Dynamic and Adjustable Particle Swarm Optimization

Chen-Yi Liao, Wei-Ping Lee, Xianghan Chen, Cheng-Wen Chiang
Department of Management Information System
Chung Yuan University
Taoyuan
Taiwan, R.O.C.

Abstract: - Particle Swarm Optimization (PSO) is a stochastic, population-based evolutionary search technique. It has difficulties in controlling the balance between exploration and exploitation. In order to improve the performance of PSO and maintain the diversities of particles, we propose a novel algorithm called Dynamic and Adjustable Particle Swarm Optimization (DAPSO). The distance from each particle to the global best position is calculated in order to adjust the velocity suitably of each particle. Four benchmark functions such as Sphere, Rosenbrock, Rastrigrin, Griewank are used for the comparison of DAPSO with the Standard PSO. The experiments prove that DAPSO has better performance than the Standard PSO.

Key-Words: - Particle Swarm Optimization (PSO), Dynamic and Adjustable Particle Swarm Optimization (DAPSO), distance, velocity.

1 Introduction

Population-based stochastic search algorithms have been very popular in recent years in the research arena of computational intelligence. Some well-known search algorithms such as Genetic Algorithm[8], Evolutionary Strategies[7], Evolutionary Programming[5], have been successfully implemented to solve simple problems to complex real world problems.

Particle Swarm Optimization (PSO) is an evolutionary computation technique, first proposed by Kennedy and Eberhart, which is inspired by flocks of birds and shoals of fish in 1995[10]. It is implemented by the common evolutionary computation techniques, and it is initialized with a population of random solutions and searches for the optimum by updating generations. And the reproduction is based on the old generations. PSO is successfully implemented in various optimization problems. It is popular due to its simplicity in its implementation, as a few parameters are needed to be tuned.

However, even through PSO is a good and fast algorithm, it has limitations when solving complex problems. The original PSO has difficulties in controlling the balance between exploration and exploitation. In order to improve the performance of PSO and maintain the diversities of particles, we propose a novel algorithm called Dynamic Adjustable Particle Swarm Optimization (DAPSO). The distance from each particle to the global best position is calculated in order to adjust the velocity suitably of each particle. Section 2 presents an overview of the PSO. In Section 3, a brief introduction of DAPSO is given. In Section 4, some experiment results are presented. Conclusion is given in Section 5.

2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an evolutionary computation technique, which is inspired by flocks of birds and shoals of fish (Kennedy and Eberhart, 1995). In PSO, a number of simple entities (the particles) are placed in the space of some problem and each evaluates its fitness as its current location. Each particle determines its movement through the space by considering the particle which had the best fitness and the history of its own, then it moves with a velocity. Finally, the swarm is likely to move close to the best location. The velocity and position of each particle is adjusted by the following formulas:

$$V_{id} = w \times V_{id} + c_1 \times rand() \times (P_{id} - X_{id}) + c_2 \times Rand() \times (P_{gd} - X_{id}) \quad (1)$$

$$X_{id} = X_{id} + V_{id} \quad (2)$$

where c1 and c2 are termed the cognitive and social learning rates. These two parameters control the relative importance of the memory of the particle itself to the memory of the neighborhood. The variable rand() and Rand() are two random functions

that is uniformly distributed in the range [0,1]. Xi = (Xi1, Xi2, … , XiD) represents the ith particle. Pi = (Pi1, Pi2, …, PiD) represents the best previous position of the ith particle. The symbol g represents the index of the best particle among all the particles. Vi = (Vi1, Vi2, … , ViD) represents the velocity of the ith particle. Variable is the inertia weight. The general process of PSO is as follows.

```
Do
    Calculate fitness of particle
    Update pbest if the current fitness
    is better than pbest
    Determine nbest for each particle:
    choose the particle with the best
    fitness value of all the neighbors as
    the nbest
    For each particle
    Calculate particle velocity
    according to (1)
    Update particle position according
    to (2)
While maximum iterations or minimum
criteria is not attained
```

Since the introduction of the PSO algorithm, several improvements have been suggested. In 1998, inertia weight was first proposed by Shi and Eberhart[3]. The function of inertia weight is to balance global exploration and local exploitation. In the following year, Clerc proposed the constriction factors to ensure the convergence of PSO[2]. Eberhart and Shi compared inertia weight with constriction factors and found that the constriction factors was better convergence than inertia weight[4].

