Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Improved Catfish Particle Swarm Optimization with Embedded Chaotic Map

Li-Yeh Chuang
Dep. of Chemical Engineering
I-Shou University
Kaohisung, Taiwan
chuang@isu.edu.tw

Sheng-Wei Tsai, and Cheng-Hong Yang
Dep. of Electronic Engineering
National Kaohsiung University of Applied Sciences
Kaohisung, Taiwan
1096305108@cc.kuas.edu.tw, and chyang@cc.kuas.edu.tw

***Abstract*—Chaotic catfish particle swarm optimization (C-CatfishPSO) is a novel optimization algorithm proposed in this paper. C-CatfishPSO introduces chaotic maps into catfish particle swarm optimization (CatfishPSO), which increase the search capability of CatfishPSO via the chaos approach. Simple CatfishPSO relies on the incorporation of catfish particles into particle swarm optimization (PSO). The introduced catfish particles improve the performance of PSO considerably. Unlike other ordinary particles, catfish particles initialize a new search from extreme points of the search space when the *gbest* fitness value (the best previously encountered value) has not changed for a certain number of consecutive iterations. This results in further opportunities of finding better solutions for the swarm by guiding the entire swarm to promising new regions of the search space, and by accelerating search efficiency. In this study, we adopted chaotic maps to strengthen the solution quality of PSO and CatfishPSO. After the introduction of chaotic maps into the process, the improved PSO and CatfishPSO are called chaotic PSO (C-PSO) and chaotic CatfishPSO (C-CatfishPSO), respectively. PSO, C-PSO, CatfishPSO and C-CatfishPSO were extensively compared on six benchmark functions. Statistical analysis of the experimental results indicates that the performance of C-CatfishPSO is better than the performance of PSO, C-PSO, and CatfishPSO.**

***Keywords*—Chaos, Chaotic map, Swarm Intelligence, Catfish Particle Swarm Optimization**

I. INTRODUCTION

Generating an ideal random sequence is of great importance in the fields of numerical analysis, sampling and heuristic optimization. Recently, a technique which employs chaotic sequences via the chaos approach (chaotic maps) has gained a lot of attention and been widely applied in different areas, such as the chaotic neural network (CNN) [1], chaotic optimization algorithms (COA) [2, 3], nonlinear circuits [4], DNA computing [5], and image processing [6]. All of the above-mentioned methods rely on the same pivotal operation, namely the adoption of a chaotic sequence instead of a random sequence, and thereby improve the results due to the unpredictability of the chaotic sequence [7].

Chaos can be described as a bounded nonlinear system with deterministic dynamic behavior that has ergodic and stochastic properties [8]. It is very sensitive to the initial conditions and the parameters used. In other word, cause and effect of chaos are not proportional to the small differences in the initial values. In what is called the "butterfly effect", small variations of an initial variable will result in huge differences in the solutions after some iteration. Mathematically, chaos is random and unpredictable, yet it also possesses an element of regularity [7].

Particle swarm optimization (PSO) is a stochastic, population-based evolutionary computer algorithm developed by Kennedy and Eberhart in 1995 [9]. PSO simulates the social behavior of organisms to describe an automatically evolving system. PSO has been successfully employed in many areas, and generally obtains better results in a faster, cheaper way compared to other methods (http://www.particleswarm.info/). PSO shows a promising performance on nonlinear function optimization and has thus received much attention [10]. However, the local search capability of PSO is poor [11] since premature convergence occurs often, especially in the case complex multi-peak search problems [12]. Recently, numerous improvements, which rely on the chaos approach, have been proposed for PSO in order to overcome this disadvantage. Chaotic maps can easily be implemented and avoid entrapment in local optima [13-17]. The inherent characteristics of chaos can improve PSO by enabling it to escape from local solutions, and thus boost the global search capability of PSO [15]. Most PSO methods use the same chaotic map, namely a logistic map [16]. Logistic maps were introduced in nonlinear dynamics of biological populations evidencing chaotic behavior [18] and are often cited as an archetypal example.

In this paper, we propose chaotic CatfishPSO (C-CatfishPSO), in which chaotic maps are applied to improve the performance of the CatfishPSO algorithm. In CatfishPSO, the catfish effect is applied to improve the performance of PSO. This effect is the result of the introduction of new particles into the search space ("catfish particles"), which replace particles with the worst fitness; these catfish particles are initialized at extreme points of the search space when the fitness of the global best particle has not improved for a certain number of consecutive iterations. This results in further opportunities of finding better solutions for the swarm by guiding the whole swarm to promising new regions of the search space [19]. The logistic map was introduced in our study to improve the search behavior and to prevent entrapment of the particles in a locally optimal solution. The proposed method was applied to six benchmark functions from the literature. Statistical analysis of the experimental results shows that the performance of C-CatfishPSO is superior to PSO, C-PSO, and CatfishPSO.

