 2002.

[14] 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.

[15] I. C. Trelea, "The particle swarm optimization algorithm: Convergence analysis and parameter selection," *Inf. Process. Lett.*, vol. 85, no. 6, pp. 317–325, 2003.

[16] L. S. Coelho and R. A. Krohling, "Predictive controller tuning using modified particle swarm optimisation based on Cauchy and Gaussian distributions," in *Proc. VI Brazilian Conf. Neural Netw.*, Sao Paulo, Brazil, Jun. 2003, pp. 1–6. In Portuguese.

[17] ——, "Predictive controller tuning using modified particle swarm optimisation based on Cauchy and Gaussian distributions," in *Proc. 8th Online World Conf. Soft Comput. Ind. Appl.*, Dortmund, Germany, Sep. 2003, pp. 7–12.

[18] B. R. Secrest and G. B. Lamont, "Visualizing particle swarm optimization—Gaussian particle swarm optimization," in *Proc. IEEE Swarm Intell. Symp.*, Indianapolis, IN, 2003, pp. 198–204.

[19] J. Kennedy, "Bare bones particle swarms," in *Proc. IEEE Swarm Intell. Symp.*, Indianapolis, IN, 2003, pp. 80–87.

[20] N. Higashi and H. Iba, "Particle swarm optimization with Gaussian mutation," in *Proc. IEEE Swarm Intell. Symp.*, Indianapolis, IN, 2003, pp. 72–79.

[21] A. Stacey, M. Jancic, and I. Grundy, "Particle swarm optimization with mutation," in *Proc. IEEE CEC*, Canberra, Australia, 2003, pp. 1425–1430.

[22] V. Miranda and N. Fonseca, "EPSO—Best of two worlds meta-heuristic applied to power system problems," in *Proc. IEEE CEC*, Honolulu, HI, 2002, pp. 1080–1085.

[23] C. Wei, Z. He, Y. Zheng, and W. Pi, "Swarm directions embedded in fast evolutionary programming," in *Proc. IEEE CEC*, Honolulu, HI, 2002, pp. 1278–1283.

[24] S. C. Esquivel and C. A. C. Coello, "On the use of particle swarm optimization with multimodal functions," in *Proc. IEEE CEC*, Canberra, Australia, 2003, pp. 1130–1136.

[25] R. A. Krohling, "Gaussian swarm: A novel particle swarm optimization algorithm," in *Proc. IEEE Conf. CIS*, Singapore, 2004, pp. 372–376.

[26] M. Clerc. (2004). TRIBES, a parameter free particle swarm optimizer. [Online]. Available: http://www.particleswarm.info

[27] G. C. Onwubolu, "TRIBES application to the flow shop scheduling problem," in *New Optimization Techniques in Engineering*, vol. 141, G. C. Onwubolu and B. V. Babu, Eds. Berlin, Germany: Springer-Verlag, 2004, pp. 517–536.

[28] R. K Ursem, "Diversity-guided evolutionary algorithms," in *Proc. Parallel Problem Solving Nature Conf. VII*, Granada, Spain, 2001, pp. 462–471.

[29] X. Yao and Y. Liu, "Fast evolutionary programming," in *Proc. 5th Annu. Conf. Evol. Program.*, San Diego, CA, 1996, pp. 451–460.

[30] K. Chellapilla, "Combining mutation operators in evolutionary programming," *IEEE Trans. Evol. Comput.*, vol. 2, no. 3, pp. 91–96, Sep. 1998.

[31] D. Du and P. M. Pardalos, *Minimax and Applications*. Norwell, MA: Kluwer, 1995.

[32] W. D. Hillis, "Coevolving parasites improve simulated evolution as an optimization procedure," *Phys. D*, vol. 42, no. 1–3, pp. 228–234, Jun. 1990.

[33] J. Paredis, "Steps towards coevolutionary classification neural networks," in *Artificial Life IV*. Cambridge, MA: MIT Press, 1994, pp. 359–365.

[34] C. D. Rosin and R. K. Belew, "Methods for competitive coevolution: Finding opponents worth beating," in *Proc. 6th Int. Conf. Genetic Algorithms Appl.*, Pittsburgh, PA, 1995, pp. 373–380.

[35] ——, "New methods for competitive coevolution," *Evol. Comput.*, vol. 5, no. 1, pp. 1–29, 1996.

[36] H. J. C. Barbosa, "A genetic algorithm for min–max problems," in *Proc. 1st Int. Conf. Evol. Comput. Appl.*, Moscow, Russia, 1996, pp. 99–109.

[37] ——, "A coevolutionary genetic algorithm for constrained optimization," in *Proc. IEEE CEC*, Washington, DC, 1999, pp. 1605–1611.

[38] M. J. Tahk and B.-C. Sun, "Coevolutionary augmented lagrangian methods for constrained optimization," *IEEE Trans. Evol. Comput.*, vol. 4, no. 2, pp. 114–124, Jul. 2000.

[39] Y. Shi and R. A. Krohling, "Co-evolutionary particle swarm optimization to solving min–max problems," in *Proc. IEEE CEC*, Honolulu, HI, 2002, pp. 1682–1687.

[40] R. A. Krohling, F. Hoffmann, and L. S. Coelho, "Co-evolutionary particle swarm optimization for min–max problems using Gaussian distribution," in *Proc. IEEE CEC*, Portland, OR, 2004, pp. 959–964.

[41] E. C. Laskari, K. E. Parsopoulos, and M. N. Vrahatis, "Particle swarm optimization for minimax problems," in *Proc. IEEE CEC*, Honolulu, HI, 2002, pp. 1576–1581.

[42] M. Clerc, Personal communication on TRIBES, Aug. 2005.

[43] ——, (2005, Jun.), Stagnation analysis in particle swarm optimization or what happens when nothing happens, Tech. Rep. [Online]. Available: http://clerc.maurice.free.fr/pso/

[44] *Matlab Statistical Toolbox*, The Mathworks, Natick, MA, 2005. Version 7.0.

[45] J. Kennedy, "Dyanamic–probabilistic particle swarms," in *Proc. GECCO*, Washington, DC, 2005, pp. 201–207.

[46] R. A. Krohling and J. P. Rey, "Design of optimal disturbance rejection PID controllers using genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 5, no. 1, pp. 78–82, Feb. 2001.