Chaotic Particle Swarm Optimization *

Yanxia Sun
F'SATIE, Department of Electrical Engineering, Tshwane University of Technology
Pretoria, South Africa
sunyanxia@gmail.com

Guoyuan Qi
F'SATIE, Department of Electrical Engineering, Tshwane University of Technology
Pretoria, South Africa
guoyuanqi@gmail.com

Zenghui Wang
F'SATIE, Department of Electrical Engineering, Tshwane University of Technology
Pretoria,South Africa
wangzengh@gmail.com

Barend Jacobus van Wyk
F'SATIE, Department of Electrical Engineering, Tshwane University of Technology
Pretoria, South Africa
vanwykb@gmail.com

Yskandar Hamam
F'SATIE, Department of Electrical Engineering, Tshwane University of Technology
Pretoria, South Africa
hamama@tut.ac.za

ABSTRACT

A new particle swarm optimization (PSO) algorithm with has a chaotic neural network structure, is proposed. The structure is similar to the Hopfield neural network with transient chaos, and has an improved ability to search for globally optimal solution and does not suffer from problems of premature convergence. The presented PSO model is discrete-time discrete-state. The bifurcation diagram of a particle shows that it converges to a stable fixed point from a strange attractor, guaranteeing system convergence.

Categories and Subject Descriptors

G.1.6 [**Optimization**]: Global optimization, Nonlinear, programming, Simulated annealing; F.1.1 [**Models of Computation**]: Computability theory, Self-modifying machines

General Terms

Algorithm, Design

*This work was supported by the grants: National Research Foundation of South Africa (No. IFR2008111000017), Tshwane University Research foundation, South Africa, the Natural Science Foundation of China (No. 10772135, 60774088), the Scientific Foundation of Tianjin City, China (No. 07JCYBJC05800) and the Scientific and Technological Research Foundation of Ministry of Education , China (No. 207005).

GEC'09, June 12-14, 2009, Shanghai, China.

Keywords

Chaos, Particle swarm optimization, Neural network, Convergence

1. INTRODUCTION

Particle swarm algorithms have been shown to successfully optimize a wide range of problems [1, 2, 3, 4, 5]. The development of the PSO and its applications was reviewed in Ref. [3]. Ref. [1] proposed a PSO approach to identify the autoregressive moving average with an exogenous variable model for short-term power load forecasting. Ref. [2] presented a modified PSO algorithm for engineering optimization with constraints that algorithm was tested on several engineering design optimization problems. Ref.[4, 5] are two recent books which reviewed particle swarm optimization algorithms and their engineering applications. The PSO algorithm is based on a metaphor of social interaction to search a space by adjusting the trajectories of individual vectors, called "particles" conceptualized as moving points in a multidimensional space. The random weight of the control parameters is used to give a kind of explosion as particle velocities and positional coordinates careen toward infinity [6]. Most algorithm versions have some undesirable dynamical properties [7, 8, 9]. Notably, the particle velocities need to be limited in order to control their trajectories. Clerc and Kennedy [6] have significantly improved the convergence tendencies of particle swarm systems by introducing a constriction coefficient and by resorting to random weights to control the search space of the particle trajectories. PSO was also combined with other algorithms to extend the search space. Typically, Ref. [9] used chaos and PSO in an alternative fashion to avoid getting trapped in local minima.

There are also some other optimizing algorithms such as neural networks [10, 11]. Since Hopfield and Tank [12] applied their neural network to the traveling salesman problem, neural networks have been shown to provide a powerful approach to a wide variety of optimization problems. How-

ever, the Hopfield neural network (HNN) is often trapped in a local minima due to the use of local optimization algorithms such as gradient descent. A number of modifications were made to Hopfield neural networks to escape from the local minima. Typical modifications are based on chaotic neural networks [13] and simulated annealing [14] to solve the global optimization problem [15]. In Refs.[16, 17, 18] the guaranteed convergence of neural networks are discussed.

As PSO was developed to model birds flocking or fish schooling for food [7], the decision of the future position (determined by velocity) can be regarded as particle intelligence. It is well known that although chaos is generated by a deterministic nonlinear system it appears pseudo random. Using a chaotic system to replace the effect of random weights of the original PSO might be convenient for analysis while maintaining stochastic search proper. Most importantly, it might avoid the explosion of particle trajectories caused to by random weights. By combining this idea with neural networks, a particle model with a simple chaotic neural network (CNN) structure is proposed in this paper..

