Comparing inertial weights and Constriction factor in particle swarm optimization

Conference Paper · February 2000

DOI: 10.1109/CEC.2000.870279 · Source: IEEE Xplore

2 authors:

B. Gireesha Obayyanahatti
Central University of Karnataka
128 PUBLICATIONS **109,877** CITATIONS

SEE PROFILE

Yuhui Shi
Southern University of Science and Technology
289 PUBLICATIONS **46,128** CITATIONS

SEE PROFILE

Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization

R. C. Eberhart
Purdue School of Engineering and Technology
799 West Michigan Street
Indianapolis, IN 46202 USA
Eberhart@engr.iupui.edu

Y. Shi
EDS Indianapolis Technology Center
12400 North Meridian Street
Carmel, IN 46032 USA
Yuhui.Shi@EDS.com

Abstract- The performance of particle swarm optimization using an inertia weight is compared with performance using a constriction factor. Five benchmark functions are used for the comparison. It is concluded that the best approach is to use the constriction factor while limiting the maximum velocity *Vmax* to the dynamic range of the variable *Xmax* on each dimension. This approach provides performance on the benchmark functions superior to any other published results known by the authors.

1 Introduction

Particle swarm optimization (PSO) is an evolutionary computation technique motivated by the simulation of social behavior. PSO was developed by Kennedy and Eberhart (Kennedy and Eberhart 1995; Eberhart, Simpson, and Dobbins 1996).

PSO is similar to a genetic algorithm (GA) in that the system is initialized with a population of random solutions. It is unlike a GA, however, in that each potential solution is also assigned a randomized velocity, and the potential solutions, called *particles*, are then "flown" through the problem space.

The authors have published several papers that describe research with a version of the particle swarm algorithm that incorporates what we call an *inertia weight* (Shi and Eberhart 1998).

Equations (1) and (2) describe the velocity and position update equations with the inertia weight included. Equation (1) calculates a new velocity for each particle (potential solution) based on its previous velocity, the particle's location at which the best fitness so far has been achieved, and the population global (or local neighborhood, in the neighborhood version of the algorithm) location at which the best fitness so far has been achieved. Equation (2) updates each particle's position in solution hyperspace. The two random numbers are independently generated. The use of the inertia weight *w*, which typically decreases linearly from about 0.9 to 0.4 during a run, has provided improved performance in a number of applications.

$$v_{id} = w*v_{id} + c_1 * rand( ) * (p_{id} - x_{id}) + c_2 * Rand( ) * (p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

Recent work done by Clerc (1999) indicates that use of a *constriction factor* may be necessary to insure convergence of the particle swarm algorithm. A detailed discussion of the constriction factor is beyond the scope of this paper, but a simplified method of incorporating it appears in equation (3), where K is a function of c_1 and c_2 as reflected in equation (4).

$$v_{id} = K*[v_{id} + c_1 * rand( ) * (p_{id} - x_{id}) + c_2 * Rand( ) * (p_{gd} - x_{id})] \quad (3)$$

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \text{ where } \varphi = c_1 + c_2, \quad \varphi > 4 \quad (4)$$

2 Experimental Approach

For comparison, five non-linear benchmark functions are used here. The first function is the Sphere function described by equation (5):

$$f_0(x) = \sum_{i=1}^{n} x_i^2 \quad (5)$$

where $x = [x_1, x_2, ..., x_n]$ is an n-dimensional real-valued vector. The second function is the Rosenbrock function described by equation (6):

$$f_1(x) = \sum_{i=1}^{n} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \quad (6)$$

The third function is the generalized Rastrigrin function described by equation (7):

$$f_2(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10) \quad (7)$$

0-7803-6375-2/00/$10.00 ©2000 IEEE.

The fourth function is the generalized Griewank function described by equation (8):

$$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1 \qquad (8)$$

The fifth function is Schaffer's f6 function which is described by Equation (9):

$$f_6(x) = 0.5 - \frac{\left(\sin\sqrt{x^2+y^2}\right)^2 - 0.5}{\left(1.0 + 0.001\left(x^2+y^2\right)\right)^2} \qquad (9)$$

In all cases, the population size was set to 30, and the maximum number of iterations was set to 10,000. Each of these values is somewhat arbitrary. The maximum number of iterations was set higher than ever found necessary in previous applications, and at the upper limit of one of the author's (RE's) patience.

