Varying the Topology and the Probability of Re-initialization in Dissipative Particle Swarm Optimisation

Mudassar Iqbal, Alex A.Freitas, Colin G. Johnson

Computing Laboratory
University of Kent, Canterbury, UK
{mi26,aaf,cgj}@kent.ac.uk

Abstract. This paper introduces two new versions of dissipative particle swarm optimization. Both of these use a new time-dependent strategy for randomly re-initializing the positions of the particles. In addition, one variation also uses a novel dynamic neighbourhood topology based on small world networks. We present results from applying these algorithms to two well-known function optimization problems. Both algorithms perform considerably better than both standard PSO and the original dissipative PSO algorithms. In particular one version performs significantly better on high-dimensional problems that are inaccessible to traditional methods.

1 Introduction

Particle swarm optimisation, PSO, is a heuristic search/optimisation technique first proposed by Kennedy and Eberhart [1,2,3]. The underlying motivation for the algorithm is drawn from the collective behaviour of social animals, phenomena such as bird flocking, fish schooling *etc.* as well elements of social psychology. After this first proposal forwarded by Kennedy et al., several researchers have analysed the performance of PSO with different parameter settings, and PSO so far has been used across number of applications [4].

In this paper we propose two novel variants on the kind of PSO algorithm known as dissipative PSO [7]. The basic idea of dissipative PSO is to introduce chaotic perturbations into the system, by randomly initialising the particle positions with a small probability. In this algorithm the probability of re-initialisation remains constant. By contrast, our new algorithm introduces a time-dependence to this probability. In addition we discuss variants of dissipative PSO which use a local neighbourhood topology drawing inspiration from small world networks [11].

The remainder of this paper is organised as follows. Section 2 presents an overview of standard particle swarm optimisation, and reviews the core ideas of dissipative PSO. Section 3 describes our two new dissipative PSO algorithms. In section 4 we present computational results which apply these algorithms to function optimisation. Finally section 5 concludes the paper and mentions directions for future work.

2 Standard and Dissipative Particle Swarm Optimisation

Like other population-based search algorithms, Particle Swarm Optimisation (PSO) is initialised with a population of random solutions (particles). Each particle *flies* in D-dimensional problem space with a velocity, which is adjusted at each time step. The particle flies towards a position, which depends on its own past best position and the position of the best of its neighbours. The quality of a particle position depends on a problem-specific objective function (fitness).

The position of the *i*th particle is represented by a vector $X_i = (x_{i1}, \ldots, x_{id}, \ldots, x_{iD})$, where $x_{id} \in [l_d, u_d]$, $d=1....D$. l_d and u_d are the lower and upper bounds for the *dth* dimension, respectively, and D represents the number of dimensions of the search space.

The best position (i.e. that with the best fitness, the so-called *pbest*) of particle *i* is recorded as $P_i = (p_{i1}, \ldots, p_{id}, \ldots, p_{iD})$. Similarly the location of the best particle among the population is recorded by the index *g* and the location P_g is called *gbest* (global best) in the case of a global neighbourhood topology, where each particle is connected to all of the other particles. It is also possible to use a local neighbourhood topology, in which case the location of the best local neighbour is called P_l (local best). The velocity of the *ith* particle $V_i = (v_{i1}, \ldots, v_{id}, \ldots, v_{iD})$, is limited to a maximum velocity $V_{max} = (v_{\max 1}, \ldots, v_{\max d}, \ldots, v_{\max D})$.

At each time step, the particles' positions are updated depending on their *pbest* and *gbest* (or *lbest*) according to following equations:

$$\left.\begin{aligned} v_{id}^{t+1} &= w * v_{id}^{t} + c_1 r_1 (p_{id} - x_{id}^{t}) + c_2 r_2 (p_{gd} - x_{id}^{t}) \\ x_{id}^{t+1} &= x_{id}^{t} + v_{id}^{t+1} \end{aligned}\right\} \quad (1)$$

Where t is the iteration index, and w ($0 \le w < 1$) is the inertia weight, determining how much of the previous velocity of the particle is preserved. This plays the role of balancing the global and local search ability of PSO [12]. c_1, c_2 are two positive acceleration constants, r_1, r_2 are two uniform random numbers sampled from U(0,1). For the velocity update equation, the second part represents the private thinking by itself; the third part is the social part, which represent the cooperation among the individuals. In the case described by equation 1, a global neighborhood was used; later we shall explore an alternative, which uses a local neighborhood.

