Proceedings of the 2004 IEEE
Conference on Cybernetics and Intelligent Systems
Singapore, 1-3 December, 2004

Gaussian Swarm: A Novel Particle Swarm Optimization Algorithm

Renato A. Krohling
Lehrstuhl Elektrische Steuerung und Regelung (ESR)
Fakultät für Elektrotechnik und Informationstechnik
Universität Dortmund
D-44221 Dortmund, Germany
E-mail: renato.krohling@uni-dortmund.de

***Abstract*—In this paper, a novel particle swarm optimization algorithm based on the Gaussian probability distribution is proposed. The standard Particle Swarm optimization (PSO) algorithm has some parameters that need to be specified before using the algorithm, e.g., the accelerating constants c_1 and c_2, the inertia weight w, the maximum velocity *Vmax*, and the number of particles of the swarm. The purpose of this work is the development of an algorithm based on the Gaussian distribution, which improves the convergence ability of PSO without the necessity of tuning these parameters. The only parameter to be specified by the user is the number of particles. The Gaussian PSO algorithm was tested on a suite of well-known benchmark functions and the results were compared with the results of the standard PSO algorithm. The simulation results shows that the Gaussian Swarm outperforms the standard one.**

Keywords: Particle Swarm Optimization, Gaussian distribution, nonlinear optimization.

I. INTRODUCTION

Particle Swarm Optimization (PSO) originally developed by Kennedy and Eberhart [1], [2] is a population-based algorithm. PSO is initialized with a population of candidate solutions. Each candidate solution in PSO, called *particle*, has associated a randomized velocity, moves 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, *pbest*. 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. The PSO has been found to be robust and fast in solving nonlinear, non-differentiable, multimodal optimization problems.

In the last few years, there is a growing interest in PSO; several heuristics have been developed to improve the convergence performance of the algorithm and set up the optimal parameters of PSO [3], [4]. Initially, the PSO parameters, e.g., the acceleration constants c_1 and c_2 was set to 2.0 for almost all applications. The maximum velocity *Vmax* was often set to about 10-20% of the dynamic range of the variable along each dimension. Like other evolutionary algorithms, the population size selected is problem-dependent. Population sizes of 20-50 were quite most common. Most approaches use a uniform probability distribution to generate random numbers. However the difficulty to obtain fine tuning of the solution and escape from local minima using a uniform distribution was made evident in [5], [6] where it was proposed the use of Cauchy and the Gaussian distributions to generate random numbers to updating the velocity equation of PSO, which was motivated by studies of mutation operators in fast evolutionary programming [7], [8]. In [9] was used a mutation operator in PSO, where the position of the particles are mutated using a Gaussian distribution. In [10] was developed a new approach based on Gaussian swarm but for visualization purpose.

Following a more theoretical line, Clerc and Kennedy [11] have introduced a *constriction factor*, which may be necessary to ensure convergence of the PSO, whereas the parameters c_1 and c_2 was set to 1.49. Recently, in [12] was presented new results about the convergence of PSO, where it was analyzed the trajectory of a deterministic particle in one dimension using dynamic systems theory. However, the deterministic approach proposed in [12] might not be straightforward generalized for the analysis of trajectory of stochastic particles in multi-dimensional search space.

In this paper, a novel approach to updating the velocity equation based on the Gaussian distribution in PSO algorithm is proposed. The inertia weight, the maximum velocity of the particles and the accelerating constants c_1 and c_2 are not more necessary to be specified. The approach was empirically examined with a suite of well-known functions that are frequently used for testing evolutionary algorithms, and also for testing PSO as well. The numerical simulation results demonstrate that the strategy can significantly improve the PSO convergence performance. The rest of the paper is organized as follows: In section 2, the standard PSO is described. In section 3, the novel Gaussian swarm is presented. Section 4 gives the simulation results for some benchmarks optimization problems followed by conclusions in section 5.

II. PARTICLE SWARM OPTIMIZATION

The PSO algorithm is described in the following [1], [2]:

(i) Initialize a population of particles with random positions and velocities in the n-dimensional problem space.

(ii) For each particle, evaluate its fitness value.

(iii) Compare each particle's fitness evaluation with the current particle's *pbest*. If current value is better than *pbest*, set its *pbest* value to the current value and the *pbest* location to the current location in n-dimensional space.

(iv) Compare fitness evaluation with the population's overall previous best. If current value is better than *gbest*, then reset *gbest* to the current particle's array index and value.