2. PRELIMINARIES

Many optimization problems can be abstracted as the following functional optimization problem:

$$\begin{cases} \mathrm{J}(X_i) = g(X_i) + \sum_{j=1}^{k} \lambda_j F(p_j(X_i)), & X_i = [x_i^1, x_i^2, \ldots, x_i^n]. \\ p_j(X) \geq 0, & j = 1, 2, \ldots, k. \end{cases} \tag{1}$$

Here $g(\cdot)$ is the objective function without constraints; X_i denotes the position vector of particle i consisting of n variables and $p_j(X_i)$ is the jth constraint.

The canonical particle swarm algorithm works by iteratively searching in a region and which is concerned with the best previous success of each particle, the best previous success of the particle swarm and the current position and velocity of each particle [7]. Every candidate solution of $J(X)$ is called a "particle". The particle searches the domain of the problem, according to

$$\begin{aligned} V_i(t+1) &= \omega V_i(t) + c_1 R_1 (P_i - X_i(t)) \\ &\quad + c_2 R_2 (P_g - X_i(t)), \end{aligned} \tag{2}$$

$$X_i(t+1) = X_i(t) + V_i(t+1), \tag{3}$$

where $V_i = [v_i^1, v_i^2, \ldots, v_i^n]$ is called the velocity of particle i; $X_i = [x_i^1, x_i^2, \ldots, x^n]$ represents the position of particle i; P_i represents the best previous position of particle i (indicating the best discoveries or previous experience of particle i); P_g represents the best previous position among all particles (indicating the best discovery or previous experience of the social swarm); ω is called inertia weight that controls the impact of the previous velocity of the particle on its current one and it is sometimes adaptive [9]; R_1 and R_2 are two random weights whose components r_1^j and r_2^j $(j = 1, 2, \ldots, n)$ are chosen uniformly within the interval $[0, 1]$ which result in a kind of explosion of the particle trajectory; c_1 and c_2 are the positive constant parameters. Generally the value of each component in V_i should be clamped to the range $[-v_{max}, v_{max}]$ to control excessive roaming of particles outside the search space.

As every particle can be seen as the model of a single fish or a single bird, the position chosen by the particle can be regarded as a state of a neural network with a random synaptic connection. According to (2)-(3), the position components of particle i can be thought of as the output of a neural network shown in Fig.1.

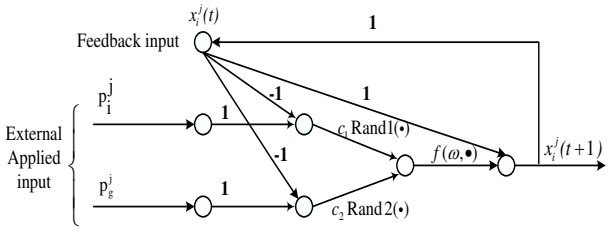

Figure 1: Particle structure.

In Fig.1, Rand1(·) and Rand2(·) are two independently and uniformly distributed random variables with range $[0, 1]$, which refer to r_1^j and r_2^j, respectively. p_i^j and p_g^j are the components of P_i and P_g, respectively. Although p_g^j is the previous best value amongst all particles, p_i^j as externally applied input, is the jth element of the best previous position P_i, and it is coupled with other components of P_i. The particles fly toward a new position according to (2)-(3), and this process is repeated until a defined stopping criterion is met.

REMARK 1. *Fig.1 has three characters: (1) the structure has feedback input that is similar to the Hopfield neural network [12]; (2) the structure has an externally applied input reminiscent of the back-propagation neural network [19]; (3) it exhibits stochastic like chaos [13].*

As pointed out by Clerc and Kennedy [6], the powerful optimization ability of the PSO comes from the interaction amongst the particles as they react to one another. The analysis of complex interaction amongst the particle swarm is beyond the scope of this paper which focuses on the construction of a simple particle in the neural network perspective and the convergence of the PSO.

