Engineering Optimization
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713641621

Fully connected particle swarm optimizer

Y. Sun[ab]; K. Djouani[ab]; G. Qi[a]; B. J. van Wyk[a]; Z. Wang[c]
[a] French South African Institute of Technology, Tshwane University of Technology, Pretoria, South Africa [b] Laboratoire Images, Signaux et Systèmes Intelligents, LiSSi, E.A. 3956, University Paris-EST, Paris, France [c] Department of Electrical and Mining Engineering, University of South Africa, Florida, South Africa

First published on: 02 February 2011

To cite this Article Sun, Y. , Djouani, K. , Qi, G. , van Wyk, B. J. and Wang, Z.(2011) 'Fully connected particle swarm optimizer', Engineering Optimization, 43: 7, 801 — 812, First published on: 02 February 2011 (iFirst)
To link to this Article: DOI: 10.1080/0305215X.2010.521242
URL: http://dx.doi.org/10.1080/0305215X.2010.521242

Engineering Optimization
Vol. 43, No. 7, July 2011, 801–812

Taylor & Francis
Taylor & Francis Group

Fully connected particle swarm optimizer

Y. Sun[a,b]*, K. Djouani[a,b], G. Qi[a], B.J. van Wyk[a] and Z. Wang[c]

[a]*French South African Institute of Technology, Tshwane University of Technology, Pretoria 0001, South Africa;* [b]*Laboratoire Images, Signaux et Systèmes Intelligents, LiSSi, E.A. 3956, University Paris-EST, Paris 94010, France;* [c]*Department of Electrical and Mining Engineering, University of South Africa, Florida 1710, South Africa*

(*Received 30 April 2010; final version received 3 August 2010*)

In this article, a new model for particle swarm optimization (PSO) is proposed. In this model, each particle's behaviour is influenced by the best experience among its neighbours, its own best experience and all its components. The influence among different components of particles is implemented by the online training of a multi-input single-output back propagation (BP) neural network. The inputs and outputs of the BP neural network are the particle position and its tendency to the best position, respectively. Therefore, the new structured PSO model is called a fully connected particle swarm optimizer (FCPSO). Simulation results and comparisons with exiting PSOs demonstrate that the proposed FCPSO effectively enhances the search efficiency and improves the search quality.

Keywords: particle swarm optimization; BP network; fully connected

1. Introduction

The PSO, first introduced by Kennedy *et al*. (1995), is a stochastic optimization technique that can be likened to the behaviour of a flock of birds or the sociological behaviour of a group of people. It has been used to solve a range of optimization problems, including neural network training (Van den Bergh *et al*. 2000) and function minimization (Shi *et al*. 1998). In a particle swarm optimizer the individuals are evolved by cooperation among the individuals through generations. Every particle finds its personal best position and the group's best position through iteration, and then modifies their progressing direction and speed to reach the optimum position rapidly. Particle swarm algorithms have been used for successfully optimizing a wide range of problems (Rumelhart *et al*. 1986, Hamidian *et al*. 2010, Kotinis 2010). A successful application of a real value model of PSO to time history optimization was reported in Gholizadeh *et al*. (2009). Perez *et al*. (2007) proposed a particle swarm approach for different structural optimization tasks.

During the search process, each particle can be regarded as an independent agent, which searches the problem space based on its own experience and the experiences of its peers. The former is the cognitive part of the particle update formula, while the latter is the social part of the particle

*Corresponding author. Email: sunyanxia@gmail.com

ISSN 0305-215X print/ISSN 1029-0273 online

DOI: 10.1080/0305215X.2010.521242
http://www.informaworld.com

update formula. Both play crucial roles to guide the particle's search. However, there are no connections among particle components. Relationships between the variables of optimization problems are used by many optimization algorithms to obtain good optimization performance. There are many classic optimization algorithms in the literature such as quasi-Newton methods (Fletcher *et al*. 1963), the Gauss–Newton method, the Levenberg–Marquardt method (More 1978), and so on. In these optimization algorithms, the search direction is determined by all variables of the optimization function. Many intelligent optimization methods are proposed based on all components of the optimization problem, such as Hopfield neural networks (Hertz *et al*. 1991, Hopfield 2007).