II. Method

A. Particle Swarm Optimization (PSO)

In original PSO [9], each particle is analogous to an individual "fish" in a school of fish. It is a population-based optimization technique, where a population is called a swarm. A swarm consists of N particles moving around in a D-dimensional search space. The position of the i_{th} particle can be represented by $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$. The velocity for the i_{th} particle can be written as $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. The positions and velocities of the particles are confined within $[X_{\min}, X_{\max}]^D$ and $[V_{\min}, V_{\max}]^D$, respectively. Each particle coexists and evolves simultaneously based on knowledge shared with neighboring particles; it makes use of its own memory and knowledge gained by the swarm as a whole to find the best solution. The best previously encountered position of the i_{th} particle is denoted its individual best position $p_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$, a value called $pbest_i$. The best value of all individual $pbest_i$ values is denoted the global best position $g = (g_1, g_2, \ldots, g_D)$ and called $gbest$. The PSO process is initialized with a population of random particles, and the algorithm then executes a search for optimal solutions by continuously updating generations. At each generation, the position and velocity of the i_{th} particle are updated by $pbest_i$ and $gbest$ in the swarm. The update equations can be formulated as:

$$\begin{aligned} v_{id}^{new} &= w \times v_{id}^{old} + c_1 \times r_1 \times \left(pbest_{id} - x_{id}^{old}\right) \\ &\quad + c_2 \times r_2 \times \left(gbest_d - x_{id}^{old}\right) \end{aligned} \tag{1}$$

$$x_{id}^{new} = x_{id}^{old} + v_{id}^{new} \tag{2}$$

In Eqs. (1) and (2), r_1 and r_2 are random numbers between (0, 1), and c_1 and c_2 are acceleration constants which control how far a particle will move in a single generation. Velocities v_{id}^{new} and v_{id}^{old} denote the velocities of the new and old particle, respectively. x_{id}^{old} is the current particle position, and x_{id}^{new} is the new, updated particle position. The inertia weight w controls the impact of the previous velocity of a particle on its current one [20]. In general, the inertia weight is linearly decreased from 0.9 to 0.4 throughout the search process in order to effectively balance the global and local search abilities of the swarm [21]. The corresponding equation can be written as:

$$w_{LDW} = \left(w_{\max} - w_{\min}\right) \times \frac{Iteration_{\max} - Iteration_i}{Iteration_{\max}} + w_{\min} \tag{3}$$

In Eq. (3), $w_{\max}$ is 0.9, $w_{\min}$ is 0.4 and $Iteration_{\max}$ is the maximum number of allowed iterations. The pseudo-code of the PSO process is shown below.

PSO pseudo-code

```
01: begin
02:   Randomly initialize particles swarm
03:   while (number of iterations, or the stopping criterion is not met)
04:     Evaluate fitness of particle swarm
05:     for $n$ = 1 to number of particles
06:       Find $pbest$
07:       Find $gbest$
08:       for $d$ = 1 to number of dimension of particle
09:         Update the position of particles by Eq. 1 and Eq. 2
10:       next $d$
11:     next $n$
12:     Update the inertia weight value by Eq. 3
13:   next generation until stopping criterion
14: end
```

B. Chaotic Particle Swarm Optimization (C-PSO)

In C-PSO, sequences generated by the logistic map substitute the random parameters r_1 and r_2 in PSO. The parameters r_1 and r_2 are modified by the logistic map based on the following equation.

$$Cr_{(t+1)} = k \times Cr_{(t)} \times (1-Cr_{(t)}) \tag{4}$$

In Eq. (4), $Cr_{(0)}$ is generated randomly for each independent run, with $Cr_{(0)}$ not being equal to {0, 0.25, 0.5, 0.75, 1} and k equal to 4. The driving parameter k of the logistic map controls the behavior of $Cr_{(t)}$ (as t goes to infinity). The behavior of the logistic map for various values of the parameter k is shown in Fig. 1-a. For low values of k ($k < 3$), Cr eventually converges to a single number. When $k = 3$, Cr oscillates between two values. This characteristic change in behavior is called a bifurcation. For $k > 3$, Cr goes through further bifurcations, eventually resulting in chaotic behavior. In fact, the bifurcation diagram is itself a fractal [22]. As Fig. 1-a shows, when k equals 4, the chaotic sequence value Cr is bounded within [0, 1]. Fig. 1-b also shows the chaotic Cr value using a logistic map for 100 iterations where $Cr_{(0)} = 0.001$.