A PSO algorithm consists of the following steps:

1) Initialize a population of *m* particles, assigning random location between ($-X_{\max}$, $X_{\max}$) and random velocity ($-V_{\max}$, $V_{\max}$) for each dimension.
2) Evaluate the desired fitness function for each particle and update *pbest* and *gbest* if needed.
3) Change the velocity and position of each particle according to equation 1
4) Loop to step 2 until a stopping criterion is met (i.e. a good fitness value is obtained, or a predefined number of iterations is performed).

2.1 Random Re-initialisations in Dissipative PSO

A peculiar property of the standard PSO algorithm is that, although it finds reasonable quality solutions much faster than many other population-based optimisation algorithms, it does not continue to improve on the quality of solutions after a certain number of generations have passed [6]. That is, it is lacking enough capability to achieve "sustainable development" [7]. The swarm becomes stagnated after a certain number of iterations.

Xie et al. (2002) devised one solution to this by introducing additional randomness into the system. This is done by randomly re-initialising particle positions with very small probability (~0.001) at every iteration, which improves the performance quite significantly with respect to standard PSO. In this way particles are not only referring to their historical positions and those of their fellows, but also they are affected by small changes in their environment. This chaotic perturbation, or negative entropy (i.e. considering swarm as an "open dissipative system" [7]) brings aspects of the outside world into the system (swarm) which prevents the system from settling at an equilibrium. Then self-organization of this complex interacting system leads to sustainable development from the fluctuations. The additional entropy put into the system is added by the following two equations, which are executed in the simple PSO after the velocity and position update equations have been evaluated (eqn.1).

The chaotic perturbation for the velocity of the particle in each dimension is computed by:

$$IF(rand() < c_v)\ THEN\ v_{id} = rand() * v_{\max,d} \qquad (2a)$$

The chaotic perturbation for the position of the particle in each dimension is computed by:

$$IF(rand() < c_l)\ THEN\ x_{id} = Random(l_d, u_d) \qquad (2b)$$

where C_v and C_l are the chaotic factors in the range [0,1] and $Random(l_d,u_d)$ is a uniform random number between l_d and u_d.

3 Dissipative PSO with Variable Probability of Adding Chaotic Perturbations

Putting forward the same argument as in dissipative version, our analysis reveals that this scheme, despite improving results significantly with respect to standard PSO and being competitive to many other variants, has some limitations and can be made much more effective.

We found that adding this chaotic perturbation to the system is most effective when done with a time dependant strategy. We specifically discovered that in the early iterations (approximately the first third) of the algorithm, it is better to re-initialise the particles with a high probability (~0.5 for each particle). By contrast, in the later iterations it is better to use a much smaller probability (~0.001), as used in dissipative PSO.

The underlying idea is that in the early iterations the overall fitness of the particles is low, so that it is effective to perform more exploration by using random re-initialisations. In this way particles find good quality *local* solutions, which they keep

on improving in the later iterations, so obtaining sustained development [7]. In other words, in earlier swarm iterations, particles are more vulnerable to environmental effects, whilst later on they rely more on the acquired knowledge of their best neighbours. Therefore their mutual non-linear interaction helps them to find better and better intermediate positions.

We have developed two versions of PSO based upon the idea of time-dependant random re-initialisation of particle positions. The first version uses the global neighbourhood topology, denoted as *gbest*; the second version uses a local neighbourhood topology, denoted as *lbest*. However, instead of keeping the neighbourhood constant as in conventional PSO, we introduce a dynamic neighbourhood topology.

In addition, we argue that for these two versions of global and local neighbourhoods, a dynamic and time varying local topology can make better use of the idea of randomly re-initialising particles to slow down the premature convergence. The idea of this dynamic, time-dependant neighbourhood is inspired by the concept of *small world networks* [10]. This local swarm is significantly more robust than the global version in much harder versions of the problem at hand, as shown later.

