Limiting the Velocity in Particle Swarm Optimization Using a Geometric Series

Julio Barrera
CINVESTAV-IPN
Departamento de Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D.F. 07360, MEXICO
julio.barrera@gmail.com

Carlos A Coello Coello*
CINVESTAV-IPN
Departamento de Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D.F. 07360, MEXICO
ccoello@cs.cinvestav.mx

ABSTRACT

Since the introduction of the particle swarm optimization (PSO) algorithm, a considerable amount of research has been devoted to devise mechanisms that can control its possible premature convergence. The most common approach to deal with premature convergence in PSO consists of controlling (e.g., by limiting) the velocity of a particle. In this paper, we present a method that consists of limiting the velocity of a particle using the elements of a sequence of a geometric series. This approach is not only simplest than the current available methods, but also presents competitive results, and even better convergence in some cases, than two other PSO-based approaches. Additionally, the proposed approach provides more flexibility to balance between exploration or exploitation, through the tuning of a single parameter.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence—*Problem Solving, Control Methods, and Search*

General Terms

Algorithms

Keywords

particle swarm optimization, velocity control, convergence

1. INTRODUCTION

The Particle Swarm Optimization (PSO) algorithm was introduced by Kennedy and Eberhart in 1995 [2] and, since then, it has gained increasing importance. The PSO algorithm is based on the social behavior of some species, such as birds in a flock that fly looking for food. A problem with the PSO algorithm is that, in its simpler form, the velocity of a particle is accumulated and tends to increase and diverge causing poor convergence, or even no convergence at all. To solve this problem, two main methods have been proposed: the Inertia Weight model from Shi and Eberhart [6] and the Constriction Factor model from Clerc and Kennedy [1]. However, these two methods give rise to another problem: premature convergence. This is normally caused by the tendency of the two above PSO models to decrease the velocity of a particle too fast.

*The second author is also affiliated to the UMI-LAFMIA 3175 CNRS.

Copyright is held by the author/owner(s).
GECCO'09, July 8–12, 2009, Montréal, Québec, Canada.
ACM 978-1-60558-325-9/09/07.

2. THE SEQUENCE BOUND METHOD

Our proposed method consists of limiting the velocity of the particles. Such velocity limit is computed at each generation according to the corresponding element of a sequence of terms obtained from a geometric series. A geometric series S [4] is a sum of algebraic terms and is given by the Equation (1).

$$S = \sum_{m=0}^{\infty} a_c r^m \tag{1}$$

The *geometric series* S converges provided that $|r| < 1$ and a_c is a constant value. The ordered list of terms $a_c r^m$ of a series with $m = 0..\infty$ is called a *sequence*. A known result is that, if the geometric series defined in Equation (1) converges, then the elements of its sequence converge to zero, i.e., $a_c r^m \to 0$ as $m \to \infty$. More information and results on geometric series can be found in [4].

We construct a limit for the velocities as follows: at the iteration t, the ith coordinate of the velocity of a particle is bounded by the term $r^t l_i$, where r is a constant value with $0 < r < 1$ in order to guarantee the convergence to zero of the bound, and $l_i = |x_{max_i} - x_{min_i}|$ is the longitude of the search space in the ith coordinate with $i = 1 \ldots n$. To compute the velocity and position of a particle we use the simplest version of PSO, which is defined by Equations (2) and (3).

$$v_{t+1} = v_t + c_1 r_1 (p_g - p_t) + c_2 r_2 (p_l - p_t) \tag{2}$$

$$p_{t+1} = p_t + v_{t+1} \tag{3}$$

After computing v_{t+1} and before computing x_{t+1}, we limit the velocity of each particle with the term $r^t l_i$. The Sequence Bound Particle Swarm Optimization (SBPSO) method is described in the algorithm of Figure 1.

In general, if we set the value of the parameter r closer to 1, the limit of the velocity is decreased more slowly. This benefits the exploration of the search space, and allows us to guarantee that the velocity limit tends to zero.

```
swarm = generate_swarm(ranges, number_of_particles)
evaluate(swarm)
for i from 1 to generations
    compute_velocities(swarm)
    bounds = compute_bounds(ranges, i)
    apply_bounds(swarm, bounds)
    update_positions(swarm)
    evaluate(swarm)
endfor
```

Figure 1: The Sequence Bound Particle Swarm Optimization Algorithm.