3. DYNAMICS OF THE SIMPLE PSO

Clerc and Kennedy [6] have reformulated (2) as

$$v_i^j(t+1) = \omega v_i^j(t) + \phi(p_i' - x_i^j), \tag{4}$$

where

$$p_i' = \frac{c_1 r_1^j p_i^j + c_2 r_2^j p_g^j}{c_1 r_1^j + c_2 r_2^j}, \quad \phi = c_1 r_1^j + c_2 r_2^j, \quad (j = 1, 2, \ldots, n).$$

Setting p_i' as a constant value, the system will reduces

$$\begin{cases} v(t+1) = \omega v(t) + \phi(p - x(t)), \\ x(t+1) = x(t) + v(t+1), \end{cases} \tag{5}$$

where ϕ and p are constants.
If we define $y(t) = p - x(t)$, (5) becomes

$$\begin{cases} v(t+1) = \omega v(t) + \phi y(t), \\ y(t+1) = -\omega v(t) + (1 - \phi) y(t). \end{cases} \tag{6}$$

with $\omega = 1$. Eqn. (6) has been analyzed for various values of φ and it has been proven that a non-random particle trajectory is cyclical or quasi-cyclic when $\phi \in (0, 4)$ [6]. The bifurcation of (6) with $\omega = 1$ is shown in Fig. 2. The corresponding Lyapunov exponents λ_1 and λ_2 both are 0 and the initial value of $v(t)$ and $y(t)$ have a great effect on the range of y. For every value of ϕ the long term value of y is shown in Fig. 2 after the initial transients have decayed from the initial conditions $v(0) = 0.1$, $y(0) = 0.01$ for a 100 iterations.

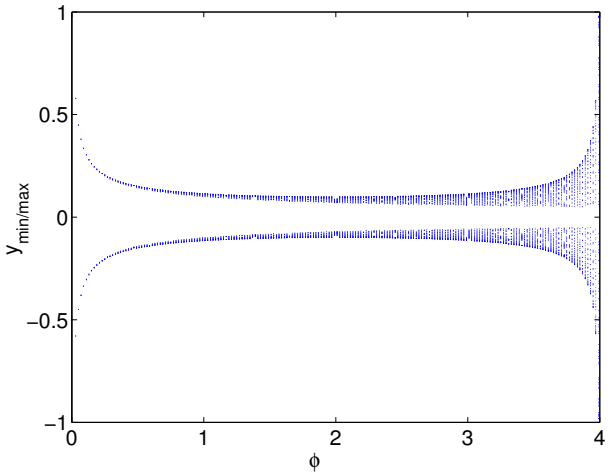

Figure 2: Bifurcation of single particle.

As seen from Fig. 2, its bifurcation becomes pseudo-chaotic as ϕ approach 4. When a random weight is added into (6), its dynamics will even become more complex and difficult to analyze.

4. A MODEL OF THE CHAOTIC PARTICLES

We will now discuss the property of the HNN and incorporate it into the chaotic PSO in the sequel. Artificial neural networks are composed of simple artificial neurons mimicking biological neurons. The HNN has a property that as each neuron in an HNN updates, an energy function is not monotonously reduced until the network stabilizes. One can therefore map an optimization problem to an HNN such that the cost function of the problem corresponds to the energy function of the HNN and the result of the HNN thus suggests a low cost solution to the optimization problem. The HNN might therefore be a good choice to model particle behavior.

As mentioned above, the HNN model of each particle position component has an external input I_i which is similar to p_i^j and p_g^j in Fig. 1, and its output is $x_i^j(t)$. Similar to the cost function of the HNN[20], the proposed PSO energy function is given by

$$J_i^j(t) = A(x_i^j(t) - p_g^j)^2 + B(x_{ip}^j(t) - p_i^j)^2 + C(x_i^j(t) - x_{ip}^j(t))^2, \quad (7)$$

where A, B and C are positive constant, the expression $(x_i^j(t) - p_g^j)^2$ implies that $x_i^j(t)$ has a tendency to shift to the best previous position component of the particle swarm P_i^j. The expressions $(x_{ip}^j(t) - p_i^j)^2$ together with $(x_i^j(t) - x_{ip}^j(t))^2$ imply that $x_i^j(t)$ has a tendency toward the best previous position of particle i. If $x_{ip}^j(t)$ reflects the previous position,the expression $(x_i^j(t) - x_{ip}^j(t))^2$ illustrates that the average velocity of particle i should be slower to save energy.