This article proposes a new fully connected particle swarm optimizer (FCPSO). Each particle's behaviour is influenced by the best experience among its neighbours, its own best experience and all its components. The influence of different components of a particle is realized by a back propagation neural network. The rest of this article is arranged as follows: Section 2 introduces particle swarm optimization. In Section 3, the basic concepts of back propagation neural networks are described. The FCPSO algorithm is described in Section 4. Section 5 describes the problems used to evaluate the new algorithm and the results obtained. Finally, the concluding remarks appear in Section 6.

2. Particle swarm optimization

Many optimization problems can be represented as the following optimization problem:

$$\begin{aligned} &\text{Find}: X^{\mathrm{T}} = [x_1, x_2, \ldots, x_n], \\ &\qquad \text{Minimize}: f(X), \\ &\qquad \text{subject to}: g_j(X) \leq 0,\ j = 1, 2, \ldots, k. \\ &\qquad\qquad X_1 \leq X \leq X_{\mathrm{u}}. \end{aligned} \tag{1}$$

Here $f(\cdot)$ is the objective function without constraints; $X(t)$ denotes the position vector consisting of n variables and $g_j(X)$ is the jth constraint.

The canonical particle swarm algorithm works by iteratively searching in a region and 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 (Kennedy *et al*. 1995). Every candidate solution is called a 'particle'. A particle status on the search space is characterized by two factors: its position and velocity, which are updated by the following equations:

$$V_i(t+1) = \omega V_i(t) + c_1 R_1 (P_i - X_i(t)) + c_2 R_2 \left(P_g - X_i(t)\right), \tag{2}$$

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

where $V_i = [v_i^1, v_i^2, \ldots, v_i^n]$ is the velocity vector of particle i; $X_i = [x_i^1, x_i^2, \ldots, x_i^n]$ 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 the inertia weight that controls the impact of the previous velocity of the particle on its current velocity and is sometimes adaptive; 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 might not guarantee the convergence of the particle trajectory; c_1 and c_2 are 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. In order to control the 'explosion' of the swarm,

Clerc *et al*. (2002) introduced a constriction coefficient which in the simplest case is called 'Type 1' (χ). In general, when several particles are considered in a multidimensional problem space, Clerc's method leads to the following update rule:

$$V_i(t+1) = \chi \left\{ V_i(t) + \varphi_1 rand_1 \left(P_i - X_i(t)\right) + \varphi_2 rand_2 \left(P_g - X_i(t)\right) \right\},$$
$$X_i(t+1) = X_i(t) + V_i(t+1), \tag{4}$$

where

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \quad \varphi_1 + \varphi_2 = \varphi > 4. \tag{5}$$

Typically, when this method is used, φ is set to 4.1 and the constant χ is thus 0.729. In general, the constriction factor improves the convergence of the particle over time by damping the oscillations once the particle is focused on the best point in an optimal region. The main disadvantage of this method is that the particles may follow wider cycles and may not converge when the individual best performance, P_i, is far from the neighbourhood's best performance, P_g (two different regions) (del Valle *et al*. 2008).

3. Back propagation neural networks

One of the most commonly used supervised ANN models is the back-propagation network that uses the back-propagation learning algorithm (Rojas *et al*. 1996). It was first described by Paul Werbos in 1974, but it wasn't until 1986, through the work of David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams (Rumelhart *et al*. 1986), that it gained recognition, and it led to a 'renaissance' in the field of artificial neural network research.

Back propagation networks are generally multi-layer perceptions (usually with one input, one hidden, and one output layer). To make sure that the hidden layers serve any useful function, multi-layer networks must have nonlinear activation functions: a multi-layer network using only linear activation functions is equivalent to a single layer, linear network. Nonlinear activation functions that are commonly used include the logistic function, the soft max function, the gaussian function, and so on.

In general, the back propagation algorithm looks for the minimum of the error function in weight space using the method of gradient descent. Since this method requires computation of the gradient of the error function at each iteration step, the continuity and differentiability of the error function must be guaranteed. Obviously a kind of activation function other than the step function should be used. The back propagation neural network is essentially a network of simple processing elements working together to produce a complex output. The combination of weights which minimizes the error function is considered to be a solution of the learning problem.

In this article, the back propagation algorithm is applied to predict the tendency of the best position of the particle swarm. The 'Levenberg–Marquardt method' (More 1978) is chosen to train back propagation neural networks.