3 Dynamic and Adjustable PSO

In this section, we propose two improved algorithms called Dynamic and Adjustable Particle Swarm Optimization 1 (DAPSO1) and DAPSO2. In DAPSOs, in order to adjust the velocity of each particle, all particles are calculated the distance from itself to the global best position by the following function.

$$\Delta x_{di} = \left|(x_{di} - x_{gbest})\right| \tag{3}$$

$$FD_d = Max(\Delta x_{di}) \tag{4}$$

where x_{di} is the position of the ith particle, x_{gbest} is the position of *gbest*. FD_d is the furthest distance from the particle to *gbest*. In DAPSO1, the velocity and position of each particle is adjusted by the following formulas:

$$V_{id} = w \times V_{id} + c_1 \times rand() \times (P_{id} - X_{id}) + c_2 \times Rand() \times (P_{gd} - X_{id}) \tag{5}$$

$$ac = rand() * 0.5 \tag{6}$$

$$V_{new} = \begin{cases} V_{id} * (1 + ac_d * \dfrac{Gene - Iter}{Gene}) & \dfrac{\Delta x_{di}}{FD_d} > 0.5 + ac_d \\ V_{id} * (1 - ac_d * \dfrac{Gene - Iter}{Gene}) & \dfrac{\Delta x_{di}}{FD_d} < 0.5 - ac_d \\ V_{id} & 0.5 - ac_d <= \dfrac{\Delta x_{di}}{FD_d} <= 0.5 + ac_d \end{cases} \tag{7}$$

$$X_{id} = X_{id} + V_{new} \tag{8}$$

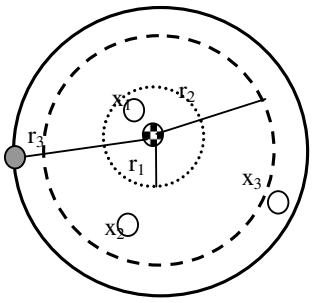
where X_{id} is updated by the velocity which is adjusted by the distance from particle to the global best and is the adjustment coefficient.

DAPSO1 and DAPSO2 differ from the adjusting method. In DAPSO2, the velocity and position of each particle is adjusted by the following formulas:

$$V_{id} = w \times V_{id} + c_1 \times rand() \times (P_{id} - X_{id}) + c_2 \times Rand() \times (P_{gd} - X_{id}) \tag{9}$$

$$V_{id} = \begin{cases} V_{id} * (1 + \dfrac{rand()}{4} * \dfrac{Gene - Iter}{Gene}) & \dfrac{\Delta x_{di}}{FD_d} > 0.5 + ac \\ V_{id} * (1 - \dfrac{rand()}{4} * \dfrac{Gene - Iter}{Gene}) & \dfrac{\Delta x_{di}}{FD_d} < 0.5 - ac \\ V_{id} & 0.5 - ac <= \dfrac{\Delta x_{di}}{FD_d} <= 0.5 + ac \end{cases} \tag{10}$$

$$X_{id} = X_{id} + V_{id} \tag{11}$$



gbest *the farthest particle from gbest*
Other particles

Fig.1: Three radius of $(0.5-ac)*FD_d$, $(0.5+ac)*FD_d$, and FD_d.

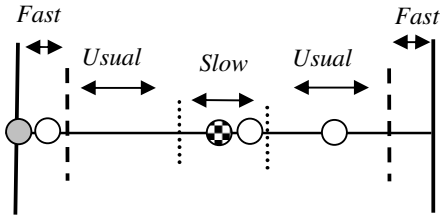


Fig.2: Adjust the velocity according to the distance from the particle to gbest.

