Parameter Selection in Particle Swarm Optimization

Yuhui Shi and Russell C. Eberhart
Department of Electrical Engineering
Indiana University Purdue University Indianapolis
723 W. Michigan St., SL160
Indianapolis, IN 46202
shi,eberhart@tech.iupui.edu

Abstract

This paper first analyzes the impact that inertia weight and maximum velocity have on the performance of the particle swarm optimizer, and then provides guidelines for selecting these two parameters. Analysis of experiments demonstrates the validity of these guidelines.

Introduction

Different from traditional search algorithms, evolutionary computation techniques work on a population of potential solutions (points) of the search space. Through cooperation and competition among the potential solutions, these techniques often can find optima more quickly when applied to complex optimization problems. The most commonly used population-based evolutionary computation techniques are motivated from the evolution of nature. Four well-known examples are genetic algorithms [6], evolutionary programming [5], evolution strategies [10] and genetic programming [9]. Different from these evolution-motivated evolutionary computation techniques, a new evolutionary computation technique, called particle swarm optimization (PSO), is motivated from the simulation of social behavior. PSO was originally designed and developed by Eberhart and Kennedy [3,4,7,8]. By adding a new inertia weight into PSO, a new version of PSO is introduced in [13]. In PSO, instead of using genetic operators, each particle (individual) adjusts its "flying" according to its own flying experience and its companions' flying experience. Each particle is treated as a point in a D-dimensional space. The *i*th particle is represented as $X_I = (x_{i1}, x_{i2}, \ldots, x_{iD})$. The best previous position (the position giving the best fitness value) of the *i*th particle is recorded and represented as $P_I = (p_{i1}, p_{i2}, \ldots, p_{iD})$. The index of the best particle among all the particles in the population is represented by the symbol *g*. The rate of the position change (velocity) for particle *i* is represented as $V_I = (v_{i1}, v_{i2}, \ldots, v_{iD})$. The particles are manipulated according to the following equation:

$$v_{id} = w * v_{id} + c_1 * \mathrm{rand}(\,) * (p_{id} - x_{id}) + c_2 * \mathrm{Rand}(\,) * (p_{gd} - x_{id}) \qquad (1a)$$
$$x_{id} = x_{id} + v_{id} \qquad (1b)$$

where c_1 and c_2 are two positive constants, rand() and Rand() are two random functions in the range [0,1], and w is the inertia weight. Equation (1a) is used to calculate the particle's new velocity according to its previous velocity and the distances of its current position from its own best experience (position) and the group's best experience. Then the particle flies toward a new position according to equation (1b).

The performance of each particle is measured according to a predefined fitness function, which is related to the problem to be solved. The inertia weight w is employed to control the impact of the previous history of velocities on the current velocity, thereby influencing the trade-off between global (wide-ranging) and local (nearby) exploration abilities of the "flying points." A larger inertia weight w facilitates global exploration (searching new areas) while a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area. Suitable selection of the inertia weight w can provide a balance between global and local exploration abilities and thus require fewer iterations on average to find the optimum. In this paper, an analysis of the impact of this inertia weight together with the maximum velocity allowed on the performance of PSO is given, followed by experiments that illustrate the analysis and provide some insights into optimal selection of the inertia weight and maximum velocity allowed.

Analysis

PSO, to some extent, resembles evolutionary programming, and is also related to cultural algorithms [11]. The addition of velocity to the current position to generate the next position is similar to the mutation operation in evolutionary programming except that "mutation" in PSO is guided by a particle's own "flying" experience and the group's "flying" experience. In another words, PSO performs "mutation" with a "conscience." By looking at the personal best elements associated with each individual as additional population members, PSO also has a form of selection even though it is quite weak [1]. In evolutionary programming, the global and local exploration abilities are brought in by mutation and controlled by the variances of the Gaussian random functions used. In order to balance between the global and local exploration abilities and obtain a quick search, the variances can also be encoded into the individuals and therefore evolved simultaneously, as done in evolution strategy [10]. In PSO the balance between the global and local exploration abilities is mainly controlled by the inertia weights. In [13], experiments have been performed to illustrate this. By setting the maximum velocity allowed to be two, it was found that PSO with an inertia weight in the range [0.9, 1.2] on average has a better performance; that is, it has a greater chance to find the global optimum within a reasonable number of iterations. Furthermore, a time decreasing inertia weight from 1.4 to 0 is found to be better that a fixed inertia weight. This is because the larger inertia weights at the beginning help to find good seeds and the later small inertia weights facilitate fine search.