4. A new fully connected particle swarm

Referring to Equation (2), the right side of which consists of three parts: the first part is the previous velocity of the particle; the second and third parts contribute to the change of the velocity of a particle. As explained in Shi *et al*. (1998), without the second and third parts, the particles will keep on 'flying' at the current speed in the same direction until they hit the boundary. PSO will

not find an acceptable solution unless there are acceptable solutions on their 'flying' trajectories, but that is a rare case. On the other hand, without the first part of Equation (2), the flying particles' velocities are only determined by their current positions and their historical best positions. The velocity itself is memoryless. If it is assumed from the beginning that particle i has the best global position, then particle i will be 'flying' at velocity 0, that is, it will keep still until another particle takes over the global best position. At the same time, each other particle will be 'flying' toward the weighted centroid of its own best position and the global best position of the population (Shi *et al*. 1998).

There are some authors who suggested adjustments to the parameters of the PSO algorithm: adding a random component to the inertia weight (del Valle *et al*. 2005, Mohagheghi *et al*. 2005), applying fuzzy logic (Shi *et al*. 2001), using a secondary PSO to find the optimal parameters of a primary PSO (Doctor *et al*. 2004), and adaptive critics (Venayagamoorthy 2004).

As can be seen from Equations (2) and (3), the position of a particle, the best experience among its neighbours and its own best experience are connected by fixed variables and random parameters. There exists a linear relationship between these elements. 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 Equations (2)–(3), the position components of particle i can be thought of as the output of a neural network as shown in Figure 1. However, the relationship and influence only relies on the corresponding dimensional components of the particle swarm which is easily seen in Figure 1. For example, the component $x_i^1(t+1)$ is only influenced by $x_i^1(t)$, $v_i^1(t)$, $p_i^1(t)$ and $p_g^1(t)$, but does not directly get information from the other components $x_i^2(t), x_i^3(t), \ldots, x_i^n(t)$, $v_i^2(t), v_i^3(t), \ldots, v_i^n(t)$, $p_i^2(t), p_i^3(t), \ldots, p_i^n(t)$ and $p_g^2(t), p_g^3(t), \ldots, p_g^n(t)$. To show the relationship among different components, neural networks can be used to model the characteristics of the optimization problem. The inputs and outputs of the neural network are the positions of the particles and the tendency of the best experience of

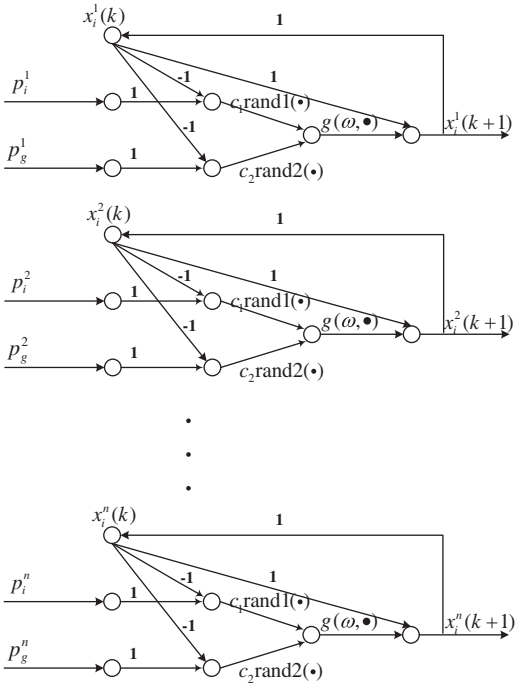

Figure 1. The structure of the classical PSO.