3.1 Locally Interacting Swarm with Small World Topology

As Mohan et al. ([8] and references therein) argue, particle positions in PSO oscillate in damped sinusoidal waves until they converge to the point in between their previous best and the global best position discovered by all particles so far. In this way particles converge to the global best position discovered so far. All particles following the same behaviour quickly converge to a good local minimum of the problem. It may be argued that many of the particles are wasting computational effort in moving toward the local minima already discovered. Whereas better results can be found if various particles explore other possible search directions.

Reasonable choices for deciding the interaction relationship between the particles can be drawn from observations on the social behaviour of animals. Many species of social animals try to keep acquaintanceship with a very small number of relatively fitter individuals. Croft et al. [9] observed that the empirical network between guppies can be closely approximated by small world networks.

Small world networks, as proposed by Watts (1998) [10], are networks, that lie in between regular lattice type structures and random networks. They tend to have a near-optimal trade-off between properties concerned with clustering and with the average distance between nodes. This can be seen in the examples given in figure 1. A comprehensive discussion of small world networks, their properties and examples can be found in [11].

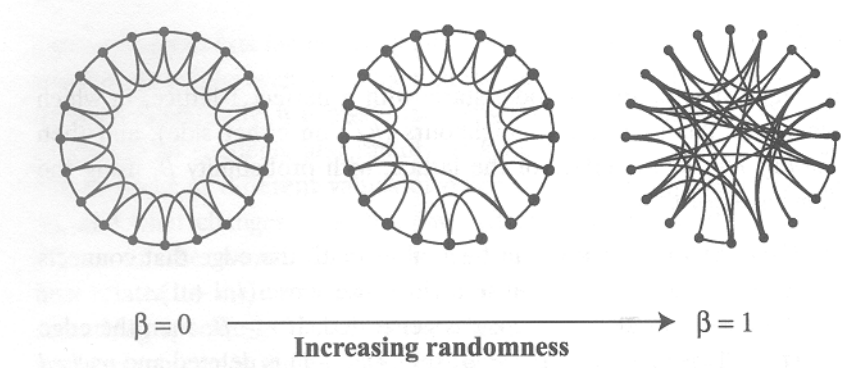

Fig.1. Small world networks lie between order and chaos, as illustrated by increasing probability of rewiring in a graph (from [11]).

We also believe that the agents should be "intelligent" enough to keep on breaking/making "friendships" based upon fitness. The rate at which particles change their neighbourhood relationships is specified by a probability of rewiring, i.e. taking an edge and connecting to a randomly chosen other node.

However, in our work rewiring is *not* done entirely at random, rather by using a strategy where each particle selects its neighbours with a probability proportional to their fitness. So we define here a probability of selecting particle *j* as a friend/neighbour of particle *i* as follows:

$$P_{ij} = \text{Rank}(j) / (\text{total number of particles}) \quad (3)$$

where all particles are ranked by assigning the highest number to the fittest individual, et cetera. For example in a population of twenty particles, the fittest particle would be assigned rank 20. Details of this process are explained in the pseudo-code below (Algorithm 1).

3.2 Edge Initialisation for Local PSO

We initialise the connections between the particles using two practically equivalent schemes. One possible start is a ring with two nearest neighbours. Another, which is more consistent with the rewiring scheme as well, is to throw edges equal to double of the number of the agents so that the average connectivity is two. In this scheme, for each edge we select one node randomly and the probability with which another randomly selected node is connected to the first one is given by the equation (3). A pseudo-code description of our PSO is given as Algorithm 1.

```
For each particle i, initialize the d^th dimension randomly
   in the range (-X_i,dmax, X_i,dmax).
Initialize the edges among particles
For each iteration t =1,…,G_max
     For each particle i =1,…,m
          For each dimension d =1,…,D
                V_id= w*V_id + C_1 rand*(P_id-X_id)
                     + C_2 rand*(P_1d-X_id);
               //P is best position by current
               //particle and P_1 is the best
              // Position in the current
              // neighbourhood of the particle
              // Limit velocity magnitude
              V  = min (V_max, max (-V_max, V) ) ;
             //  Update Position
             X_id = X_id + V_id;
         End for each dimension d
      Compute fitness of current particle
        and, if needed, update the
        historical information.
    End for each particle i
   Rewire K randomly selected edges with probability p.
End for each iteration t
```