By looking at equation (1) more closely, it can be seen that the maximum velocity allowed actually serves as a constraint that controls the maximum global exploration ability PSO can have. By setting a too small maximum velocity allowed, maximum global exploration ability is limited, and PSO will always favor a local search no matter what the inertia weight is. By setting a large maximum velocity allowed, then the PSO can have a large range of exploration ability to select by selecting the inertia weight. Since the maximum velocity allowed affects global exploration ability indirectly and the inertia weight affects it directly, it will generally be better to control global exploration ability through inertia weight only. A way to do that is to delete maximum velocity allowed in the algorithm implementation, and allow inertia weight itself to control exploration ability. But this should be done with care, since it's not a good idea for PSO to do global exploration all the time because this will mean that the system will always be eager to explore new areas and consequently make the system lack local exploration ability, and fail to find the solution. From the above, it's clear that choosing a large inertia weight to facilitate more global exploration is not a good strategy, and a smaller inertia weight should be selected to achieve a balance between global and local exploration so that a faster search results.

Experiments and Discussion

In order to see the influence that the inertia weight has on PSO performance under different maximum velocities allowed, the benchmark problem of Schaffer's f6 function [2] was chosen as the test problem since it is well-known and its global optimum is known. The PSO implementation was written in C and compiled using the Borland C++ Version 4.5 compiler. For purposes of comparison, all the simulations use the same parameter settings for the PSO implementation except the inertia weight w and maximum velocity allowed. The population size (number of particles) is 20. The dynamic range for each element of a particle is defined as (-100, 100), that is, the particle cannot move out of this range in each dimension and thus Xmax = 100. The maximum number of iterations allowed is 4000. If the PSO implementation cannot find a acceptable solution within 4000 iterations, it is ruled that it fails to find the global optimum in this run.

Different inertia weights w under different maximum velocities (Vmax) allowed have been chosen for simulation. For each selected w and Vmax, 30 runs are performed and the iterations required for finding the global optimum are recorded. First, the maximum velocity allowed was set to 3, 30 runs were done for a set of different inertia weights, and the results are given in Table 1. From Table 1, it is seen that only when $w = 0.9$ do all the 30 runs find the global optimum; all other weights have some runs that fail to find the global optimum within 4000 iterations. The number of failures versus inertia weights is shown in Figure 2. For comparison, the result in [13] is adopted here and shown in Figure 1. Comparing Figure 2 and Figure 1, it is easy to see that the inertia weight with no failures has changed from 1.05 to 0.9, which is consistent with our analysis in the previous section. Also notice that the average number of iterations required to find the global optimum has decreased from 1912 to 738. This is a significant improvement.

According to the analysis in the previous section, it is natural to think that with an increase of maximum velocity allowed, the inertia weight without failure and the average number of iterations required would be expected to decrease. To illustrate this, two experiments are performed with Vmax = 4 and Vmax = 5, respectively. The results are recorded in Tables 2 and 3, and the number of failures versus inertia weights appear in Figures 3 and 4, respectively. From Figures 3 and 4, we see now that for both $w = 0.9$ and $w = 0.8$ the PSO implementation finds the global optimum for all 30 runs. The inertia weight without failure is moving toward the zero, but slowly. From Tables 2 and 3, the average number of iterations ($w = 0.8$) has dropped from 738 ($w = 0.9$) to 439 (Vmax = 4) and 366 (Vmax = 5). From the previous results, it is observed that the inertia weight without failure changes more slowly than Vmax does. To further clarify this, we set Vmax = 10 and run the experiment again with varying inertia weights. Results are given in Table 4, and number of failures versus weights is shown in Figure 5. The only inertia weight without failure is $w = 0.8$. The average number of iterations is 460, which is a little larger than the previous two, but still very good compared with those for Vmax <= 3.

From the previous experiments, we know the inertia weight without failure decreased more slowly than the maximum velocity allowed increased. So what will happen if we eliminate the constraint of Vmax? Will this degrade the performance of PSO or even destroy it? To explore this, we set Vmax = Xmax. This is reasonable since [-Xmax, Xmax] is the dynamic range of the elements of each particle. The result is given in Table 5, and number of failures versus weights is illustrated in Figure 6. It's a surprise to find that $w = 0.8$ is the inertia weight without failure again. This time, the average number of iterations increased to 974 (still better than the result in [13]), but this seems to be a good sign since we may not need to consider how to select Vmax. In many practical problems, it's difficult to select the best Vmax without trial-and-error.