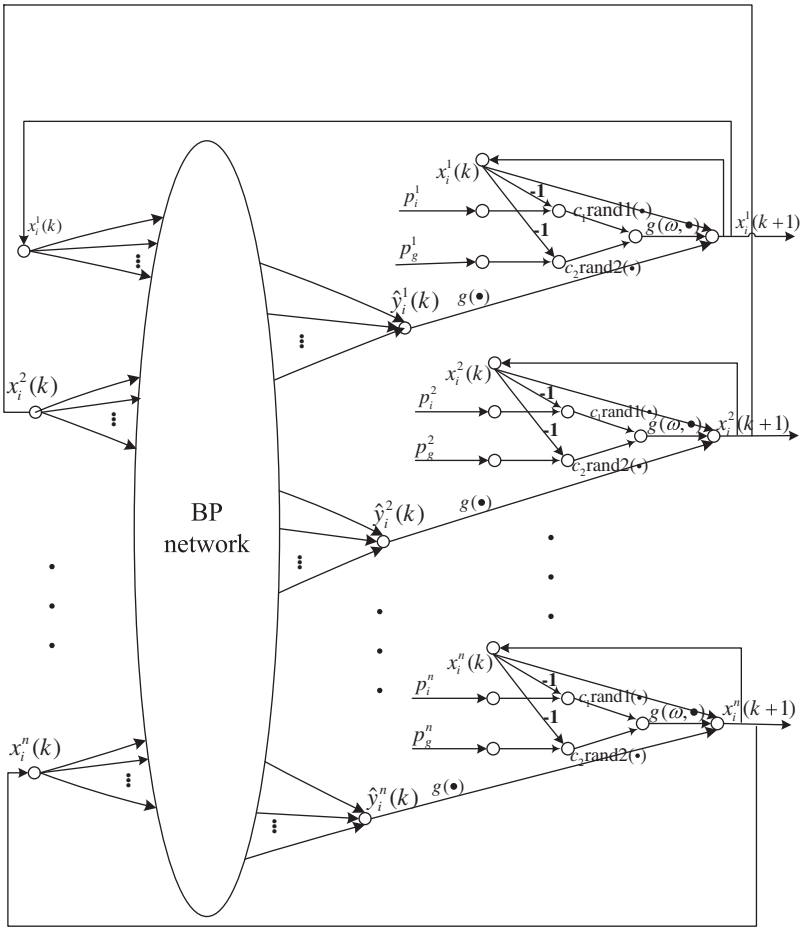

Figure 2. The structure of the fully connected PSO based on the multi-input multi-output BP neural network.

the particles, respectively. Here the back propagation neural network is used to learn this relationship. The fully connected particle swarm structure based on the multi-input multi-output BP neural network is therefore proposed, and is shown in Figure 2. To reduce the complexity of the proposed structure, multi-input single-output back propagation neural networks are used, as shown in Figure 3. In this structure, all the components of each particle are the inputs of the back propagation neural networks. The output of the back propagation neural network, $\hat{y}$, reflects the tendency of the best experience of the particle. In order to take advantage of each component of the particle itself, an improvement index $\Delta X_i(t)$ is added to Equation (3). In the authors' proposed fully connected particle swarm optimization, when p_g or p_i is updated, a back propagation neural network is used to get the improvement index $\Delta X_i(t)$. The inputs of the training neural network are each component of the particle's position, $X_i(t)$. The output of the training neural network is

$$y_i^j = \frac{f\left(X_i^j(t_0)\right) - f\left(X_i^j(t_1)\right)}{X_i^j(t_0) - X_i^j(t_1)}, \tag{6}$$

where $f(X_i^j(t_0))$ is the fitness function when p_g or p_i is updated; $f(X_i^j(t_1))$ is the fitness function corresponding to time t_1 of the particle swarm; $X_i^j(t_0)$ is the position of the particle corresponding