Algorithm 1. Pseudo-code for PSO with local time-varying topology

4 Computational Results

4.1 Experimental Design and Benchmark Function

We have done experiments with two commonly used test functions (see e.g. [13]), which are very difficult to optimise. We have compared the two versions (global and local neighbourhood) of our new PSO algorithm—described in the previous section—with two other PSO algorithms, namely Standard PSO and Dissipative PSO. The two functions reported here both have global minimum at origin. The first one is generalized Rastrigin function (f_1):

$$f_1 = \sum_{d=1}^{D} \left(x_d^2 - 10 \cos \left(2 \pi x_d \right) + 10 \right) \qquad (4)$$

Rastrigin's function is based on a function with the addition of cosine modulation to produce many local minima. Thus, the test function is highly multimodal. However, the locations of the minima are regularly distributed.

The second function is the Rosenbrock function (f_2):

$$f_2 = \sum_{d=1}^{D} 100\,(X_d - X_{d-1}^2)^2 + (X_{d-1} - 1)^2 \qquad (5)$$

Rosenbrock's valley is a classic optimisation problem, and this function is also known as the Banana function. The global optimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial, however convergence to the global optimum is difficult and hence this problem has been repeatedly used in assessing the performance of optimisation algorithms.

For all the dimensions $d=1,\ldots,D$, $x_{\max,d}$ = 500 for both functions and the initialization range is $x_d \in [-x_{\max,d}, x_{\max,d}]$. Maximum velocity is $V_{\max} = x_{\max,d}$; The acceleration constants are $C_1 = 2$ and $C_2 = 2$. The fitness value is the function value. The results of our experiments are averaged over 30 runs, except where stated otherwise. We report results for population sizes of 50 and 100 particles only, mainly to avoid bulky tables. In all versions, we used time decreasing inertia [13].

In figure 2 we show performance results for varying the rewiring probability. From this we can see that the best performance is obtained when this probability is in the range 0.1–0.15. Therefore we have chosen to use 0.15 in the remainder of the experiments in the paper.

4.2 Results and Discussion

Tables 1 and 2 report the results for the Rosenbrock and Rastrigin function, respectively. Each table gives a detailed comparison of the results for standard PSO (SPSO) and dissipative PSO (DPSO), and the two PSO versions that we developed (GPSO and LPSO). DPSO1 in the tables is a slight variation of dissipative PSO with inertia weight fixed at 0.4—a parameter value which was also used by Xie et al. [7]. GPSO stands for the *gbest* version of standard PSO with the chaotic perturbations introduced in section 3. LPSO stands for the *lbest* version with small world like topological relationship along with the same scheme of chaotic perturbation. Each cell of these tables shows the mean fitness value of the best particle found by the corresponding version of PSO. m denotes the number of particles in the population, and $G_{\max}$ denotes the number of iterations of the PSO.

It is clear from the table 1 that for different numbers of dimensions and numbers of iterations the two new algorithms outperform the standard and dissipative PSO in all settings. GPSO and especially LPSO are more robust as the number of dimensions is increased. In this case LPSO performs reasonably better than GPSO, due mainly to the strategy of adding perturbation to the system.

Table 2 shows a similar analysis for the Generalized Rastrigin function and the same argument holds there, except that GPSO is closer in performance to LPSO, in this case. In more difficult settings, (i.e. higher number of dimensions of the Rosenbrock function), we see a clear advantage of the locally interacting LPSO over the globally interacting GPSO. Figure 3 shows such a comparison of LPSO with respect to GPSO and other PSO variants previously discussed. LPSO scales very well with the dimensionality (hardness) of the problem.

Figure 2 shows the dependence of LPSO over the rewiring probability which indicates that a greater than zero (of course less than 1) rewiring probability is better

than fixed topology (rewriting probability zero). Nonetheless, rewiring probability values in the range 0.15-0.20 are enough to achieve good results.