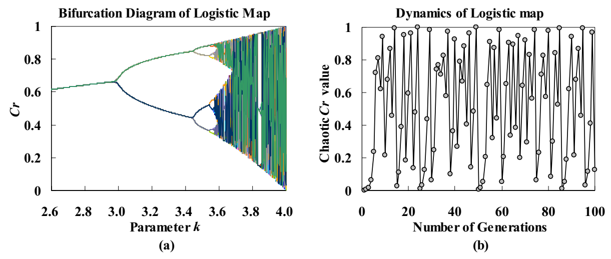


Figure 1. Bifurcation Diagram and Dymamics of Logistic Map

The velocity update equation for C-PSO can be formulated as:

$$\begin{aligned} v_{id}^{new} &= w \times v_{id}^{old} + c_1 \times Cr \times \left(pbest_{id} - x_{id}^{old}\right) \\ &\quad + c_2 \times (1 - Cr) \times \left(gbest_d - x_{id}^{old}\right) \end{aligned} \tag{5}$$

In Eq. (5), Cr is a function based on the results of the logistic map with values between 0.0 and 1.0. The pseudo-code of C-PSO is shown below.

C-PSO pseudo-code

```
01: begin
02:   Randomly initialize particles swarm
03:   Randomly generate $Cr_{(0)}$
04:   while (number of iterations, or the stopping criterion is not met)
05:     Evaluate fitness of particle swarm
06:     for $n$ = 1 to number of particles
07:       Find $pbest$
08:       Find $gbest$
09:       for $d$ = 1 to number of dimension of particle
10:         Update the Chaotic $Cr$ value by Eq. 4
11:         Update the position of particles by Eq. 5 and Eq. 2
12:       next $d$
13:     next $n$
14:     Update the inertia weight value by Eq. 3
15:   next generation until stopping criterion
16: end
```

C. Catfish Particle Swarm Optimization (CatfishPSO)

The underlying idea for the development of CatfishPSO was derived from the observation of the catfish effect when catfish were introduced into large holding tanks of sardines

[19]. The catfish—in competition with the sardines—stimulate renewed movement amongst the sardines. Similarly, the introduced catfish particles stimulate a renewed search by the other "sardine" particles in CatfishPSO. In other words, the "catfish" particles lead the "sardine" particles, which are trapped in a local optimum, on to a new region of the search space, and thus to potentially better particle solutions. These catfish particles are essential for the success of a given optimization task. The pseudo-code of the CatfishPSO is shown below. Further details on CatfishPSO mechanisms can be found in Chuang *et al.* [19].

CatfishPSO Pseudo-code

```
01: Begin
02:  Randomly initialize particles swarm
03:  while (number of iterations, or the stopping criterion is not met)
04:    Evaluate fitness of particle swarm
05:    for $n$ = 1 to number of particles
06:      Find pbest
07:      Find gbest
08:      for $d$ = 1 to number of dimension of particle
09:        Update the position of particles by Eq. 1 and Eq. 2
10:      next d
11:    next n
12:    if fitness of gbest is the same Seven times then
13:      Sort the particle swarm via fitness from best to worst
14:      for $n$ = number of Nine-tenths of particles to number of particles
15:        for $d$ = 1 to number of dimension of particle
16:          Randomly select extreme points at Max or Min of the search space
17:          Reset the velocity to 0
18:        next d
19:      next n
20:    end if
21:    Update the inertia weight value by Eq. 3
22:  next generation until stopping criterion
23: end
```

D. Chaotic Catfish Particle Swarm Optimization (C-CatfishPSO)

In C-CatfishPSO, a logistic map is embedded into CatfishPSO, which updates the parameters r_1 and r_2 based on Eq. (4) with k equal to 4. The chaotic sequence value Cr is bounded within [0, 1] if k equals 4. The logistic map improves the search capability of CatfishPSO significantly by ergodic and stochastic properties. The particle velocities are updated according to Eq. (5). The pseudo-code for C-CatfishPSO is shown below.