In all cases for which the inertia weight was used, it was set to 0.9 at the beginning of the run, and made to decrease linearly to 0.4 at the maximum number of iterations. Inertia weight cases used a *Vmax* set to the maximum range *Xmax*. Each of the two *(p-x)* terms was multiplied by an acceleration constant of 2.0 (times a random number between 0 and 1).

In all cases for which Clerc's constriction method was used, φ was set to 4.1 and the constant multiplier K is thus 0.729. This resulted in the previous velocity being multiplied by 0.729, and each of the two *(p-x)* terms being multiplied by 0.729 * 2.05 = 1.49445 (times a random number between 0 and 1). In the initial comparisons, *Vmax* was set to 100,000, since it was believed that *Vmax* isn't even needed when Clerc's constriction approach is used. Note that this is functionally equivalent to using equation (1) with w = .729 and $c_1 = c_2$ = 1.49445. These new parameter values are, however, related (see equation (4)).

In all cases (intertia weight and Clerc's constriction methods) particles were allowed to fly outside of the region defined by *Xmax* (see the remarks about flying outside of the Solar System in Section 4.1).

3 Initial Results

3.1 Introduction

The surprising result of these comparisons is that it appears that setting *Vmax* = *Xmax* significantly improves results when using the constriction approach. In fact, the bottom line appears to be that the most consistent way to obtain good results (and almost always the fastest way) is to use the constriction approach with *Vmax* = *Xmax*. But we are getting ahead of ourselves. Let us first examine the initial comparisons. Each version of each approach (inertia weight and constriction factor) was run 20 times for each test function.

3.2 Spherical Function

The spherical function was always run with 30 dimensions, the most usually reported in the literature, a value for *Xmax* of 100, and was run until an error less than 0.01 was obtained. Table 1 gives the number of iterations required to reach this error value using the inertia weight method; Table 2 gives iterations needed using the constriction method.

Table 1: Spherical function iterations using inertia weight

1561	1587	1537	1530	1500
1584	1547	1504	1485	1538
1532	1582	1501	1495	1500
1615	1506	1523	1553	1576

The average number of iterations using the inertia weight is thus 1537.8; the range of values is 130 (from 1485 to 1615), about 8.5 percent of the average.

Table 2: Spherical function iterations using constriction factor and *Vmax=100000*

575	552	550	559	599
553	547	535	503	579
556	514	583	519	572
542	550	570	560	523

The average number of iterations using the constriction factor with *Vmax* = 100000 is thus 552.05; the range of values is 96 (from 503 to 599), about 17.4 percent of the average. The constriction method thus yields faster results, with a higher range/average quotient.

3.3 Rosenbrock Function

The Rosenbrock function was always run with 30 dimensions, the most usually reported in the literature, a value for *Xmax* of 30, and was run until an error less than 100 was obtained. Table 3 gives the number of iterations required to reach this error value using the inertia weight method; Table 4 contains iterations needed using the constriction method.

Table 3: Rosenbrock function iterations using inertia weight

3469	3560	3723	4211	3416
3439	3223	3526	3241	3375
4180	3269	4382	3404	3202
3356	4506	2866	3105	2894

The average number of iterations using the inertia weight is thus 3517.35; the range of values is 1640 (from 2866 to 4506), about 46.6 percent of the average.

Table 4: Rosenbrock function iterations using constriction factor and *Vmax*=100000

518	1122	655	1289	651
768	577	3081	1023	700
623	549	2670	671	4541
619	475	796	4793	2361

The average number of iterations using the constriction factor with *Vmax* = 100000 is 1424.1; the range of values is 4318 (from 475 to 4793), about 303 percent of the average. The constriction method thus again yields faster results, with a higher range/average quotient.

3.4 Rastrigrin Function

The Rastrigrin function was always run with 30 dimensions, the most usually reported in the literature, a value for *Xmax* of 5.12, and was run until an error of less than 100 was obtained. It was with this test that problems started to appear with the constriction method (with *Vmax = 100000*). Table 5 gives the number of iterations required to reach the error value of 100 using the inertia weight method. Using the constriction method, the results for which are in Table 6, convergence to the specified error value was not achieved in one of the 20 runs; that run was terminated after 10000 iterations with an error of about 125.

Table 5: Rastrigrin function iterations using inertia weight

1392	1122	1283	1523	1514
1346	1431	895	1367	1567
1370	1333	1320	1704	1379
1390	1379	743	1012	1348

The average number of iterations using the inertia weight is thus 1320.9; the range of values is 961 (from 743 to 1704), about 73 percent of the average.