3. EXPERIMENTS AND RESULTS

For assessing the performance of our proposed SBPSO algorithm, we decided to compare it with respect to the Comprehensive Learning Particle Swarm Optimizer (CLPSO) reported in [3], and with respect to the Inertia Weight Particle Swarm Optimizer (IWPSO) reported in [5].

Our test suite consists of five problems that have been commonly used to validate mechanisms that prevent premature convergence in PSO algorithms: the Sphere, Rosenbrock, Ranstrigin, Griewank, and Ackley. They are described in Equations (4) to (8).

$$f_1 = \sum_{i=1}^{n} x_i^2 \tag{4}$$

$$f_2 = \sum_{i=1}^{n-1} [(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)] \tag{5}$$

$$f_3 = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)) \tag{6}$$

$$f_4 = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{7}$$

$$f_5 = 20 + e - 20e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} \tag{8}$$

In all the experiments that we performed, the number of dimensions was set to 30. We adopted a setup similar to the one described in [3]: P_{ci} is computed in the range $[0.05, 0.5]$ using the empirical relation indicated therein. The number of iterations for each experiment was 5000, we used a swarm of 40 particles, an initial inertia value $\omega_i = 0.9$ and final inertia value of $\omega_f = 0.4$, the learning constants had equal values of $C_1 = C_2 = 1.49445$, and in the case of CLPSO a refreshing gap $m = 7$ was adopted. For our proposed SBPSO, the value of the parameter r was set to 0.998. The initialization, search space ranges and limits for the velocity and position of the particles adopted for the PSO algorithms are shown in Table 1 and are the same as in [3].

Each experiment has been repeated 100 times, in order to obtain statistical measures. Table 2 shows the mean of the best values obtained after the 5000 iterations.

From Table 2, we can observe that our proposed SBPSO method obtained the best results in two problems. However, it is worth noting that our proposed SBPSO approach shows better results than IWPSO in all the test problems. Also, except for f_4, the results obtained by our SBPSO in two of the three problems in which it is outperformed by CLPSO, are competitive.

Table 1: Search and Initialization Ranges

Function	Search space	Init. range	Limit
$f_1(x)$	[-100.0,100.0]	[-100.0,50.0]	100.0
$f_2(x)$	[-2.048,2.048]	[-2.048,2.048]	100.0
$f_3(x)$	[-600.0,600.0]	[-600.0, 200.0]	600.0
$f_4(x)$	[-5.12,5.12]	[-5.12,2.0]	10.0
$f_5(x)$	[-32.768,32.768]	[-32.768,16.0]	40.0

Table 2: Mean of the best value obtained.

Function	CLPSO	IWPSO	SBPSO
f_1	7.79e-16	0.001	**2.20e-39**
f_2	25.55	31.66	**22.69**
f_3	**5.45e-10**	2.021e-2	1.89e-3
f_4	**1.79e-3**	24.8	8.42
f_5	**1.04e-8**	2.33	2.26e-3

4. CONCLUSIONS AND FUTURE WORK

Our proposed approach only requires one extra parameter with respect to a traditional PSO algorithm. Additionally, it does not require any changes in the equations for computing the velocity and position of the particles and does not use the inertia parameter. Nevertheless, and in spite of its simplicity, our proposed approach outperforms the inertia weight model and obtains competitive results with respect to a more elaborate PSO algorithm.

Although our SBPSO method does not show better results in all the test functions adopted, we argue that its simplicity can make it an interesting choice for those wishing to use a simple, but still competitive PSO algorithm. Furthermore, our approach can also be combined with other methods. As part of our future work, we plan to test our SBPSO with other PSO methods that have been reported in the specialized literature.

5. REFERENCES

[1] M. Clerc and J. Kennedy. The particle swarm: explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[2] J. Kennedy and R. C. Eberhart. Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[3] J. Liang, A. Qin, P. Suganthan, and S. Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, 10(3):281–295, June 2006.

[4] W. Rudin. *Principles of Mathematical Analysis.* McGraw-Hill, New York, 1976.

[5] Y. Shi and R. Eberhart. Empirical study of particle swarm optimization. *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*, 3, 1999.

[6] Y. Shi and R. C. Eberhart. A modified particle swarm optimizer. *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998)*, 1998.