According to (2)-(3) or Fig. 1, the PSO must use random weighting to simulate birds flocking or fish searching for food. An intelligent particle by implication exhibits chaos-like behavior. Ref. [13] proposed a kind of chaotic neuron, which includes relative refractoriness in the model to simulate chaos in a biology brain. The convergence theorems of (7)have been given in [16] and [18], but the theorems were proposed based on many limitations associated with energy functions. Ref. [17] presented a convergence theorem for the HNN with arbitrary energy functions and discrete-time dynamics for discrete neuronal input-output functions. The theorem is that for a network of neurons with discrete input–output neuronal response functions, for any change of state in any neuron i, the energy is guaranteed to decrease $\triangle J_i^j(t) < 0$, if $f(\cdot)$ is a monotonously increasing function and the network is updated according to the following rules:
(1)The network is updated asynchronously and,
(2)If the following equations have nonzero solutions for $\triangle x_i^j(t)$ and $\triangle u_i^j(t)$,

$$x_i(t) + \triangle x_i(t) = f(u_i(t) + \triangle u_i(t)), \quad (8)$$

$$\triangle u_i(t) = \frac{-w \triangle J_i(t)}{\triangle x_i(t)}, \quad (9)$$

the state of neuron i is updated; it remains unchanged otherwise. Where $u_i(t)$ is the input of the neurons, and $x_i(t)$ is the state of the neurons, where $\omega > 0$ is the updating rate. In this paper we set $\omega = 1$. With (8)and (9) satisfied, the convergence of the energy function given by (7) can be guaranteed.
For simplicity, the neuron input-output function is chosen as a linear saturation function (10) and (13).The dynamics of component j^{th} of particle i can be described as

$$x_i^j(t+1) = \begin{cases} 1 & \text{if } ku_i^j(t+1) > 1 \\ ku_i^j(t+1) & \text{if } ku_i^j(t+1) \in [0,1] \\ 0 & \text{if } ku_i^j(t+1) < 0, \end{cases} \quad (10)$$

$$u_i^j(t+1) = u_i^j(t) + \triangle u_i^j(t+1) - z(t)(x_i^j(t) - I_0), \quad (11)$$

$$\triangle u_i^j(t+1) = \frac{-\Big(2A(x_i^j(t) - p_g^j) + 2C(x_i^j(t) - x_{i_p}^j(t))\Big)}{1 + kA + kC}, \quad (12)$$

$$x_{i_p}^j(t+1) = \begin{cases} 1 & \text{if } ku_{i_p}^j(t+1) > 1 \\ ku_{i_p}^j(t+1) & \text{if } ku_{i_p}^j(t+1) \in [0,1] \\ 0 & \text{if } ku_{i_p}^j(t+1) < 0, \end{cases} \quad (13)$$

$$u_{i_p}^j(t+1) = u_{i_p}^j(t) + \triangle u_{i_p}^j(t+1) - z(t)(x_{i_p}^j(t) - I_0), \quad (14)$$

$$\triangle u_{i_p}^j(t+1) = \frac{-\Big(2B(x_i^j(t) - p_i^j) - 2C(x_i^j(t) - x_{i_p}^j(t))\Big)}{1 + kB + kC}, \quad (15)$$

$$z(t+1) = (1 - \beta)z(t). \quad (16)$$

Here:
$z(t)$ is the refractory strength;
β is the damping factor of the time-dependent $z(t)$, $(0 \leq \beta \leq 1)$;

I_0 is the positive parameter.

By solving (7), (9), (10) and (13) simutanoeusly, we can get (12) and (15). The self-coupling terms $z(t)(x_i^j(t) - I_0)$ and $z(t)(x_{ip}^j(t) - I_0)$ are added to (11) and (14) to cause chaos. Here, the asynchronous updating arrangement is used for the particle dynamic model, implying that the (10)-(11) and (13)-(14) updating times are different. They use the same time indices because particle does not distinguish between asynchronous updating sequences.

As can be seen from (10) and (13), the particle position component x_i^j is located in the interval $[0, 1]$ and therefore the optimization problem variable interval must be mapped to $[0, 1]$ and vice versa using

$$x^j = \frac{x_j - a_j}{b_j - a_j}, \quad j = 1, 2, \cdots, n. \tag{17}$$

and

$$x_j = a_j + x^j(b_j - a_j), \quad j = 1, 2, \cdots, n. \tag{18}$$

Here, a_j and b_j are the lower boundary and the upper boundary of x_j, respectively.

5. DYNAMICS OF CHAOTIC PSO

In this section, the dynamics of the chaotic PSO is analyzed. The first subsection describes the dynamics of the simplest chaotic particle swarm with different parameter values. The second subsection discusses the convergence of the chaotic particle swarm. Constraints are not being considered for simplicity.