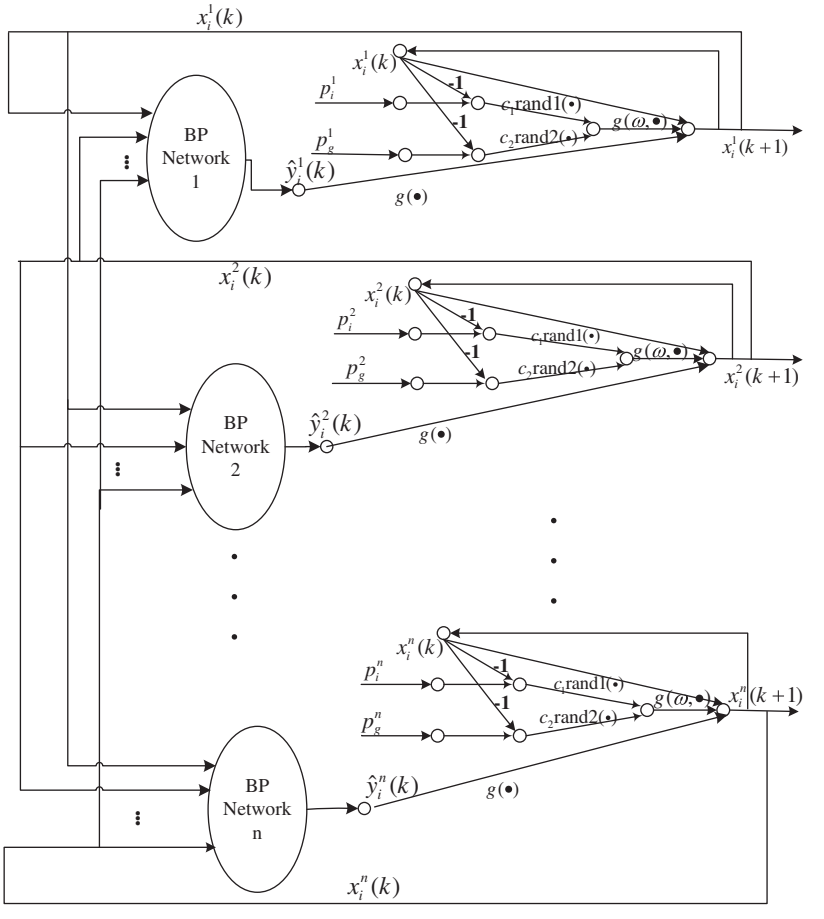

Figure 3. The structure of the fully connected PSO based on the multi-input single-output BP neural network.

to the updated p_g^j or p_i^j of particle i; and $X_i^j(t_1)$ is the position of the particle at time t_1. For the improvement index $\Delta X_i^j(t)$, a random part is introduced to improve the searchability. Based on trial and error, the parameter α can be chosen in the range [0.01, 0.2]; in this article, 0.01 is used. When the particles get trapped into local minima, the random part helps the particles to escape from the local minima. The jth component of particle i is described as

$$\Delta X_i^j(t) = g(y_i^j) = \frac{\alpha * rand1(\hat{y}_i^j)}{generations}. \tag{7}$$

Equations (2) and (3) are therefore changed into the following equations:

$$V_i(t+1) = \omega V_i(t) + c_1 R_1(P_i - X_i(t)) + c_2 R_2(P_g - X_i(t)), \tag{8}$$

$$X_i(t+1) = X_i(t) + V_i(t+1) + \Delta X_i(t). \tag{9}$$

The following procedure can be used for implementing the proposed particle swarm algorithm.

(1) Initialize the BP neural network, the swarm and assign a random position in the problem hyperspace to each particle and calculate the fitness function which is given by the optimization problem whose variables correspond to the elements of the particle position coordinates.
(2) Synchronously update the positions of all the particles according to Equations (5) and (6), and change the two states every iteration.
(3) Evaluate the fitness function for each particle.
(4) For each individual particle, compare the particle's fitness value with its previous best fitness value. If the current value is better than the previous best value, then set this value as the p_i^j, and the particle's position related to the previous value and Equation (3) is used to train the BP neural network. (The data from the first 150 iterations is used to train the neural network using 20 epochs before it is switched into the loop where it is trained online during each iteration using five epochs.)
(5) Repeat steps (2)–(4) until a stopping criterion is met (*e.g.* a maximum number of iterations or a sufficiently good fitness value).

5. Numerical simulation

To test the performance of the proposed algorithm, four optimization problems were used. These four optimization problems are standard test functions for minimization methods and each of them has several local minima. In the numerical simulation of the proposed particle swarm algorithm, the particle swarm population size is 20 and the rest of the parameters are as follows: inertia weight $\omega = 0.5 + (rand/2)$; learning rates $c_1 = c_2 = 1.5 + rand$ and maximum velocity $V_{\max}$ set to the dynamic range of the particle in each dimension (Toscano *et al*. 2004). The total number of iterations is 500. The neural network is switched into the loop after 150 iterations. It is initially trained online using 20 epochs, but only five epochs are used for the remaining iterations. The training epochs in the first iterations and in the remanent iteration of the back propagation neural network are set as 20 and five, respectively, and the other parameters are set to the defaults.