Table 6: Rastrigrin function iterations using constriction factor and *Vmax*=100000

372	283	329	258	248
421	316	474	5000	299
10000*	254	285	7056	259
307	233	439	304	780

*error value not reached in 10000 iterations

The average number of iterations using the constriction factor with *Vmax*=100000 is 943 for 19 of the 20 runs. The target error was not reached within 10000 iterations for one run. For the 19 successful runs, the range of values is 6823 (from 233 to 7056), about 724 percent of the average. The constriction method yielded wide variance with this function, and it is not clear that it would be a better choice than the inertia weight method for the Rastrigrin test function.

3.5 Griewank Function

The Griewank function was always run with 30 dimensions, the most usually reported in the literature, a value for *Xmax* of 600, and was initially run until an error of less than 0.05 was obtained. Even more problems were found with the constriction method (with *Vmax*=100000) on this function than on the last one. Table 7 contains the number of iterations required to reach the error value of 0.05 with the inertia weight method.

Table 7: Griewank function iterations using inertia weight, error = 0.05

2927	2802	2756	3332	2767
3011	2785	2928	3891	2556
2759	2816	2769	2869	2772
2761	2894	2786	2888	2941

The average number of iterations using the inertia weight (to an error of .05) is thus 2900.5; the range of values is 1335 (from 2556 to 3891), about 46 percent of the average.

Using the constriction factor with *Vmax=100000*, in three runs the full 10000 iterations were run without achieving the error value of .05. It was decided to relax the error level to .1 and try again. Table 8 shows the number of iterations required to reach the error value of 0.1 using the constriction method.

Table 8: Griewank function iterations using constriction factor, error=0.1, *Vmax=100000*

10000*	483	415	431	444
414	10000*	414	399	384
440	406	423	413	10000*
398	425	418	663	459

*error value not reached in 10000 iterations

The average number of iterations using the constriction factor (*Vmax=100000, error=0.1*) is 437 for 17 of the 20 runs. The target error was not achieved within 10000 iterations for three runs. For the 17 successful runs, the range of values is 279 (from 384 to 663), about 64 percent of the average. For comparison, Table 9 contains the number of iterations required to reach the error value of 0.10 with the inertia weight method.

Table 9: Griewank function iterations using inertia weight, error=0.10

2645	2721	2682	2718	2728
2638	3035	2664	2947	2827
2822	2856	2741	2776	2704
2646	2749	2798	2754	2703

The average number of iterations using the inertia weight (to an error of 0.10) is 2757.7; the range of values is 397 (from 2638 to 3035), about 14 percent of the average. Even though the constriction method converged on the desired error value much more quickly 85 percent of the time, it did not converge 15 percent of the time, and the inertia weight method appears more reliable for the Griewank test function.

3.6 Schaffer's f6 Function

Schaffer's f6 function was always run with two dimensions, as it is defined, a value of *Xmax* of 100, and was run until an error value of .00001 was achieved. Table 10 gives the number of iterations required to reach this error value using the inertia weight method; Table 11 contains iterations needed using the constriction method.

Table 10: Schaffer's f6 function iterations using inertia weight

422	449	627	569	748
478	748	481	707	422
530	325	372	439	491
513	500	339	458	629

The average number of iterations using the inertia weight is thus 512.35; the range of values is 409 (from 339 to 748), about 80 percent of the average.

Table 11: Schaffer's f6 function iterations using constriction factor, *Vmax*=100000

206	832	843	310	741
288	608	156	239	139
860	391	899	411	127
237	433	105	498	288

The average number of iterations using the constriction factor is 430.55; the range of values is 794 (from 105 to 899), about 184 percent of the average. The constriction method thus yields faster results, with a higher range/average quotient.

4 Observations and Improvements

4.1 Observations

One of the authors (RE) watched the particles "fly" for all of the runs reported above. It was observed that the variance using the constriction method and a *Vmax* of 100000 was much greater than when using the inertia weight method. In fact, the scale of the area used to observe the particles had to be increased by 10 times on both the *x* and *y* scales (100 times in area) in order that the constriction method particles not fly off-screen. It was like watching spacecraft explore the Milky Way Galaxy in order to find a target known to be in the Solar System.

If you know that the target is in the Solar System, it makes sense to limit the distance that can be covered in one time step to the largest dimension of (distance across) the system. In our case, we call this maximum distance *Xmax*. Note that if we limit our maximum velocity to *Xmax*, we are not limiting our exploration to the Solar System; our spacecraft particles can still overshoot the system, sometimes by a wide range. But we are limiting our search to at least some reasonable vicinity of the system. It is, of course, generally assumed that the optimum we are seeking is somewhere within this dynamic range defined by *Xmax*.