5.1 Dynamics of the simplest chaotic particle swarm

In this section, the dynamics of the simplest particle swarm model, which is similar in conception to (5), is analyzed. Eqns. (7) and (10)-(16) are the dynamic model of the single particle with subscript j ignored.
The values of the parameters in (7) and (10)-(16) are set as

$$\begin{aligned} A &= 0.02, B = C = 0.01, \quad \varepsilon = 1, z(t+1) = (1-\beta)z(t), \\ k &= 15, I_0 = 0.2, p_i = 0.5, p_g = 0.5. \end{aligned}$$

All the parameters are fixed except $z(t)$ which is varied. Fig. 3 shows the time evolution of $x(t), z(t)$ and the Lyapunov exponent λ of $x(t)$. Here, λ is defined as

$$\lambda = \lim_{n \to +\infty} \frac{1}{m} \sum_{t=0}^{m-1} ln|\frac{dx_i(t+1)}{dx_i(t)}|. \tag{19}$$

The asynchronous updating arrangement should be changed to synchronous updating to calculate the Lyapunov exponent. The convergence process of the simple particle position is described by the nonlinear bifurcation which makes the particle converge to a stable fixed point from a strange attractor. In the following section, it is shown that the fixed point is determined by the best previous position P_g among all particles and the best position P_i of the individual particle .

REMARK 2. *The model is a deterministic Chaos-Hopfield neural network swarm which is different from existing ones with stochastic parameters. Its search orbits show an evolution process of inverse period bifurcation from chaos to periodic orbits then to sink. As chaos is ergodic and the particle is always in chaotic state at the beginning (for example, in Fig.3), it can escape from the trap when the particle undergoes the suboptimal state. This proposed PSO model will therefore not suffer from premature convergence problems.*

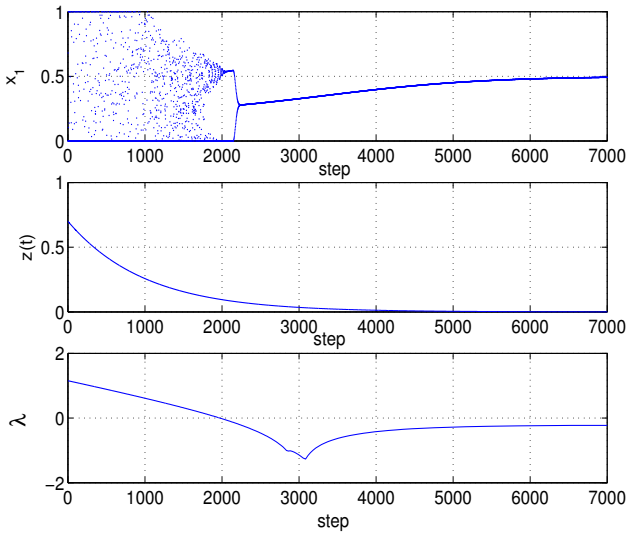

Figure 3: Time evolutions of $x(t), z(t)$ and the spectrum of Lyapunov exponent λ.

5.2 Convergence of the particle swarm

According to (7), the interaction among particles in a swarm is derived from the best previous position amongst all particles. From (7) and (10)-(16), if $t \to \infty$, $z(t)$ decreases linearly to 0. It would be easy to analyze (10)-(16) after $z(t) = 0$ is fixed. There is only one equilibrium, i.e.

$$x_{ie}^j = \frac{(AB + AC)p_g^j + BCp_i^j}{AB + BC + AC}, \tag{20}$$

$$x_{ipe}^j = \frac{(AB + BC)p_i^j + ACp_g^j}{AB + BC + AC}. \tag{21}$$

It is easy to show that the particle model (7)and (10)-(16) has only one equilibrium as $t \to \infty$, i.e., $z(t) = 0$. Hence, as $t \to \infty$, X_i belongs to the sphere whose origin is P_g and the radius is $\parallel \max(X_{rie}) \parallel$ where the components x_{rie}^k of X_{rie} are $\max(x_{ie}^j)$. For the above simplest chaotic particle swarm and the parameters, the particle position approach 0.5 as seen in Fig. 3(a). As can be seen from (20) and (21), $x_{ie}^j \approx p_g^j$ if $A \gg B, A \gg C$ or $(AB + AC) \gg C$.

6. NUMERICAL SIMULATION

To test the performance of the proposed algorithms,two famous benchmark optimization problems [21] are used.