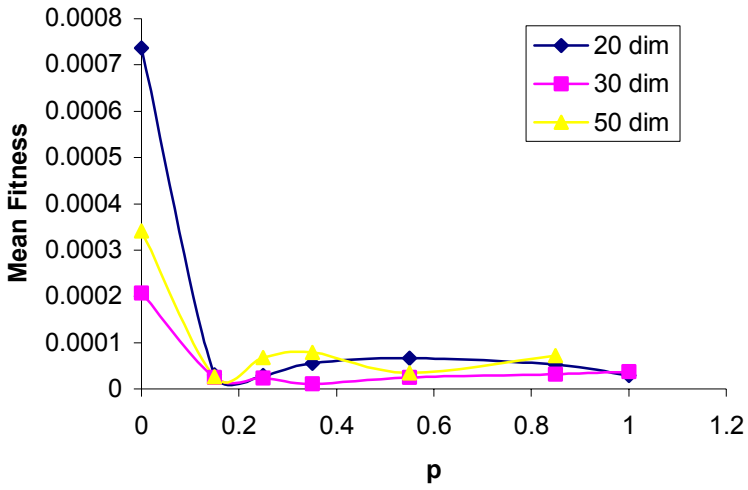

Fig.2. Average performance (over 50 runs) vs rewiring probability for LPSO

Another point to note is that in [7], the authors found that dissipative PSO is more effective (at least for the Rastrigin function) if the inertia weight is fixed at 0.4 rather than decreasing with time. We used both versions of DPSO, as in tables 1 and 2, for both functions (Rastrigin and Rosenbrock). An inconsistency is that with same algorithm as described in [7], fixing w=0.4 performs similarly to original DPSO in case of the Rosenbrock function, while it does not in case of the Rastrigin function. This inconsistency is not present (on average) in GPSO and LPSO. Rather, time varying inertia is still a reasonably better candidate in our opinion, based on the experiments.

Figure 3 shows the clear advantage of using LPSO over GPSO for the Rosenbrock function for a number of dimensions greater than 40. Even for fewer dimensions LPSO finds near-optimum solutions earlier than GPSO, as is clear from figure 3;while in case of the Rastrigin function both of the algorithms find the global minimum, but LPSO finds it in fewer iterations.

As can be observed in figure 4,our two new versions of PSO do not suffer from stagnation. They find good quality solutions in the earlier steps of the exploration, and then continue to find better solutions when the traditional variants of the algorithm have ceased improvement.

Figure 4 shows a comparison of different variants of PSO for 30 dimensional Rosenbrock (4a) and Rastrigin (4b) functions. These illustrate that our two new algorithms both are capable of the desirable “sustained development” property. The variants illustrated are the standard PSO algorithm (SPSO), dissipative PSO (DPSO), a variant on dissipative PSO with inertia weight fixed at 0.4 (DPSO1), and our GPSO and LPSO algorithms.

Table 1: Mean Fitness for Rosenbrock function

m	Dim	Gmax	SPSO	DPSO	DPSO1	GPSO	LPSO
50	10	1000	341.14	148.3341	163.3249	0.480304	0.304061
		2000	191.1035	71.03811	70.26753	0.118322	0.000074
		3000	273.6187	6.704371	21.04526	0.000362	0.000005
	20	1000	2641.243	192.2827	183.4044	2.167039	1.43718
		2000	381.8313	158.2822	122.8941	0.495267	0.001106
		3000	231.5867	38.80004	52.3064	0.001013	0.000156
	30	1000	12807.11	527.6025	250.2157	3.800333	3.696746
		2000	768.244	72.62108	239.0663	0.025852	0.000372
		3000	698.212	32.42213	49.98738	0.837883	0.000443
100	10	1000	232.2697	24.51027	108.5668	0.131957	0.000117
		2000	158.6715	2.635007	34.76893	0.000277	0.000001
		3000	144.2241	10.56071	19.37426	0.000082	0
	20	1000	562.5305	82.65499	110.8093	0.5198	0.500947
		2000	330.7465	18.88946	24.30801	0.00069	0.000059
		3000	238.1916	24.70273	40.67028	0.000417	0.000002
	30	1000	14229.18	428.8488	260.206	0.164011	0.011915
		2000	601.6497	50.94217	58.09047	0.002213	0.000679
		3000	481.476	23.65155	18.82164	0.000588	0.000022