5.1. *The Schaffer f6 function*

To demonstrate the efficiency of the proposed FCPSO, the famous Schaffer f6 function was chosen as a test problem. This function with two variables is given by

$$f(X) = 0.5 + \frac{\sqrt{\left(\sin\left(x_1^2 + x_2^2\right)^2\right} - 0.5}{1.0 + 0.001\left(x_1^2 + x_2^2\right)^2}. \tag{10}$$

This function has a single global optimum at $X = (0.0, 0.0)$ and $f_{\min}(X) = 0.0$, with a large number of local optima. The global optimum is difficult to find because the value at the best local optimum differs by only about 10^{-3} from the global optimum. The local optima crowd around the global optimum.

Fix the generations to 500 and the total runs to 10. When the total runs are 10, the best solution of the proposed FCPSO is $f(X) = 6.9672 \times 10^{-9}$ at $X = (0.0435 \times 10^{-4}, -0.2603 \times 10^{-4})$. Figure 4 shows the convergence history of the FCPSO and the constriction factor PSO algorithms (Clerc *et al*. 2002) when solving the Schaffer f6 function. The solid line is the result from the FCPSO. Almost the same result can be obtained from the constriction factor PSO indicated by the dotted line. The dashed line is the best from the constriction factor PSO. The dotted line is the best results from the original PSO (Kennedy *et al*. 1995). When the constriction factor PSO

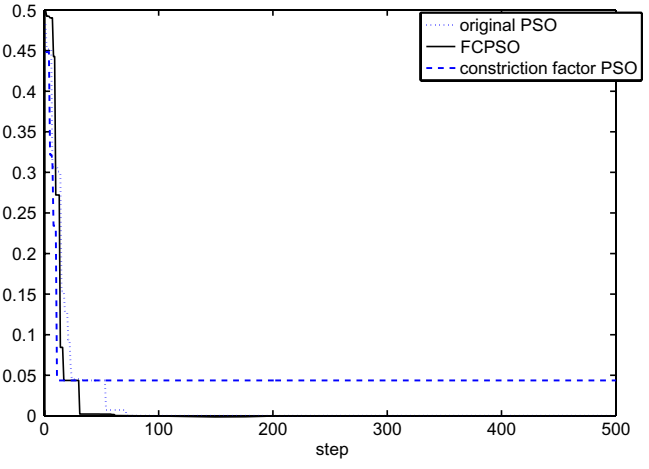

Figure 4. Time evolution of Schaffer f6 functions.

is used to optimize the Schaffer f6 function, the best solution $f(X) = 0.0$ can be obtained by the best particle at $X = (0.0, 0.0)$. Although the best solution obtained by the constriction factor PSO is sometimes better than the FCPSO solution, the constriction factor PSO results are generally much worse – as indicated by the mean, standard deviation and worst results in Table 1.

Table 1. Results using the FCPSO, the original PSO and the constriction factor PSO.

Problem	Method	Best	Mean	Std.dev	Worst	Computation time
Schafferf6	Original PSO	0.0	0.0091	0.0193	0.0474	0.1791
Schafferf6	Constriction factor PSO	0.0	0.0087	0.01840	0.0087	0.1613
Schafferf6	FCPSO	6.9672×10^{-10}	7.3811×10^{-9}	9.2388×10^{-9}	2.9886×10^{-8}	9.7515
Goldstein	Original PSO	3.0037	3.0423	0.0646	3.2129	0.0827
Goldstein	Constriction factor PSO	3.0	3.0	0.0	3.0	0.0613
Goldstein	FCPSO	3.0	3.0	0.0	3.0	14.5360
Hartmann ($n = 3$)	Original PSO	−3.8522	−3.8562	0.0035	−3.8522	0.0723
Hartmann ($n = 3$)	Constriction factor PSO	−3.8628	−3.8628	0.0	−3.8628	0.0643760
Hartmann ($n = 3$)	FCPSO	−3.8628	−3.8628	0.0	−3.8628	14.6641
Hartmann ($n = 3$)	Original PSO	−3.1607	−2.9907	0.1377	−2.7543	0.0833
Hartmann ($n = 3$)	Constriction factor PSO	−3.3257	−3.2768	0.0631	−3.2035	0.0761
Hartmann ($n = 3$)	FCPSO	−3.3257	−3.3253	7.8429×10^{-4}	−3.3231	42.1032
Shekel ($q = 5$)	Original PSO	−4.4485	−3.1046	1.1690	−0.8750	0.0726
Shekel ($q = 5$)	Constriction factor PSO	−10.1532	−6.6494	3.7646	−2.6305	0.0645
Shekel ($q = 5$)	FCPSO	−10.1532	−10.1494	0.0067	−10.1315	25.2875
Shekel ($q = 7$)	Original PSO	−4.5930	−2.8736	0.8463	−1.8398	0.0780
Shekel ($q = 7$)	Constriction factor PSO	−10.4029	−5.5254	3.4959	−2.7519	0.0712
Shekel ($q = 7$)	FCPSO	−10.4029	−10.3860	0.0378	−10.2799	26.5765
Shekel ($q = 10$)	Original PSO	−3.5928	−2.5214	0.4569	−2.0131	0.0894
Shekel ($q = 10$)	Constriction factor PSO	−10.5364	−7.6499	3.7960	−2.4273	0.0805
Shekel ($q = 10$)	FCPSO	−10.5364	−10.5351	0.0016	−10.5324	28.4438