6.1 Rastrigin Function

To demonstrate the efficiency of the proposed technique, one famous function Rastrigin function is chosen as the optimization problem. The Rastrigin function with two vari-

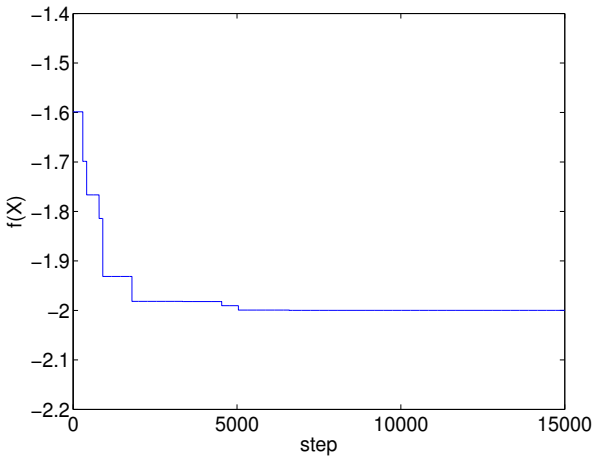

Figure 4: Time evolutions of Rastrigin function

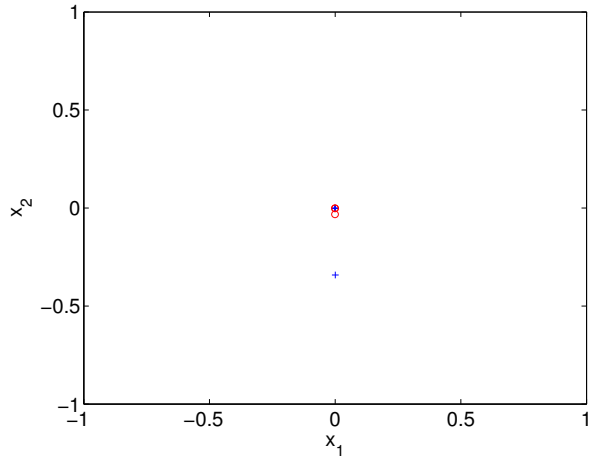

Figure 5: The final convergent particle states achieved from the proposed Chaotic PSO for Rastrigin function.

ables is described as:

$$f(X) = x_1^2 + x_2^2 - cos(18x_1) - cos(18x_2), \quad -1 < x_i < 1, i = 1, 2. \tag{22}$$

The global minimum is equal to -2 and the minimum point is (0, 0). There are about 20 local minima arranged in a lattice configuration.

The proposed technique is applied with a population size of 20 and the chaotic particle swarm parameters are set as follows

$A = 0.02, B = C = 0.01, \beta = 0.001, z(0) = 0.7,$

$z(t+1) = (1-\beta)z(t), k = 30 - \frac{15(z(0)-z(t))}{z(0)}, I_0 = 0.2.$

The position of every particle is initialized with random value. The cost of the Rastrigin function is shown in Fig. 4 with time evolution. The global minimum at -2 is obtained by the best particle with $(x_1, x_2) = (0, 0)$.

From Fig. 4, we can see the proposed method gives good optimization result. Because there are two variables in the Rastrigin function, the final convergent particle states are shown in the plane as Fig. 5. In Fig. 5, the '$*$' at $(0, 0)$ is the best experience of all particles, and the other '$*$' ones are

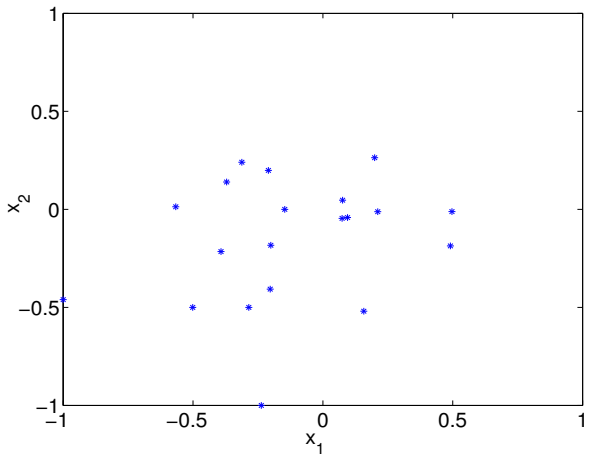

Figure 6: The final particle states achieved from the original PSO for Rastrigin function.