To compensate the for the increase in the average number of iterations brought on by deleting Vmax, a time decreasing inertia weight is employed instead of a fixed weight. It's expected that improvement can be obtained by doing so because we believe that the increase in average number of iterations is due to the significant increase in global search caused by deleting Vmax. A time-dependent inertia weight is one way to compensate for this. Based on the previous results, we define the inertia weight w to linearly decrease from 0.9 to 0.4 during the first 1500 iterations and stay constant at 0.4 for the remaining 2500 iterations. Thirty runs were conducted and the results are given in Table 6. From Table 6, we can see that all 30 runs found the optimum and the average number of iterations is the lowest among all of the experimental settings. Another significant observation is that the variance of iterations required to find the global optimum is also the smallest.

Conclusions

In this paper, we have analyzed the impact of the inertia weight and maximum velocity allowed on the performance of PSO. A number of experiments have been done with different inertia weights and different values of maximum velocity allowed. It is concluded that when Vmax is small (<= 2 for the f6 function), an inertia weight of approximately 1 is a good choice, while when Vmax is not small (>= 3), an inertia weight $w = 0.8$ is a good choice. When we lack knowledge regarding the selection of Vmax, it is also a good choice to set Vmax equal to Xmax and an inertia weight $w = 0.8$ is a good starting point. Furthermore if a time varying inertia weight is employed, even better performance can be expected.

Even though good experimental results have been obtained in this paper, only a small benchmark problem has been tested. The selection of the inertia parameter and maximum velocity allowed may be problem-dependent. To fully justify the benefits of selecting parameters as described in this paper, more problems need to be tested. By doing so, a clearer understanding of PSO performance will be obtained. Indeed, a fuzzy system [12] may be a good candidate for online tuning of the inertia weight.

References

1. Angeline, P. J. (1998), Using selection to improve particle swarm optimization, IEEE Intl. Conf. on Evolutionary Computation, Anchorage, AK, in press.
2. Davis, L., Ed. (1991), Handbook of Genetic Algorithms, New York, NY: Van Nostrand Reinhold.
3. Eberhart, R. C., Dobbins, R. W., and Simpson, P. K. (1996), Computational Intelligence PC Tools, Boston: Academic Press.
4. Eberhart, R. C., and Kennedy, J. (1995). A new optimizer using particle swarm theory, Proc. Sixth Intl. Symp. on Micro Machine and Human Science (Nagoya, Japan), IEEE Service Center, Piscataway, NJ, 39-43.
5. Fogel, L. J. (1994), Evolutionary programming in perspective: the top-down view, in Computational Intelligence: Imitating Life, J.M. Zurada, R. J. Marks II, and C. J. Robinson, Eds., IEEE Press, Piscataway, NJ.
6. Goldberg, D. E. (1989), Genetic Algorithms in Search, Optimization, and Machine Learning, Reading, MA: Addison-Wesley.
7. Kennedy, J., and Eberhart, R. C. (1995). Particle swarm optimization, Proc. IEEE Intl. Conf. on Neural Networks, IEEE Service Center, Piscataway, NJ, IV: 1942-1948.
8. Kennedy, J. (1997), The particle swarm: social adaptation of knowledge, Proc. IEEE Intl. Conf. on Evolutionary Computation, IEEE Service Center, Piscataway, NJ, 303-308.
9. Koza, J. R. (1992), Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA.
10. Rechenberg, I. (1994), Evolution strategy, In Computational Intelligence: Imitating Life, J. M. Zurada, R. J. Marks II, and C. Robinson, Eds., IEEE Press, Piscataway, NJ.
11. Reynolds, R. G. (1994), An introduction to cultural algorithms, in Proc. 3rd Ann. Conf. On Evolutionary Programming, A. Sebald and D. Fogel, Eds., River Edge, NJ:World Scientific Publishing, 131-139.
12. Shi, Y. H., Eberhart, R. C., and Chen, Y. B. (1997), Design of evolutionary fuzzy expert system, Proc. 1997 Artificial Neural Networks in Engineering Conf.
13. Shi, Y. H., Eberhart, R. C., (1998), A modified particle swarm optimizer, IEEE Intl. Conf. on Evolutionary Computation, Anchorage, AK, in press.