Table 2: Mean Fitness for Rastrigin Function

m	Dim	Gmax	SPSO	DPSO	DPSO1	GPSO	LPSO
50	10	1000	4.164868	5.016658	4.013005	1.260285	1.591936
		2000	2.886982	2.367004	1.992383	0.822669	0.630141
		3000	2.55558	1.606797	1.770281	0.165827	0.530645
	20	1000	28.197018	29.025015	24.124841	2.852309	3.91351
		2000	21.823471	18.407085	15.008319	1.691578	0.596976
		3000	18.630007	15.60987	12.642538	1.028139	0
	30	1000	92.064652	71.495613	57.198965	9.299062	10.911399
		2000	51.326238	56.932776	47.700015	5.241589	1.98992
		3000	41.109071	43.85957	31.746499	3.482368	0.962423
100	10	1000	2.787931	2.555413	2.056251	0.630141	0.331653
		2000	1.392944	1.392945	1.061291	0	0
		3000	0.995646	0.971996	0.795968	0	0
	20	1000	19.563307	21.598892	15.225288	2.752967	1.227118
		2000	14.610109	16.329376	9.810171	0.563811	0
		3000	11.276265	10.569544	6.441457	0	0
	30	1000	68.321198	58.365233	38.607121	3.750204	0.961795
		2000	39.203337	37.342493	25.864482	0.895464	0
		3000	32.460681	31.96258	18.602715	0	0

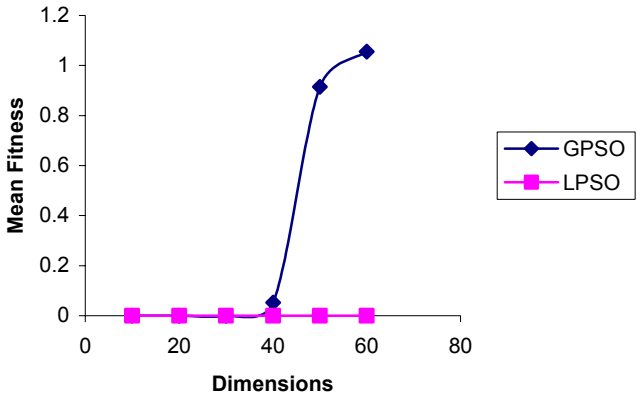

Fig. 3. Performance of GPSO and LPSO for increasing dimensions of the Rosenbrock function for m=100, G_{max} = 3000.

5 Conclusions

We have proposed two new variants on dissipative PSO, based on time-dependent variation of the probability of re-initialising the particles. One of these two versions (LPSO) also incorporates additional ideas drawn from small world networks, which are used to adjust the topology of the particle neighbourhoods. We have compared these two new algorithms on two well-known function optimisation problems, and they have been shown to perform better than both standard PSO and the original dissipative PSO. In particular they demonstrate the "sustained development" property, the lack of which causes premature convergence to a local optimum. Furthermore, the local neighbourhood version performs better than the global neighbourhood version for problems with a large number of dimensions.

Future work will focus on extending these ideas to more challenging problem domains, in particular moving beyond simple function optimisation problems. Another direction will be a more extensive analysis of how particular strategies for re-initialisation influence the performance of the system.

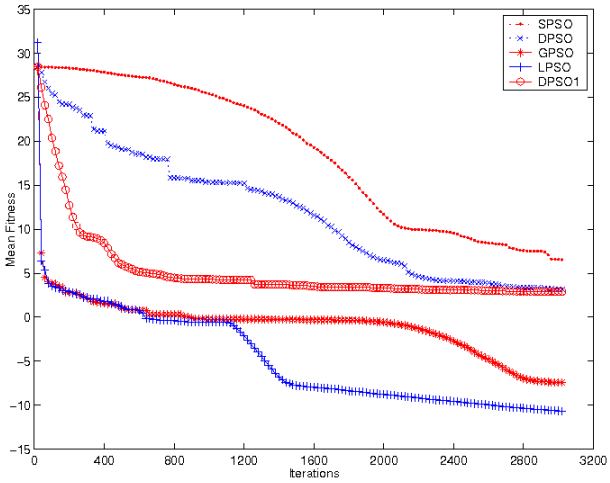

Fig. 4 (a). Rosenbrock function , dimension=30, m=100, G_{max}=3000.