4.2 Constriction Method with Vmax=Xmax

It was therefore decided to try the constriction method on all of the test functions configured as before except to set *Vmax=Xmax*. The results are presented in Tables 12-16, in the same order of test functions as above. The results were surprisingly better.

Table 12: Spherical function iterations using constriction factor and *Vmax=Xmax=100*

557	544	511	525	546
520	535	495	514	573
525	531	533	534	518
523	521	557	512	519

For the spherical function, the average number of iterations is 529.65; the range is 78 (from 495-573), about 15 percent of the average. This represents an improvement over the results with *Vmax=100000*, both in number of iterations and range.

Table 13: Rosenbrock function iterations using constriction factor and *Vmax=Xmax=30*

402	748	810	864	853
1394	760	448	529	594
615	562	544	1211	725
627	454	479	449	901

The average number of iterations for the Rosenbrock function is 668.75; the range is 992 (from 402 to 1394), about 148 percent of the average. This represents a significant improvement over the results with *Vmax=100000*, both in number of iterations required and the range.

Table 14: Rastrigrin function iterations using constriction factor and *Vmax=Xmax=5.12*

182	209	172	219	252
174	183	262	171	336
177	212	161	252	186
191	177	251	263	239

For the Rastrigrin function, the average number of iterations is just 213.45; the range is 175 (from 161 to 336), about 82 percent of the average. This is a very significant improvement over the results with *Vmax=100000*, both in number of iterations required and the range. Note that the error value was achieved in each of the 20 runs, a result not obtained with the larger *Vmax*.

Table 15: Griewank function iterations using constriction factor, error=0.1, *Vmax=Xmax=600*

283	366	360	287	316
305	282	297	329	328
306	310	321	288	296
290	316	365	294	313

For the Griewank function, the average number of iterations is 312.6; the range is 84 (from 282 to 366), about 27 percent of the average. This is another significant improvement over the results with *Vmax=100000*, both in number of iterations required and the range. Note that the error value was achieved in each of the 20 runs, a result not obtained with the larger *Vmax*.

Table 16: Schaffer f6 function iterations using constriction factor, *Vmax=Xmax=100*

897	693	691	121	257
225	2046	241	552	330
175	311	312	1203	480
300	290	94	508	922

The average for the Schaffer f6 function is thus 532.4; the range is 1952 (from 94 to 2046), about 367 percent of the average. Here we have the only case for which the restricted *Vmax* did not yield better results. However, if the one "outlier" value of 2046 is removed, the average for the remaining 19 runs is about 453, not far from the value obtained for the larger *Vmax*.

5 Discussion and Conclusion

Comparing the PSO algorithm with inertia weight represented by equations (1) and (2) with the algorithm with constriction factor represented by equations (3) and (4), we see that equations (1) and (2) are equivalent to equations (3) and (4) if the inertia weight w is set to be K, and c_1 and c_2 meet the conditions $\varphi = c_1 + c_2$, $\varphi > 4$. The PSO algorithm with the constriction factor can be considered as a special case of the algorithm with inertia weight since the three parameters are connected through equation (4).

From the experimental results, it can be concluded that the best approach to use with particle swarm optimization as a "rule of thumb" is to utilize the constriction factor approach while limiting *Vmax* to *Xmax*, or utilize the inertia weight approach while selecting w, c_1, and c_2 according to equation (4). This is easy and straightforward, particularly since the particle swarm paradigm as currently implemented requires the specification of *Xmax* (the Solar System dimension) anyway. The results here also indicate that improved performance can be obtained by carefully selecting the inertia weight w, c_1, and c_2. A method to dynamically adapt the inertia weight is under investigation.

Acknowledgments

The cooperation and comments of Jim Kennedy are appreciated and acknowledged. Portions of this paper are adapted from a chapter of *Swarm Intelligence: Collective Adaptation*, to be published in about October 2000 by Morgan Kaufmann Publishers; their permission to use this material is gratefully acknowledged.

References

Clerc, M. (1999). The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. *Proc. 1999 ICEC*, Washington, DC, pp 1951-1957.

Eberhart, R., Simpson, P., and Dobbins, R. (1996). *Computational Intelligence PC Tools*. Boston: Academic Press Professional.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. *Proc. 1995 ICEC*, Perth, Australia.

Shi, Y. and Eberhart, R. (1998) Parameter selection in particle swarm optimization. In *Evolutionary Programming VII: Proc. EP98*, New York: Springer Verlag pp. 591-600.