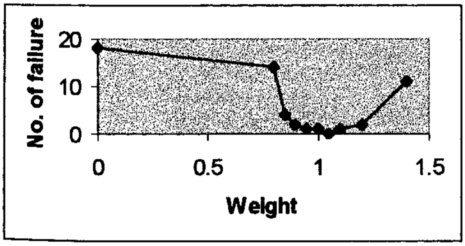

Figure 1 Number of failures vs. inertia Weights. Vmax=2

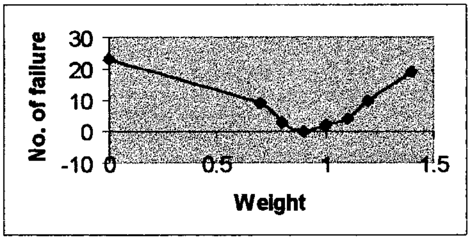

Figure 2 Number of failures vs. inertia weights. Vmax=3

Table 1 Numbers of iterations for finding the global optimum using different inertia weights. The blank cells mean these runs didn't find the global optimum within the maximum number of generations (4000). Vmax=3

	Weights							
No.	1.4	1.2	1.1	1	0.9	0.8	0.7	0
1	3957	2772	1202	564	664	91	2430	75
2		3238	444	623	841	144	100	
3		1968	857	2784	641	96		
4		594	600	3580	707	463	139	68
5			2846	1587	680			
6	243	1815		1223	910	248	1330	
7		668	3321	2940	859	820	132	
8	2380		2747	2392	625	982		
9		1799	2145	1021	630	144	217	104
10		3582	1375	2415	657	162		
11					779	1081	65	
12		2450	2515	1278	1272	123	113	99
13		3413	2232	1597	1649	1454		
14	3843		762	1647	769	200		
15		1282	1912	2792	617	178		
16		672	472	3493	373	198	429	
17		2044	937	59	391	126	394	67
18	1653		1257	2217	359	276		
19	1800		3943	2709	286	1577	87	
20			961	1183	381	136	506	
21				395	710	134	103	
22	2723	1586	2897	1280	294			
23		242	2941	489	1620	100	187	
24		1876	2356	2763	2095	226	111	
25		415	2497	1832	241	73	90	
26	3315		2747	281	370	982	133	848
27	3266	750	911		496	182	1549	
28	345			1081	805	1454	93	3383
29		3292	198	1934	677		84	
30	2728	2336	627	124	756	187	145	
Aver.	2387	1840	1758	1653	738	438	402	663

Table 2 Numbers of iterations for finding the global optimum using different inertia weights. Vmax=4

	Weights							
No.	1.2	1.1	1	0.9	0.8	0.7	0.6	0
1	3387	2676	1880	437	209	175	339	
2		1068	928	1657	322	130		
3	3433			1540	734	489		
4				1188	135	126		
5			732	811	298	3295	61	
6		1187		808	115	218	147	69
7		2341	3046	575	811	93	1193	
8	2676		288	1063	1392		104	

9	3406		3499	645	387	1605		64
10		319		515	92	107	1722	1178
11	3349	3971	3597	1513	371			
12		2029	1871	1195	143	135		328
13		1059		861	489	344	804	339
14			1029	730	526	136		315
15	2689		3225	1193	566	770	292	
16			3056	492	285	1035		230
17		3408	788	995	452	353	750	1676
18			1895	242	88	214	61	757
19	1058	1551	2850	1178	499	136	92	
20	1589		2417	718	328	104		
21	727	3547	2705	2008	364	107	390	52
22				771	266	925		324
23	2993		1558	247	771		317	
24	2544		3170	947	207	252		
25	171	2132	2640	240	132	100	66	854
26		2120	395	2274	158	85	639	834
27	777			1303	551	176	138	
28				564	530	145	44	1162
29		2379	655	422	1870	887		
30	2214		1042	1241	82	575		1602
Aver.	2215	2128	1967	946	439	471	421	652

Table 3 Numbers of iterations for finding the global optimum using different inertia weights. Vmax=5

No.	weights					
	1.0	0.9	0.8	0.7	0.6	0
1	2553	353	136	409	285	180
2		1939	554	168		225
3	3867	700	450	126	398	93
4	2143	1806	316	111		
5		476	191	1967		354
6	864	1348	544			1327
7	486	620	467	438	116	1940
8	1724	676	270			550
9		467	315	235	162	1601
10		1464	1448	80	2148	2841
11	639	471	290	53	375	2545
12		502	302	188		326
13		495	548	1459		138
14		2578	244	262		3711
15		864	268		1752	300
16		2148	231	655	69	1612
17	925	2073	191	129	84	
18	2921	1262	166	79	324	2602
19	3065	841	189	2255		1168
20	3717	458	173	775	92	174
21	3656	1004	145	122	1665	564
22		757	302		296	1383

23		1695	824	137	349	485
24		941	297	62	342	2416
25		1450	193			1015
26	3608	587	380	96		1450
27		1311	729	155	75	340
28	3384	836	361	2262	907	
29		1858	223	121	558	122
30	3285	711	240	223	172	
aver.	2456	1090	366	503	535	1133