C-CatfishPSO Pseudo-code

```
01: Begin
02:  Randomly initialize particles swarm
03:  Randomly generate $Cr_{(0)}$
04:  while (number of iterations, or the stopping criterion is not met)
05:    Evaluate fitness of particle swarm
06:    for $n$ = 1 to number of particles
07:      Find pbest
08:      Find gbest
09:      for $d$ = 1 to number of dimension of particle
10:        Update the Chaotic $Cr$ value by Eq. 4
11:        Update the position of particles by Eq. 5 and Eq. 2
12:      next d
13:    next n
14:    if fitness of gbest is the same Seven times then
15:      Sort the particle swarm via fitness from best to worst
16:      for $n$ = number of Nine-tenths of particles to number of particles
17:        for $d$ = 1 to number of dimension of particle
18:          Randomly select extreme points at Max or Min of the search space
19:          Reset the velocity to 0
20:        next d
21:      next n
22:    end if
23:    Update the inertia weight value by Eq. 3
24:  next generation until stopping criterion
25: end
```

III. NUMERICAL SIMULATION

A. Benchmark functions

In order to illustrate, compare and analyze the effectiveness and performance of the PSO, C-PSO, CatfishPSO and C-CatfishPSO algorithms for optimization problems, six representative benchmark functions were used to test the algorithms. These six benchmark functions are shown below.

Ellipsoid

$$f_1(x) = \sum_{i=1}^{D} i x_i^2 \tag{6}$$

Rosenbrock

$$f_2(x) = \sum_{i=1}^{D-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right) \tag{7}$$

Rastrigrin

$$f_3(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right) \tag{8}$$

Griewark

$$f_4(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{9}$$

Ackley

$$f_5(x) = -20\exp^{\left(-0.2\times\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)} - \exp^{\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)} + 20 + e \tag{10}$$

Schwefel

$$f_6(x) = 418.9809D - \sum_{i=1}^{D} x_i \sin\left(\sqrt{|x_i|}\right) \tag{11}$$

These six benchmark functions can be grouped into unimodal functions (*Ellipsoid* and *Rosenbrock*) and multimodal functions (*Rastrigrin*, *Griewark*, *Ackley* and *Schwefel*); in multimodal functions the number of local minima increases exponentially with the problem dimension.

B. Parameter settings

In our experiments, three different dimension (Dim.) sizes were tested for each function, namely 10, 20 and 30 dimensions, and the corresponding maximum number of generations (Gen.) was set to 1000, 1500 and 2000, respectively. In addition, a population (Pop.) size of 20 was used for each function. The same sets of parameters were assigned for PSO, C-PSO, CatfishPSO and C-CatfishPSO, i.e. c_1=c_2=2 [9]. For each experimental setting, we executed 1000 independent runs for PSO, C-PSO, CatfishPSO and C-CatfishPSO with different random seeds. The parameter settings of the six benchmark functions are summarized in TABLE I.

TABLE I. PARAMETER SETTINGS OF THE SIX BENCHMARK FUNCTIONS

	Name	Search Space	Asymmetric Initialization Range	Optimal
F_1	***Ellipsoid***	$-100 \le x_i \le 100$	$50 \le x_i \le 100$	0
F_2	***Rosenbrock***	$-100 \le x_i \le 100$	$15 \le x_i \le 30$	0
F_3	***Rastrigrin***	$-10 \le x_i \le 10$	$2.56 \le x_i \le 5.12$	0
F_4	***Griewark***	$-600 \le x_i \le 600$	$300 \le x_i \le 600$	0
F_5	***Ackley***	$-100 \le x_i \le 100$	$50 \le x_i \le 100$	0
F_6	***Schwefel***	$-500 \le x_i \le 500$	$-500 \le x_i \le -250$	0

C. *Experimental Results and Discussion*

In this section, the performances of PSO, C-PSO, CatfishPSO and C-CatfishPSO are compared by means of the best fitness and the standard deviation among six benchmark functions under equal conditions. In order to authenticate whether the proposed method is significantly different from other methods or not, we adopted a z-test on pairs of groups of results. The z-test gives a *p-value* which is compared to a constant called α to determine whether a difference between alternative method is significant or not. In our case, we adopted a 95% confidence interval in the z-test for all benchmark functions. In other words, if *p-value* $<\alpha = 0.05$ then a test is reported as significant. The mean fitness values and standard deviations of PSO, C-PSO, CatfishPSO and C-CatfishPSO for these six benchmark functions are listed in TABLE II. If any mean fitness values and standard deviations are $< 10^{-300}$, TABLE II will display 0.000±0.000 instead. The experimental results in TABLE II are divided into three areas for analysis:

- C-PSO compared to PSO

In C-PSO, a chaotic map was embedded to determine the PSO parameters r_1 and r_2. The PSO parameters r_1 and r_2 cannot ensure optimal ergodicity in the search space because they are absolutely random [14] i.e. the r_1 and r_2 are generated by a linear congruential generator (LCG) with a random seed. The generated sequence of LCG consists of pseudo-random numbers that have periodic characteristics [23]. Furthermore, the generated sequence of a logistic map also consists of pseudo-random numbers, but there are no fixed points, periodic orbits, or quasi-periodic orbits in the behavior of the chaos system [22]. As a result, the system can avoid entrapment in local optima [15]. In terms of mean fitness values, TABLE II indicate that C-PSO outperformed PSO on the *Ellipsoid*, *Rosenbrock*, *Rastrigrin*, *Griewark*, *Ackley* and *Schwefel* benchmark functions. Amongst these six benchmark functions, C-PSO seems superior to PSO. However, this result is misleading since the performance of C-PSO is as the z-test at $\alpha = 0.05$ proved, only statistically superior in the *Ackley* function.