the best experience of each particle; the '$\circ$' ones are the final states of the particles corresponding to the '$*$' ones in Fig.6. When the original PSO [7] is used to optimize Rastrigin function, the global minimum -2 can not be obtained by the best particle with $(x_1, x_2) = (0, 0)$ as seen in Fig. 6. In this numerical simulation, the particle swarm population size is also 20 and their parameters $c_1$and c_2 are both set to 2. The final particle states are shown in Fig. 6. Compared the results obtained from the proposed chaotic PSO in Fig. 5 and the original PSO in Fig. 6, it can be obviously found that the particles of the proposed chaotic PSO are attracted to the best experience of all the particles finally and the convergence is better the original one. The chaotic PSO can guarantee the convergence of the particles swarm but the states of the original particle swarm are ruleless.

6.2 Goldstein Price Function

Another famous benchmark optimization problem is GP-Goldstein-Price:

$$\begin{aligned} f(x) \quad = \quad & (1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 \\ & + 6x_1x_2 + 3x_2^2))(30 + (2x_1 - 3x_2)^2 (18 - 32x_1 \\ & + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)). \end{aligned} \tag{23}$$

The global minimum is equal to 3 and the minimum point is (0, -1). There are about 4 local minima arranged in a lattice configuration.

The population size and the chaotic PSO parameters are same to the Rastrigin function. The position of every particle is initialized with random value. The global minimum at 3.0540 is obtained by the best particle with $(x_1, x_2) = (-0.0154, -1.0035)$. From Fig. 7, we can see the proposed method gives good optimization result. Because there are two variables in the Goldstein Price function, the final convergent particle states are shown in the plane as Fig. 7, it can be obviously found that the particles of the proposed chaotic PSO are attracted to the best experience of all the particles finally and the convergence is good. In Fig. 7, the '$*$' at $(0, -1)$ is the best experience of all particles; the '$\circ$' ones are the final states of the particles corresponding to the '$*$' ones in Fig.8. When the original PSO [7] is used

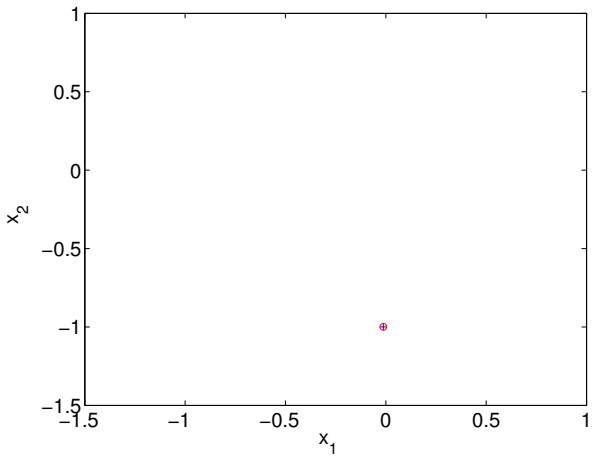

Figure 7: The final convergent particle states achieved from the proposed Chaotic PSO for Goldstein Price function.

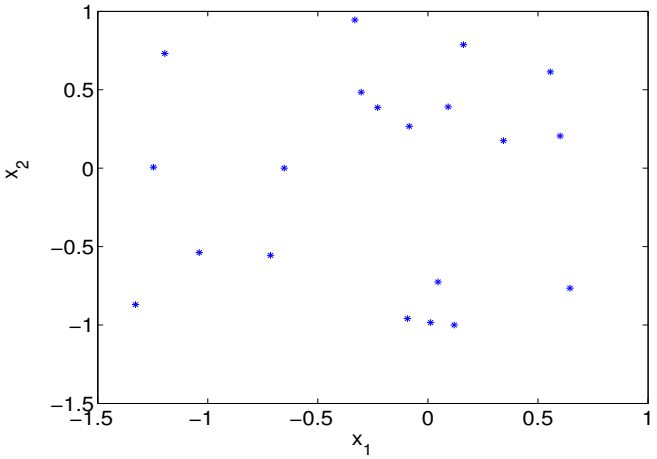

Figure 8: The final convergent particle states achieved from original PSO for Goldstein Price function.

to optimize Goldstein Price function, the global minimum 3 can not be obtained by the best particle as seen in Fig. 8. In this numerical simulation, the particle swarm population size is also 20 and their parameters $c_1$and c_2 are both set to 2. The final particle states are shown in Fig. 8.