Table 4 Numbers of iterations for finding the global optimum using different inertia weights. Vmax=10

	weights					
No.	1.0	0.9	0.8	0.7	0.6	0
1		2464	1438	104	61	1634
2			456	129	175	1547
3		133	151	75		
4		3855	471	1990	538	384
5			685			305
6		3862	444	310	500	
7			201	190		
8		2962	772	194	115	402
9			183	72	2368	2517
10		2405	706	3034		459
11		2363	193	96	57	1107
12		74	175	285	474	1659
13		2474	340	409	265	
14		1477	249	1257		
15			317	2205	510	
16		2851	756	2743	79	
17		1356	541	386	409	2178
18		1782	682	253	61	
19	1147		608	1151	3015	186
20		1881	125	330	166	617
21			277	1864	76	239
22	584	1307	348	453	74	
23	244	958	475	450	237	
24			1283	1138	475	
25		1464	269	2508		
26		1928	195	359	121	1210
27		3179	633	172	939	278
28		1368	377	497	96	147
29			308	104	470	1093
30		2431	143	137		152
aver.	658	2027	460	789	490	895

Table 5 Numbers of iterations for finding the global optimum using different inertia weights. Vmax=Xmax

	weights						
No.	0.9	0.8	0.7	0.6	0.5	0.4	0
1		1336	2442				729
2		1022	263	309	16		
3		426		78	1616		
4		2078	272	378	307		144
5		1163	345	2135	106	384	
6		433	3749	755	90		224
7		896	165	3774	50		166
8		710			617	335	947
9		1326	2982	534	163		1075
10		1727	173	1393	1029	236	39
11		798	455		456		2278
12		352	1807	1434	139	49	
13		719	1322	790	256	594	
14		1113	2000	1295			358
15		843	220	248	513	898	512
16		365	264	90		608	261
17		1010	263	453	67		1055
18		924	91	1221	299		
19		1778	1075	101			
20		339	1174		153		447
21		1455	148	144	650	64	2945
22		1030	563	136	1454		3468
23		471	735	124	1160		326
24		1869	112	610		346	238
25		1779	230	228	96		3212
26		360	185	83		94	337
27		498	240	2535		69	
28		1091		3915	143	99	
29		816	2071	166	80		154
30		506	398	253			678
aver.		974	879	927	430	290	933

Table 6 Numbers of iterations for finding the global optimum using time varying inertia weights. Vmax=Xmax

Index	Iterations	Index	Iterations	Index	Iterations
1	423	11	429	21	407
2	231	12	398	22	309
3	317	13	427	23	305
4	373	14	362	24	421
5	321	15	338	25	394
6	226	16	510	26	243
7	250	17	373	27	337
8	241	18	302	28	208
9	293	19	402	29	378
10	284	20	461	30	359
Average					344

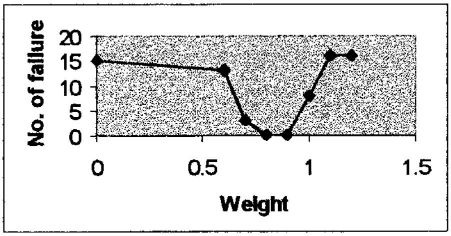

Figure 3 Number of failures vs. inertia weights. Vmax=4

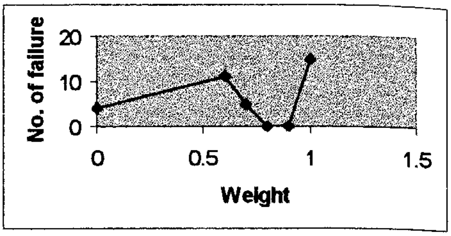

Figure 4 Number of failures vs. inertia weights. Vmax=5

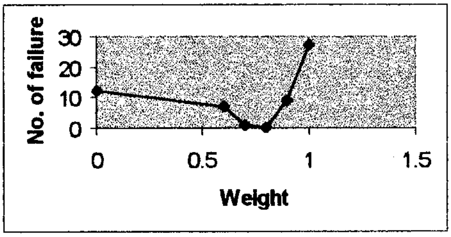

Figure 5 Number of failures vs. inertia weights. Vmax=10

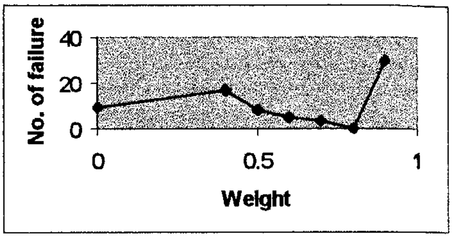

Figure 6 Number of failures vs. inertia weights. Vmax=Xmax