(v) Change the velocity and position of the particle according (1) and (2), respectively:

$$v_i = wv_i + c_1 rand(p_i - x_i) + c_2 Rand(p_g - x_i) \tag{1}$$

$$x_{i+1} = (x_i + v_i). \tag{2}$$

(iv) Loop to step (ii) until a stopping criterion is met, usually a maximum number of iterations (generations).

The vector $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T$ stands for the position of the i-th particle, $v_i = [v_{i1}, v_{i2}, \ldots, v_{in}]^T$ stands for the velocity of the i-th particle and $p_i = [p_{i1}, p_{i2}, \ldots, p_{in}]^T$ represents the best previous position (the position giving the best fitness value) of the i-th particle. The index g represents the index of the best particle among all the particles in the swarm. 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. Particles' velocities along each dimension are clamped to a maximum velocity *Vmax*. If the sum of accelerations causes the velocity on that dimension to exceed *Vmax*, which is a parameter specified by the user, then the velocity on that dimension is limited to *Vmax*.

The inertia weight w represents the degree of the momentum of the particles. The second part is the "cognition" part, which represents the independent behaviour 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 *pbest* and *gbest* positions.

In [11] a *constriction factor K* was introduced, and the velocity updating (1) has been substituted by

$$v_i = K[v_i + c_1 rand\,(p_i - x_i) + c_2 Rand\,(p_g - x_i)]. \tag{3}$$

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{4}$$

where $\varphi = c_1 + c_2$, $\varphi > 4$ and K is a function of c_1 and c_2. A detailed discussion of the constriction factor is beyond the scope of this paper. For more details, the reader is referred to [11]. Usually, when the constriction factor is used, φ is set to 4.1 ($c_1 = c_2 = 2.05$), and the constriction factor K is 0.729. Each of the two coefficients of the $(p_i - x_i)$ terms is calculated by multiplying 0.729*2.05=1.49445 (times a random number between 0 and 1) with expected value equal 0.5, provides a mean value 0.729.

New results presented in [12] based on the theory of dynamic systems for analysis of a particle trajectory have been carried out with a different parameter set (w=0.6, and $c_1 = c_2$ = 1.7). The expected value for each of the two $(p_i - x_i)$ terms is calculated by multiplying 1.7*0.5=0.85, since the expected value of a uniform random number between 0 and 1 is 0.5. The results compared to that proposed in [11] showed a slightly superior performance. In the next section, we present a simplified swarm without momentum term, where the coefficients of the two $(p_i - x_i)$ terms do not need to be specified, but are generated automatically according to the Gaussian distribution.

III. GAUSSIAN SWARM: A PARTICLE SWARM OPTIMIZATION USING GAUSSIAN DISTRIBUTION

Observing that the expected values for the two $(p_i - x_i)$ terms are 0.729 [11] and 0.85 [12], it is proposed a probability distribution that generates random number with expected values between the limits [0.729 0.85]. It should be pointed out that the coefficients of the two $(p_i - x_i)$ terms must be positive for convergence. A candidate distribution is the Gaussian probability density function given by

$$g(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}. \tag{5}$$

A Numerical implementation that provides positive random numbers with mean value 0.8 and standard deviation 0.6 can be generated using a Gaussian distribution $N(0,1)$ as shown in Fig. 1. The goal consists of generating the coefficients of the $(p_i - x_i)$ terms according to the Gaussian probability distribution. This can be accomplished by defining the velocity equation as

$$v_i = |randn|(p_i - x_i) + |Randn|(p_g - x_i). \tag{6}$$

where $|randn|$ and $|Randn|$ are positive random numbers generated according to the absolute value of the Gaussian probability distribution, i.e., $abs[N(0,1)]$. For comparison purposes only, the standard deviation using a uniform distribution, as usually employed in standard PSO, is 0.28.

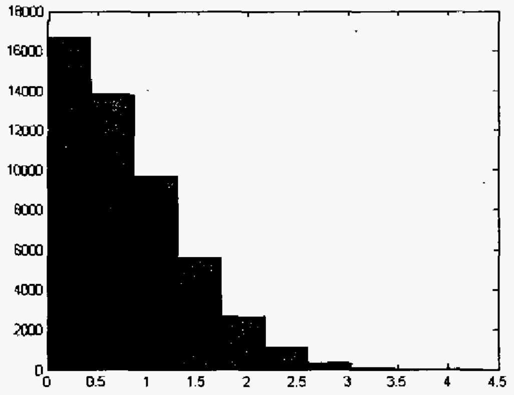

Fig. 1 Histogram of the random numbers generated using $abs[N(0,1)]$ results in random numbers with approximately expected value 0.8 and standard deviation 0.6.