Compared the results obtained from the proposed chaotic PSO in Fig. 7 and the original PSO in Fig. 8, it can be obviously found that the particles of the proposed chaotic PSO are attracted to the best experience of all the particles finally and the convergence is better the original one. The chaotic PSO can guarantee the convergence of the particles swarm but the states of the original particle swarm are ruleless.

7. CONCLUSION

This paper proposed a particle model without random weights for the PSO. It was shown that using a chaotic system to determine the weights of the PSO guaranteed convergence. As the model is derived from an HNN, it more naturally describes the character of particles. An additional issue is that the convergent range can be chosen based on the knowledge of the convergence. As this is a general particle model, many techniques, which have been proposed for the original PSO, can be used together with the new model. This will be explored in future work.

8. ACKNOWLEDGEMENTS

The authors thank the reviewers for many helpful comments and suggestions.

9. REFERENCES

[1] C. M. Huang, C. J. Huang and M. L. Wang, A Particle Swarm Optimization to Identifying the ARMAX Model for Short-Term Load Forecasting. IEEE Tran. on Power Systems, 20(2): 1126–1133, May 2005.

[2] X. H. Hu, R. C. Eberhart, Y. H. Shi, Engineering optimization. with particle swarm. Proc. of the IEEE Swarm Intelligence Symposium, pages 53–57, 2003.

[3] X. H. Hu, Y. H. Shi, R. Eberhart, Recent advances in particle swarm. Congress on Evolutionary Computation, pages 90–97, 2004.

[4] M. Clerc, Particle Swarm Optimization, ISTE Publishing Company, 2006.

[5] N. Nedjah, L. D. M. Mourelle, Systems Engineering Using Particle Swarm Optimization, Nova Science Publishers, 2007.

[6] M. Clerc and J. Kennedy, The particle Swarm: explosion, stability, and convergence in multidimension complex space. IEEE Trans Evol. Compute, 6(1):58–73,2002.

[7] J. Kennedy, R. Eberhart, J. Kennedy, R. Eberhart, Particle Swarm Optimization. in: Proc. IEEE Int. Conf. Neural Networks, Perth, Australia, pages 1942–1948, November. 1995 .

[8] S. L. Ho, S. Y. Yang, G. Z. Ni, E. W. C. Lo, H. C. Wong, A particle swarm optimization-based method for multiobjective design optimizations . IEEE Trans. Magn., 41:1756–1759, 2005.

[9] B. Liu, L. Wang, Y. H. Jin, F. Tang, D. X. Huang, Improved particle swarm optimization combined with chaos. Chaos, Solitons and Fractals, 25:1261–1271, september 2005.

[10] Z .Wang,Y.Liu, K.Fraser, X.Liu, Stochastic stability of uncertain Hopfield neural networks with discrete and distributed delays. Phys. Lett. A, 354(4):288–297, June 2006.

[11] T.Tanaka, E.Hiura, Computational abilities of a chaotic neural network. Phys. Lett. A, 315(3-4):225–230, August 2003.

[12] J. J. Hopfield, D. W. Tank, Neural network of decisions in optimization problems. Biol. Cybern., 52(3):141-152, July 1985.

[13] K. Aihara, T. Takabe and M. Toyoda, Chaotic neural networks. Phys. Lett A, 144(6-7):333–340, March 1990.

[14] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Optimization by simulated annealing. Recchi, Science, 220:671–680, 1983.

[15] L. Chen and K. Aihara, Chaotic simulated annealing by a neural networks model with transient chaos. Neural Networks, 8(6):915–930, 1995.

[16] L. Chen, K.Aihara, Global searching ability of chaotic neural networks. Physica D, 104(3):286–325, 1997.

[17] L. Wang, On Competitive Learning. IEEE Trans. Neural Networks, 8:445–447, 1997.

[18] L. Wang and K. Smith, .On chaotic simulated annealing. IEEE Trans Neural Networks, 9 (4): 716–718, 1998.

[19] D. E. Rumelhart, G. E. Hinton and R. J. Williams, Nature, 1986, 323: 533–536.

[20] J. J. Hopfield, Neurons with graded response have collective computational properties like two-state neurons. Proc. Natl. Acad. Sci. USA, 81: 3088-3092, 1984 .

[21] Wang L, Intelligent optimization algorithms with applications, Beijing: Tsinghua University and Springer Press, 2001.