- CatfishPSO compared to PSO

TABLE II indicate that CatfishPSO outperformed PSO on the *Ellipsoid*, *Rosenbrock*, *Rastrigrin*, *Griewark*, and *Ackley* benchmark functions, a fact that is supported by the z-test at $\alpha = 0.05$. In PSO, each particle only relies on its individual *pbest* value and the global best position *gbest* to update its position at each generation. If *gbest* is trapped in a local optimum, the particles cluster together and lose their ability to explore the search space in later generations. In order to avoid such a scenario, the worst 10% of the swarm are replaced by catfish particles when *gbest* has not changed for a certain number of generations.

After the catfish particles are introduced, they initialize a renewed search from extreme points of the search space, and thus find better solutions by guiding the entire swarm to promising new regions. They also improve the search efficiency of the swarm [19]. The catfish particles not only allow the swarm to discover better solutions within the area of the swarm itself, but also to obtain better solutions located outside the swarm area.

- C-CatfishPSO compared to both C-PSO and CatfishPSO

TABLE II further shows that C-CatfishPSO outperformed CatfishPSO and C-PSO on the *Ellipsoid*, *Rosenbrock*, *Rastrigrin*, *Griewark*, *Ackley* and *Schwefel* benchmark functions. For the majority of these six benchmark functions, CatfishPSO proved to be superior to C-PSO. However, the quality of solutions achieved by CatfishPSO can further be improved when chaotic maps are embedded (C-CatfishPSO), since the rank of C-CatfishPSO in the all benchmark functions is higher than the rank of CatfishPSO. The rank indicates the numbers of significance at $\alpha=0.05$ determined with a z-test between the tested method and another method. The experimental results and statistical analyses clearly demonstrate that the performance of C-CatfishPSO is superior to PSO, C-PSO and CatfishPSO.

Fig. 2 plots the mean best fitness in the form of logarithm values over the number of generations for PSO, C-PSO, CatfishPSO and C-CatfishPSO with 20 particles on six 30-dimensional functions. These figures clearly show that the search efficiency of C-CatfishPSO is superior to that of PSO, C-PSO, and CatfishPSO on all six benchmark functions. At the same time, the best solution obtained by C-CatfishPSO is also superior to the best solution obtained by PSO, C-PSO and CatfishPSO. TABLE II indicates that CatfishPSO and C-CatfishPSO are capable of finding optimal solutions in *Ellipsoid*, *Rastrigrin* and *Griewark* benchmark functions, where the fitness value was $< 10^{-300}$. For the sake of convenience, we show the graphs in Figs. 2-a, 2-c and 2-d only up to the bottom of the chart, and demonstrate that CatfishPSO and C-CatfishPSO are capable of finding optimal solutions within 1500 generations, a fact supported by TABLE II, in which no standard deviation can be given for either CatfishPSO or C-CatfishPSO for the *Ellipsoid*, *Rastrigrin* and *Griewark* benchmark functions.

In Figs. 2-a to 2-f, we can observe a peculiar feature of CatfishPSO. The curve of CatfishPSO is displayed as step shape, and this phenomenon is indicative of the CatfishPSO's capability to breakthrough and leave in local optimal solution. The curve of C-PSO always obtained the best solution early on. Moreover, it does not have the same to breakthrough the local optimal solution as CatfishPSO. In summary, it can be concluded that using a chaotic map to substitute the random parameters r_1 and r_2 is a significant approach to improve the performance of CatfishPSO in unimodal and multimodal different trait benchmark functions.