The larger standard deviation provided by the Gaussian distribution compared to the uniform one, improves the ability of PSO to escape from local minima. This novel approach is completely different from that proposed in previous works presented in [5], [6], where it was used a truncated Gaussian distribution to generate random numbers between minus one and plus one, and then mapped to [0 1]. Additionally, in [5], [6] was used the usual velocity updating (1). Based on simulations studies, it was observed that the momentum term with inertia weight *w* could be set to zero, because it might not improve the convergence of the Gaussian PSO. So, it is no more necessary to specify the maximum velocity *Vmax* of the swarm.

To the best of our knowledge, it is the first time that an asymmetric probability distribution has been introduced in the context of PSO. For more mathematical details see Appendix.

IV. SIMULATION RESULTS

A. *Benchmark functions*

The Gaussian PSO has been tested on three different kinds of optimization problems: a) unimodal function, b) functions with a few local minima and c) functions with many local minima.

a) Unimodal function

The Rosenbrock function $f_1(\boldsymbol{x})$ is given by

$$f_1(\boldsymbol{x}) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$$

with $-30 \le x_i \le 30$, (i=1,2).

The global minimum is $f_1(\boldsymbol{x}^*) = 0$, and $\boldsymbol{x}^* = (1,1)$.

b) Functions with a few local minima

The Six-hump camel back function $f_2(\boldsymbol{x})$ is given by

$$f_2(\boldsymbol{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$

with $-2 \le x_i \le 2$, (i=1,2).

The global minimum is $f_2(\boldsymbol{x}^*) = -1.0316$, and the optimal solution is $\boldsymbol{x}^*$ = (-0.0898,0.7126), and (0.0898,-0.7126).

The Goldstein-Price function $f_3(\boldsymbol{x})$ is given by

$$f_3(\boldsymbol{x}) = [1 + (x_1 + x_2 + 1)^2.(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$$
$$.[30 + (2x_1 - 3x_2)^2.\ (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$

with $-5 \le x_i \le 5$, (i=1,2).

The global minimum is $f_3(\boldsymbol{x}^*) = 3$, and $\boldsymbol{x}^* = (0,-1)$.

c) Functions with many local minima

The Rastrigin function $f_4(\boldsymbol{x})$ is given by

$$f_4(\boldsymbol{x}) = \sum_{i=1}^{n} \left\{ x_i^2 - 10\cos(2\pi x_i) + 10 \right\}$$

with $-5.12 \le x_i \le 5.12$, (i=1,...,n).

The global minimum is $f_4(\boldsymbol{x}^*) = 0$, and $\boldsymbol{x}^* = (0,...,0)$.

The Griewank function $f_5(\boldsymbol{x})$ is given by

$$f_5(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$$

with $-600 \le x_i \le 600$, (i=1,...,n).

The global minimum is $f_5(\boldsymbol{x}^*) = 0$, and $\boldsymbol{x}^* = (0,...,0)$. This function has many local minima, so that it is very difficult to find the global minimum.

B. *Results and Discussion*

The PSO algorithms were implemented using Matlab [13]. During the numerical experiments, both the Gaussian and the standard PSO algorithm were run with an initial population of random values that was created using a uniform probability distribution. Such running trials were repeated for each of the chosen function for 10 times. All the running trials were carried out with a population of 30 particles. The accelerating constants c_1 and c_2 for the standard PSO are chosen as $c_1 = c_2 = 2$. The results for the Rosenbrock function $f_1(\boldsymbol{x})$, the Six-hump camel back function $f_2(\boldsymbol{x})$, the Goldstein-Price function $f_3(\boldsymbol{x})$, the Rastrigin function $f_4(\boldsymbol{x})$, and the Griewank function $f_5(\boldsymbol{x})$ are shown in Figs. 2, 3, 4, 5, 6 respectively.

From the results obtained, it can be observed that for the five benchmark minimization problems, the Gaussian PSO algorithm shows considerably better convergence than the standard PSO. For the functions studied, the Gaussian PSO algorithm converged to the minimum at earlier generations than the standard PSO algorithm. The standard PSO algorithm may rapidly stagnate, and the solution no longer improves anymore, while the Gaussian PSO algorithm can still search solution progressively till the global optimum is found. This is especially important for functions with many local minima, like the Griewank function, where it was observed that the standard PSO get trapped into a local minimum and can not escape. The Gaussian PSO algorithm has a higher ability to escape from local minima and therefore able to obtain the global optimum. But due the high number of local minima of the Griewank function the Gaussian PSO might also present difficulties and the weight term might help to escape from local minima as shown in previous works [3], [4].