In Fig.1, x_1, x_2 and x_3 all have difference distance from itself to global best position. x_1 drops within the radius of $(0.5-ac)*FD_d$. The distance from x_2 to global best position is between $(0.5-ac)*FD_d$ and $(0.5+ac)*FD_d$. x_3 drops beyond the radius of $(0.5+ac)*FD_d$ in DAPSO1. In DAPSOs, we define the "long distance" as the distance from the particle to the global best beyond $(0.5+ac)*FD_d$ and the "short distance" as the distance from the particle to the global best is smaller than $(0.5-ac)*FD_d$. In DAPSO1, the particles far away from the global best should be given larger value of velocity so it may explore an unknown region, whereas those close to the global best should be given smaller value of velocity so that it may exploit the neighbourhood of the global best.

In DAPSO2, if there were many particles far away from the global best position, then the velocities should be given a larger value. If there were many particles near from the global best position, then the velocities should be given a smaller value. DAPSO1 only adjusts the velocity of the certain particle, but in DAPSO2, the velocities of all particles are adjusted together.

The general flow of DAPSOs and the flowchart of DAPSO are shown as follows.

Step 1. Initialization of a population of particles with random positions and velocities

Step 2. Evaluation of particles.

Step 3. Calculate the distance from each particle to the global best position and save the farthest distance in the memory.

Step 4. Adjust particle's velocity according to its distance from itself to the global best position.

Step 5. Update particle's position by the adjusted velocity.

Step 6. Repeat Step.2~Step.5 until termination criteria are met.

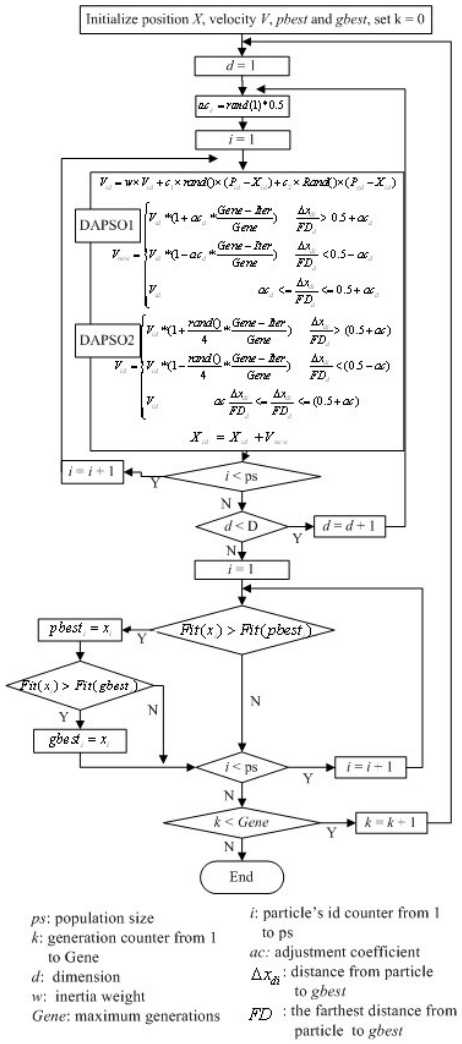


Fig.3: Flowchart of DAPSO.

4 Experiments

In this section, four benchmark functions, listed in Table 1, are used for the comparison of DAPSO with Standard PSO. These functions are all minimization problems with minimum value zeros.

The initial range of the population and Vmax are listed in Table 2. All functions are tested on 10, 20 and 30 dimensions. The maximum number of generations is set as 1000, 1500 and 2000 corresponding to the dimensions 10, 20 and 30. The population sizes are 20, 40 and 80. The general parameters of PSO are set as c1=c2=2 for all the PSO runs. All experiments were run 30 times. The mean value and standard deviation of the results are presented.

Table 1: Benchmark functions.

Test Functions	Formulations
Sphere function *f1*	$f_1(x)=\sum_{i=1}^{n} x_i^2$
Rosenbrock function *f2*	$f_2(x)=\sum_{i=1}^{n}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$
Rastrigrin function *f3*	$f_3(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$
Griewank function *f4*	$f_4(x)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$

Table 2: Initialization ranges and *Vmax*.