IV. Conclusion

In this paper, Chaotic CatfishPSO (C-CatfishPSO) has been introduced, which adopts a chaotic map to improve the performance over the CatfishPSO algorithm. In CatfishPSO, catfish particles initialize a new search from extremes of the search space when the *gbest* value has not progressed for a certain number of iterations. Better solutions can be found by guiding the entire swarm to more promising regions in the search space. C-CatfishPSO achieved far better performance than PSO, C-PSO and CatfishPSO. This performance was validated by statistical analysis with a z-test at $\alpha = 0.05$ on representative benchmark functions, under equal conditions. It can be concluded that the introduction of new individuals

TABLE II. MEAN FITNESS VALUES FOR SIX BENCHMARK FUNCTIONS

Benchmark functions	Pop.	Dim.	Gen.	PSO	C-PSO	CatfishPSO	C-CatfishPSO
	20	10	1000	8.94E-22±6.01E-21	600.00±2398.979	8.09E-26±5.72E-25	**0.000±0.000**
		20	1500	2200.00±4184.520	400.00±1979.487	8.64E-09±6.09E-08	**0.000±0.000**
		30	2000	2200.00±4184.520	2.47E-23±1.74E-22	2.46E-05±8.94E-05	**0.000±0.000**
F*$_1$ *Ellipsoid			Average	1466.667±2789.680	333.333±1459.489	8.20E-06±5.46E-01	**0.000±0.000**
			P-Value	0.767	0.436	**0.0**	**0.0**
	Significant (P-Value <α=0.05)			No	No	**Yes**	**Yes**
			Rank	0	0	**2**	**3**
	20	10	1000	95.893±230.136	28.178±426.317	5.855±0.413	**3.597±3.708**
		20	1500	167.604±318.927	27.770±248.084	16.257±0.385	**4.527±6.290**
		30	2000	268.148±421.517	27.707±043.068	26.555±0.456	**4.359±7.528**
F*$_2$ *Rosenbrock			Average	177.215±323.527	27.885±239.157	16.228±0.418	**4.115±5.842**
			P-Value	0.772	0.406	**0.0**	**0.0**
	Significant (P-Value <α=0.05)			No	No	**Yes**	**Yes**
			Rank	0	0	**2**	**3**
	20	10	1000	5.128±02.627	4.399±02.753	**0.000±0.000**	**0.000±0.000**
		20	1500	22.021±07.176	12.675±07.048	**0.000±0.000**	**0.000±0.000**
		30	2000	47.735±12.037	22.693±10.869	**0.000±0.000**	**0.000±0.000**
F*$_3$ *Rastrigrin			Average	24.967±07.280	13.256±06.890	**0.000±0.000**	**0.000±0.000**
			P-Value	1.0	0.859	**0.0**	**0.0**
	Significant (P-Value <α=0.05)			No	No	**Yes**	**Yes**
			Rank	0	0	**2**	**2**
	20	10	1000	0.102±00.056	0.061±0.050	**0.000±0.000**	**0.000±0.000**
		20	1500	0.480±06.361	0.002±0.009	**0.000±0.000**	**0.000±0.000**
		30	2000	2.455±14.650	0.361±5.692	**0.000±0.000**	**0.000±0.000**
F*$_4$ *Griewark			Average	1.012±07.022	0.141±1.917	**0.000±0.000**	**0.000±0.000**
			P-Value	0.582	0.439	**0.0**	**0.0**
	Significant (P-Value <α=0.05)			No	No	**Yes**	**Yes**
			Rank	0	0	**2**	**2**
	20	10	1000	19.791±2.351	8.92E-07±2.82E-05	**8.88E-16±9.96E-32**	**8.88E-16±9.87E-32**
		20	1500	19.999±1.66E-05	2.91E-14±5.79E-13	**8.88E-16±9.96E-32**	**8.88E-16±9.87E-32**
		30	2000	19.983±0.498	1.74E-14±4.58E-15	8.54E-02±5.46E-01	**8.88E-16±9.87E-32**
F*$_5$ *Ackley			Average	19.924±0.950	2.97E-07±9.40E-06	2.85E-02±5.46E-01	**8.88E-16±9.87E-32**
			P-Value	1.0	**0.0**	**0.0**	**0.0**
	Significant (P-Value <α=0.05)			No	**Yes**	**Yes**	**Yes**
			Rank	0	**1**	**1**	**3**
	20	10	1000	2037.388±163.855	1894.348±163.581	2022.755±147.039	**1825.659±162.621**
		20	1500	4501.628±153.607	4307.703±182.864	4410.713±142.607	**4221.962±134.176**
		30	2000	6922.967±147.999	6694.397±193.135	6784.719±129.244	**6591.520±134.252**
F*$_6$ *Schwefel			Average	4487.328±155.154	4298.816±179.860	4406.062±139.630	**4213.047±143.683**
			P-Value	0.960	0.280	0.784	**0.027**
	Significant (P-Value <α=0.05)			No	No	No	**Yes**
			Rank	0	0	0	**3**

Legend: Rank indicates the numbers of significance at α=0.05 between tested method and another method as indicated by a Z-test

(catfish particles) into a group has a significantly positive effect on the entire swarm. Chaotic maps were used to effectively improve the solution quality of CatfishPSO since they avoid entrapment in local optima by virtue of their characteristics of ergodic orbits and aperiodicity. Our experimental results indicate that C-CatfishPSO integrates the advantages of C-PSO and CatfishPSO, i.e., the capability to breakthrough local optimal solutions in unimodal and multimodal global optimization problem.