The results presented are still preliminary, but show clearly that the Gaussian PSO algorithm is effective to find the global optimum of the benchmarks investigated. The proposed approach has been applied to solve constrained optimizations problems formulated as min-max problems with very good results. Work in progress is investigating the potential of Gaussian PSO applied to optimization problems with many local minima in higher dimensions, e.g., the Griewank function and the Rastrigin function. The first results are very promising and will be reported elsewhere in future.

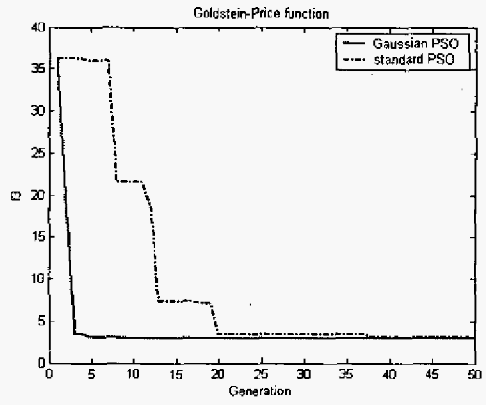

Fig. 4 Result for the Goldstein-Price function using Gaussian and standard PSO.

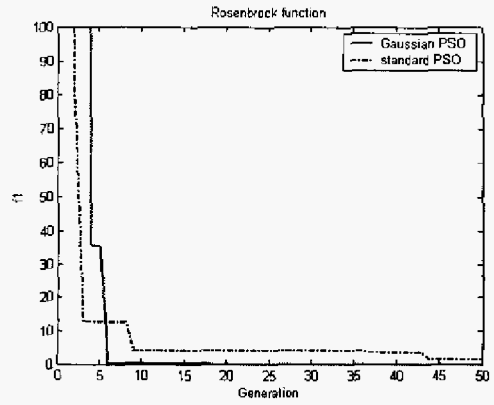

Fig. 2 Result for the Rosenbrock function using Gaussian and standard PSO.

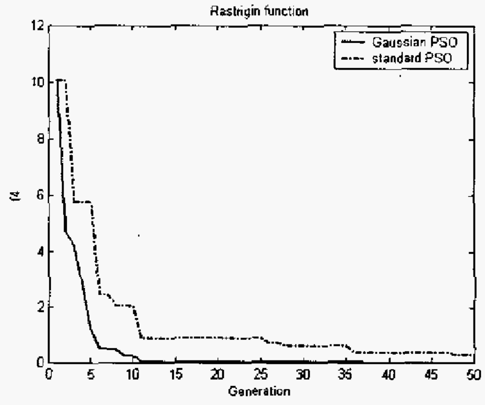

Fig. 5 Result for the Rastrigin function using Gaussian and standard PSO.

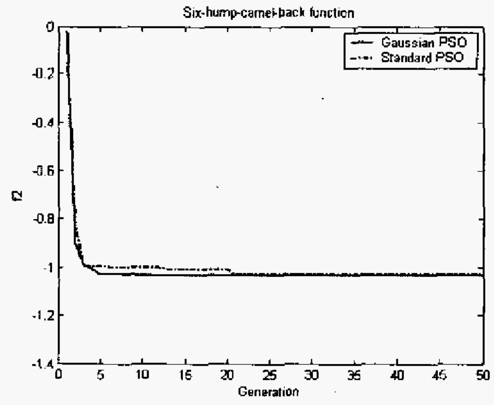

Fig. 3 Result for the Six-hump camel back function using Gaussian and standard PSO.

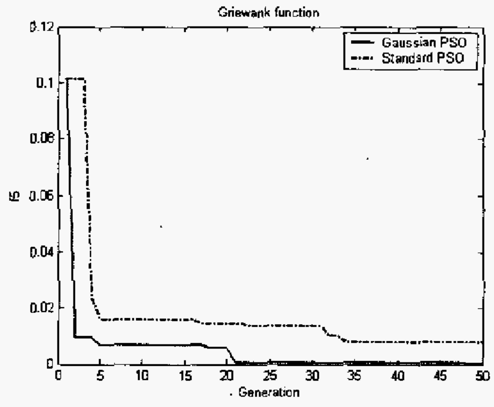

Fig. 6 Result for the Griewank function using Gaussian and standard PSO.