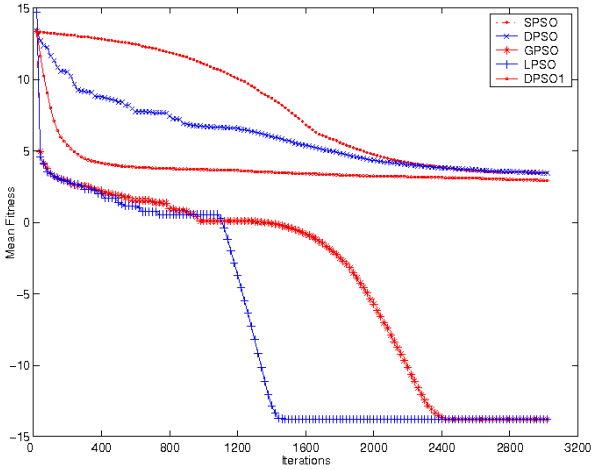

Fig. 4 (b). Rastrigin function, dimension=30, m=100, G_{max}=3000

Table 3. Mean and standard deviation of final value for experiments in figure 4.

	SPSO	DPSO	DPSO1	GPSO	LPSO
Rosenbrock	481.5, 932.3	23.6, 29.8	18.8, 23.6	0.00058, 0.0008	0.00002, 0.0004
Rastrigin	32.5, 7.5	32.0, 21.2	18.6, 8.3	0, 0	0, 0

Acknowledgements

This work was supported by the EPSRC under grant GR/T11265/01 (*eXtended Particle Swarms*). The first author was additionally supported by a scholarship from the University of Kent Computing Laboratory.

References

1. J. Kennedy and R. Eberhart. Particle swarm optimization, *Proc. IEEE Int. Conf. on Neural Networks,* pp. 1942-1948, 1995.
2. R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory, *Proc. 6 th Int. Symposium on Micro Machine and Human Science*, pp. 39-43, 1995.
3. Y. Shi and R. Eberhart. Parameter selection in particle swarm optimization, *Proc. 7th Annual Conf. on Evolutionary Programmin*g, pp. 591-600, 1998.
4. R. Eberhart and Y. Shi. Particle swarm optimization: developments, applications and resources, *Proc. IEEE Int. Conf. on Evolutionary Computatio*n, pp. 81-86, 2001.
5. J. Kennedy. Stereotyping: improving particle swarm performance with cluster analysis, *Proc. IEEE Int. Conf. on Evolutionary Computation*, pp. 1507-1512, 2000.
6. P. J. Angeline. "Evolutionary optimization versus particle swarm optimization: philosophy and performance difference," *Proc. 7th Annual Conf. on Evolutionary Programming*, pp. 601-610, 1998.
7. X, Xie, W, Zang and Z Yang. "A dissipative swarm optimisation, *Proceedings of the IEEE Congress on Evolutionary Computing (CEC 2002)*, Honolulu, Hawaii USA, May 2002.
8. E. Ozcan and C. Mohan. Particle swarm optimization: surfing the waves. *Proc. 1999 Congress on Evolutionary Computation*, 1939–1944. Piscataway, NJ: IEEE Service Center, 1999.
9. D. P. Croft and J. Krause, and R. James. Social networks in the guppy (Poecilia reticulata). *Proceedings of the Royal Society of London: Biology Letters*, pp. S516-S519, 2004.
10. D.J. Watts and S. Strogatz. Collective dynamics of 'Small world' networks. *Nature* 393, pp. 440-442, 1998.
11. D.J. Watts. *Small Worlds: The Dynamics of Networks Between Order and Randomness*. Princeton University Press, 1999.
12. Y. Shi and R. Eberhart. A modified particle swarm optimizer, *Proc. IEEE Int. Conf. on Evolutionary Computation*, pp. 69-73, 1998.
13. Y. Shi. Particle swarm optimisation code, `http://www.engr.iupui.edu/~shi`