ACKNOWLEDGMENT

This work is partly supported by the National Science Council in Taiwan under grant NSC97-2622-E-151-008-CC2, NSC96-2221-E-214-050-MY3, NSC96-2622-E-151-019-CC3, NSC96-2622-E-214-004-CC3, NSC95-2221-E-151-004-MY3, NSC95-2221-E-214-087 and NSC95-2622-E-214-004-CC3.

REFERENCES

[1] K. Aihara, T. Takabe and M. Toyoda, "Chaotic neural networks," *Physics Letters A*, Vol. 144, pp. 333-340, 1990.

[2] B. Li and W.S. Jiang, "Optimizing complex functions by chaos search," *Cybernetics and Systems*, Vol. 29, No. 4, pp. 409-419, 1998.

[3] Z. Lu, L. S. Shieh and G. R. Chen, "On robust control of uncertain chaotic systems: a sliding-mode synthesis via chaotic optimization," *Chaos, Solitons & Fractals*, Vol. 18, No. 4, pp.819-827, 2003.

[4] P. Arena, R. Caponetto, L. Fortuna, A. Rizzo and M. L. Rosa, "Self organization in non recurrent complex system," *International Journal of Bifurcation and Chaos*, Vol. 10, No. 5, pp. 1115-1125, 2000.

[5] G. Manganaro and J. Pineda de Gyvez, "DNA computing based on chaos," *in Proceedings of the 1997 IEEE international conference on evolutionary computation*, Piscataway, 1997, pp. 255-260.

[6] H. Gao, Y. Zhang, S. Liang and D. Li, "A new chaotic algorithm for image encryption," *Chaos, Solitons & Fractals*, Vol. 29, Issue 2, pp. 393-399, July 2006.

[7] B. Alatas, E. Akin and A. Bedri Ozer, "Chaos embedded particle swarm optimization algorithms," *Chaos, Solitons & Fractals*, In Press, Corrected Proof, Available online 30 October 2007.

[8] H. G. Schuster, "Deterministic chaos: an introduction," 2nd revised ed. Weinheim, Federal Republic of Germany: Physick-Verlag GmnH; 1988.

[9] J. Kennedy and R.C. Eberhart, “Particle swarm optimization,” *IEEE International Conference on Neural Networks*, Vol. 4, pp. 1942-1948, Perth, Australia. 1995.

[10] Y. Liu, Z. Qin, Z. Shi and J. Lu, “Center particle swarm optimization,” *Neurocomputing*, Vol. 70, Issue 4-6, pp. 672-679, January 2007.

[11] P. J. Angeline, “Evolutionary optimization versus particle swarm optimization: philosophy and performance differences,” *Lecture Notes in Computer Science*, Springer, Berlin, Vol. 1447, pp. 601-610, 1998.

[12] Y. Jiang, T. Hu, C. C. Huang and X. Wu, “An improved particle swarm optimization algorithm,” *Applied Mathematics and Computation*, Vol. 193, Issue 1, pp. 231-239, 1 October 2007.

[13] L. Wang, D. Z. Zheng and Q. S. Lin, “Survey on chaotic optimization methods,” *Comput Technol Automat*, Vol. 20, pp. 1-5, 2001.

[14] J. Chuanwen and E. Bompard, “A self-adaptive chaotic particle swarm algorithm for short term hydroelectric system scheduling in deregulated environment,” *Energy Conversion and Management*, Vol. 46, Issue 17, pp. 2689-2696, October 2005.

[15] B. Liu, L. Wang, Y. H. Jin, F. Tang and D. X. Huang, “Improved particle swarm optimization combined with chaos,” *Chaos, Solitons & Fractals*, Vol. 25, Issue 5, pp. 1261-1271, September 2005.

[16] T. Xiang, X. Liao, K.W. Wong, “An improved particle swarm optimization algorithm combined with piecewise linear chaotic map,” *Applied Mathematics and Computation*, Vol. 190, Issue 2, pp. 1637-1645, 15 July 2007.

[17] L. dos Santos Coelho and V. C. Mariani, “A novel chaotic particle swarm optimization approach using Hénon map and implicit filtering local search for economic load dispatch,” Chaos, Solitons & Fractals, In Press, Corrected Proof, Available online 11 May 2007.