V. CONCLUSIONS

In this paper, a novel Gaussian swarm has been proposed. PSO is *de facto* one of the most promising contemporary optimization algorithms in general because its convergence, effectiveness, robustness, and simplicity of implementation. These features of the algorithm have attracted its attention to solve nonlinear, non-differentiable, multimodal optimization problems. However, some parameters of PSO need to be specified by the user before using the algorithm. The goal of this work was to change the velocity updating equation of PSO in order that the coefficients of the two $(p_i - x_i)$ terms are automatically generated by using a Gaussian probability distribution. So, there is no more need to specify the parameters accelerating constants c_1 and c_2. Furthermore, using the Gaussian PSO the inertia weight w was set to zero and therefore the maximum velocity *Vmax* is no more necessary to be defined. So, the only parameter to be specified by the user is the number of particles. Additionally, the use of a Gaussian distribution improves the convergence ability of PSO. The Gaussian PSO algorithm was studied and compared with the standard PSO algorithm on a suite of well-known benchmark functions. The simulation results indicate that the proposed algorithm using Gaussian distribution outperforms the standard PSO. The results showed that the Gaussian PSO algorithm converges rapidly and has more ability to escape from local minima than the standard PSO. On the one hand, future investigations are being made on applying the Gaussian PSO algorithm to control systems optimization problems. On the other hand, some work has been carried out on stochastic convergence of the Gaussian swarm.

APPENDIX

Let the Gaussian probability distribution $g(x)$ given by

$$g(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

with expected value 1 and variance 0, i.e. N(0,1). Then the expected value $E(x)$ of the absolute of N(0,1) can be calculated by

$$E(x) = \mu = \frac{1}{\sqrt{2\pi}} \int_0^\infty 2x e^{-\frac{x^2}{2}} dx = 0.798$$

and the variance by

$$V(x) = \sigma^2 = E\left(\left(x - E(x)\right)^2\right) = \frac{1}{\sqrt{2\pi}} \int_0^\infty (x-\mu)^2 2x\, e^{-\frac{x^2}{2}} dx$$

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

Therefore, the standard deviation is $\sigma = 0.60$. The skewness which is a measure of the asymmetry of the distribution is calculated by $(\mu^3/\sigma^3) = 2.37$ (positive tail). The kurtosis, which is a measure of the peakedness of the distribution is calculated by $(\mu^4/\sigma^4) - 3 = 0.16$.

ACKNOLEDGMENT

R. A. Krohling thanks Dr. F. Hoffmann (Lehrstuhl ESR) for supporting this research work.

REFERENCES

[1] J. Kennedy, and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks* (Perth, Australia), IEEE Service Center, Piscataway, NJ, IV: pp. 1941-1948, 1995.

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

[3] Y. Shi, and R. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation*, Vol. 3, pp. 1945-1950, 1999.

[4] Y. Shi, and R. Eberhart, "Parameter selection in PSO optimization," in *Procedings of the 7th Annual Conference on Evolutionary Programming*, San Diego, USA, pp. 25-27, March 1998.

[5] L. S. Coelho, and R. A. Krohling, "Predictive controller tuning using modified particle swarm optimisation based on Cauchy and Gaussian distributions," in *Proceedings of the 8th On-line World Conference on Soft Computing in Industrial Applications*, WSC8, 2003.

[6] R. A. Krohling, F. Hoffmann, L. S. Coelho, "Co-evolutionary particle swarm optimization for min-max problems using Gaussian distribution," in *Proceedings of the 2004 Congress on Evolutionary Computation*, Portland, Oregon USA, pp. 959-964, 2004.

[7] X. Yao, and Y. Liu, "Fast evolutionary programming", in *Proceedings of 5th Annual Conference on Evolutionary Programming*, San Diego, CA, pp. 451-460, 1996.

[8] K. Chellapilla, "Combining mutation operators in evolutionary programming," *IEEE Trans. on Evolutionary Computation*, vol. 2, no.3, pp. 91-96, 1998.

[9] N. Higashi, and H. Iba, "Particle swarm optimization with Gaussian mutation," in *Proceedings of the IEEE Swarm Intelligence Symposium*, Indianapolis, IN, pp. 72-79, 2003.

[10] B.R. Secrest, and G.B. Lamont, "Visualizing particle swarm optimization - Gaussian particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium*, Indianapolis, IN, pp. 198-204, 2003.

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

[12] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, no. 6, pp. 317-325, 2003.

[13] B. Birge "PSOt - a particle swarm optimization toolbox for use with Matlab," in *Proceedings of the IEEE Swarm Intelligence Symposium*, Indianapolis, IN, pp. 182-186, 2003.