Test Functions	*Vmax*	Initialization Range
Sphere function *f1*	100	(50, 100)
Rosenbrock function *f2*	100	(15, 30)
Rastrigrin function *f3*	10	(2.56, 5.12)
Griewank function *f4*	600	(300, 600)

Table 3: The mean fitness value and standard derivation for Sphere function.

Popu. Size	Dim.	Gene.	Standard PSO		DAPSO1		DAPSO2	
			Mean Best	St. Dev.	Mean Best	St. Dev.	Mean Best	St. Dev.
20	10	1000	5.48E-20	1.67E-19	5.08E-12	1.44E-11	1E-30	4.16E-30
	20	1500	8.3E-12	1.66E-11	0.006151	0.01351	1.45E-17	5.84E-17
	30	2000	2.92E-08	6.53E-08	0.247817	0.143835	7.11E-11	3.09E-10
40	10	1000	5.5E-24	2.1E-23	4.58E-22	1.26E-21	1.08E-38	5.45E-38
	20	1500	6.11E-15	1.97E-14	1.97E-07	4.22E-07	1.66E-27	8.74E-27
	30	2000	8.18E-11	1.98E-10	1.21E-05	3.98E-05	2.84E-42	1.27E-14
80	10	1000	1.73E-18	4.4E-18	1.15E-26	4.802E-26	2.96E-45	8.26E-45
	20	1500	3.38E-18	6.53E-18	1.29E-15	4.32E-15	4.36E-32	5.85E-32
	30	2000	2.1E-13	2.68E-13	1.41E-10	3.7E-10	5.92E-24	1.69E-23

Table 4: The mean fitness value and standard derivation for Rosenbrock function.

Popu. Size	Dim.	Gene.	Standard PSO		DAPSO1		DAPSO2	
			Mean Best	St. Dev.	Mean Best	St. Dev.	Mean Best	St. Dev.
20	10	1000	133.3891	316.0486	53.19439	165.1511	76.72229	181.8278
	20	1500	160.7462	309.2228	1202.878	3151.682	171.721	322.933
	30	2000	223.125	360.6773	2995.28	6011.84	274.0211	299.0556
40	10	1000	70.4366	160.4263	41.91747	69.1192	74.61149	94.89749
	20	1500	154.2409	223.3388	260.8026	300.2551	53.32104	90.1557
	30	2000	189.5397	277.9211	303.1766	427.297	186.9029	350.1764
80	10	1000	44.3157	62.1308	19.94126	46.71211	22.11648	55.87017
	20	1500	172.6501	535.1584	134.1581	314.6489	76.11878	111.1735
	30	2000	195.9554	270.9865	155.0454	254.3396	143.2973	174.3063

Table 5: The mean fitness value and standard derivation for Rastrigrin function.

Popu. Size	Dim.	Gene.	Standard PSO		DAPSO1		DAPSO2	
			Mean Best	St. Dev.	Mean Best	St. Dev.	Mean Best	St. Dev.
20	10	1000	4.81036	2.45335	5.81817	3.331006	5.334921	3.113383
	20	1500	21.71077	7.76718	22.92239	6.387469	21.45805	6.041894
	30	2000	53.1911	10.1221	61.21653	15.73108	46.10063	13.18137
40	10	1000	3.32043	1.61849	3.516072	1.805159	3.118471	1.626242
	20	1500	16.8017	4.89573	17.39875	7.887492	15.32236	4.758035
	30	2000	37.3208	9.6876	41.13881	12.26015	34.92302	11.37834
80	10	1000	2.32178	1.1187	2.291675	1.283505	2.093427	1.338046
	20	1500	12.53698	3.13126	11.92332	3.549624	10.7124	4.250956
	30	2000	29.78461	8.18153	30.90371	9.178175	25.9353	7.069035

Table 6: The mean fitness value and standard derivation for Griewank function.