[18] R. M. May, “Simple mathematical models with very complicated dynamics,” Nature, Vol. 1976, pp. 261-459.

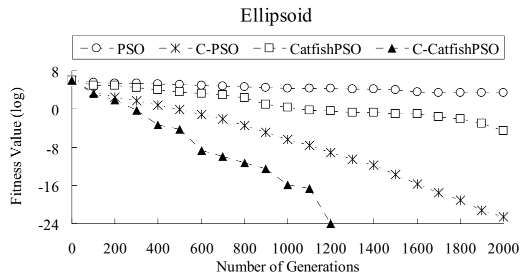
[19] L.Y. Chuang, S. W. Tsai and C. H. Yang, “Catfish Particle Swarm Optimization,” *IEEE Swarm Intelligence Symposium 2008 (SIS 2008)*, St. Louis, Missouri, 21-23 Sep 2008, pp. 1-5.

[20] Y. Shi and R.C. Eberhart, “A modified particle swarm optimizer,” *in Proceedings of IEEE International Conference on Evolutionary Computation*, Anchorage, AK, May 1998, pp. 69-73.

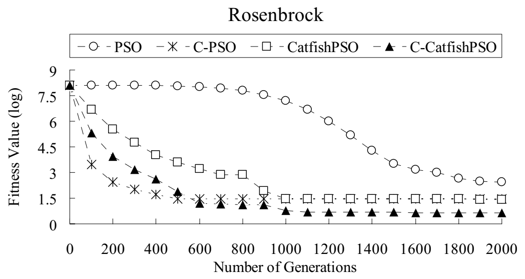
[21] Y. Shi and R.C. Eberhart, “Empirical study of particle swarm optimization,” *in Proceedings of Congress on Evolutionary Computation*, Washington, DC, 1999, pp. 1945-1949.

[22] D. Kuo, “Chaos and its computing paradigm,” *IEEE Potentials Magazine*, Vol. 24, Issue 2, pp. 13-15, April-May 2005.

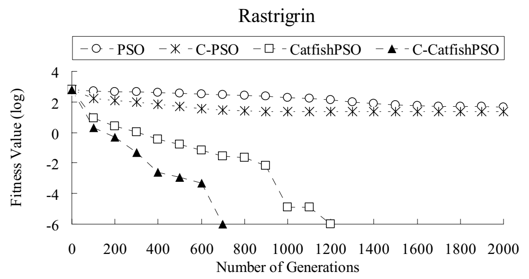
[23] D. E. Knuth. “The Art of Computer Programming,” Vol. 2: Seminumerical Algorithms, Third Edition. Addison-Wesley, 1997. Section 3.2.1: The Linear Congruential Method, pp.10-26.



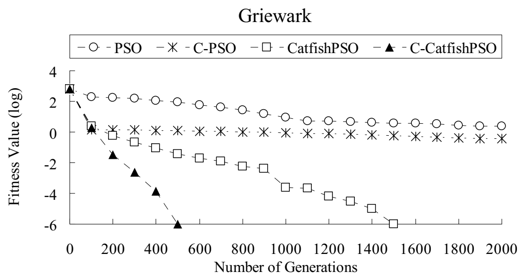
(a) Ellipsoid function



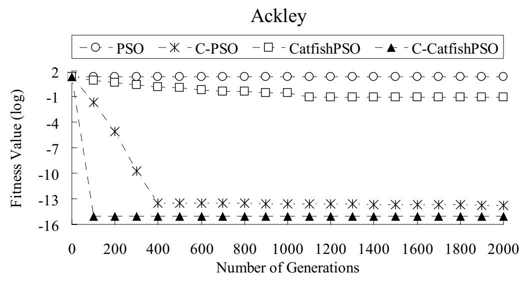
(b) Rosenbrock function



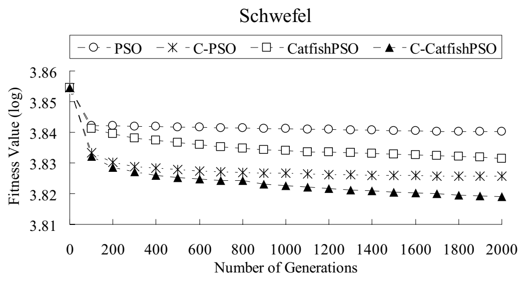
(c) Rastrigrin function



(d) Griewark function



(e) Ackley function



(f) Schwefel function

Figure 2. Six benchmark functions for PSO, C-PSO, CatfishPSO and C-CatfishPSO