Popu. Size	Dim.	Gene.	Standard PSO		DAPSO1		DAPSO2	
			Mean Best	St. Dev.	Mean Best	St. Dev.	Mean Best	St. Dev.
20	10	1000	0.09015	0.04242	0.10784	0.052975	0.085375	0.038538
	20	1500	0.02621	0.02624	0.024692	0.022493	0.029952	0.027586
	30	2000	0.02487	0.02220	0.026929	0.018606	0.031479	0.030537
40	10	1000	0.07724	0.03871	0.079622	0.038493	0.068643	0.025666
	20	1500	0.02824	0.02657	0.032117	0.031035	0.025966	0.031017
	30	2000	0.01467	0.01745	0.01434	0.018109	0.012845	0.018303
80	10	1000	0.072	0.03254	0.083686	0.035153	0.064111	0.029819
	20	1500	0.028402	0.027861	0.029899	0.023863	0.026533	0.024003
	30	2000	0.011466	0.021425	0.013935	0.020599	0.009996	0.012303

DAPSO1 can improve the performance for the Rosenbrock Function if the dimension is 10, the standard deviation illustrates that DAPSO1 is stable if the dimension is small in most cases.

But for the Rastrigrin function and Griewank function, the performance of DAPSO1 is worse than PSO's.

Table 3 illustrates that DAPSO2 can improve the performance whether dimension is 10, 20 or 30 for the Sphere Function.

For the Rosenbrock function, DAPSO2 has the better performance than PSO. And the standard deviation shows that DAPSO2 is a stable algorithm.

For the Rastrigrin function, DAPSO2 has better performance than PSO.

DAPSO2 has better performance than PSO if the problem is simple for the Griewank function. Table 6 illustrates that DAPSO2 performs better if the problem of great complexity, but it needs a larger population (population size is 40 or 80) to evolve.

4 Conclusion

This paper proposed DAPSO which consideration to the distance from particle to the global best. The distance from each particle to the global best position is calculated in order to adjust the velocity of particle suitably.

In section 4, four benchmark functions such as Sphere, Rosenbrock, Rastrigrin, Griewank are used for the comparison of DAPSO with Standard PSO. The experimental results prove that the DAPSO can achieve good results. And the standard deviation illustrates that DAPSO is stable even though the problem has the great complexity.

References:

[1] Angeline, P. J., Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Differences, *1998 Annual Conference on Evolutionary Programming*, 1998.

[2]Clerc, M., The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization, *Proceedings of the IEEE Congress on Evolutionary Computation,* 1999.

[3]Eberhart, R. C. and Shi, Y., A Modified Particle Swarm Optimization, *Proceedings of IEEE International Conference on Evolutionary Computation,* 1998, pp.69-73.

[4]Eberhart, R. C. and Shi, Y., Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization, *Proceedings of the 2000 Congress on Evolutionary Computation*, Vol. 1, 2000, pp.84-88.

[5]Fogel, D. and Scbald, A.V., Use of Evolutionary Programming in the Design of Neural Networks for Artifact Detection, *Proceedings of the Twelfth Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 1990, pp. 1408-1409.

[6]Grimaldi, E., Grimaccia, F., Mussetta, M., Zich., R. and Pirinoli, Genetical Swarm Optimization: A New Hybrid Evolutionary Algorithm for Electromagnetic Applications, *Proceedings of the 18th International Conference on Applied Electromagnetic and Communications*, 2005, pp. 1-4.

[7]Greenwood, G.W., Lang, C. and Hurley, S., Scheduling Tasks in Real Time Systems using Evolutionary Strategies. *Proceedings of the Third Workshop on Parallel and Distributed Real-Time Systems*, 1995, pp. 195-196.

[8]Holland, J., Genetic Algorithms, *Scientific American*, 1992, pp. 44-50.

[9]Juang, C., A Hybrid of GA and PSO for Recurrent Network Design, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 34, No. 2, 2004, pp.997-1006.

[10] Kennedy, J. and Eberhart, R.C., Particle Swarm Optimization, *Proceedings of the Fourth IEEE International Conference on Neural Networks*, 1995, pp. 1942-1948.

[11] Poli, R., Chio, C. and Langdon, W., Exploring Extended Particle Swarm: A Genetic Programming Approach, *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, Washington, 2005, pp. 169-176.