5.2. *The Goldstein function*

The Goldstein function when $X \in R^2$ is given by

$$f(X) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_2^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right], \quad (11)$$

where

$$S = \{X \in R^2 | -2 \leq x_i \leq 2, i = 1, 2\}$$

and S is the range of the parameter x_i. It is an eighth-order polynomial in two variables which is well behaved near each of its four minima. The best solution of the Goldstein function is 3.0 at $X = (0.0, -1.0)$ (Al-Sultan *et al*. 1997). The remaining three local minima are: $f(1.2, 0.8) = 840$, $f(1.8, 0.2) = 84$ and $f(-0.6, 0.4) = 30$. The best result from the proposed FCPSO for the Goldstein test function is 3.0 at $X = (0.0, -1.0)$. The result from CPSO (Liu *et al*. 2005), the original PSO and GA are $-3.0 \pm 5.025\text{e} - 15$, 4.6202 ± 1.4554 and 3.1471 ± 0.9860, respectively. By comparing the results of the FCPSO, the original PSO and GA, it is clear that the FCPSO shows superior performance.

5.3. *The Hartmann function*

The Hartmann function when $n = 3, 6$; $q = 4$ is given by

$$f(X) = -\sum_{i=1}^{q} c_i \exp\left[-\sum_{j=1}^{n} a_{ij}(x_j - p_{ij})^2\right], \quad (12)$$

where

$$S = \{X \in R^n | \, 0 \leq x_j \leq 1, j = 1, 2, \ldots, n\}$$

and S is the range of the parameter x_j. Tables 2 and 3 show the parameter values for the Hartmann function when $n = 3, q = 4$ and $n = 6, q = 4$, respectively. The Hartmann function has 4 and 6 local minima when $n = 3$ and $n = 6$, respectively.

When $n = 3$, $X_{\min} = (0.114, 0.556, 0.882)$ and $f(X_{\min}) = -3.86$. When $n = 6$, $X_{\min} = (0.201, 0.150, 0.477, 0.275, 0.311, 0.657)$ and $f(X_{\min}) = -3.32$ (Al-Sultan *et al*. 1997).

The best result from the proposed FCPSO for the Hartmann test function when $n = 3$ is $f(X) = -3.8628$ where $X = (-0.7707, 0.1113, 0.7051)$. The best results from CPSO (Liu *et al*. 2005), original PSO and GA are -3.8610 ± 0.0033, -3.8572 ± 0.0035 and -3.8571 ± 0.0070, respectively. When comparing the results from FCPSO, CPSO (Liu *et al*. 2005), the original PSO and GA, the best result from the FCPSO is better.

The best result from the proposed FCPSO for the Hartmann function when $n = 6$ is $f(X) = -3.3257$ where $X = (-0.5970, -0.7011, -0.0446, -0.4508, -0.3767, 0.2960)$. The best result

Table 2. The parameters of the Hartmann function (when $n = 3$ and $q = 4$).

i	a_{i1}	a_{i2}	a_{i3}	c_i	p_{i1}	p_{i2}	p_{i3}
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.03815	0.5473	0.8828

Table 3. The parameters of the Hartmann function (when $n = 6$ and $q = 4$).

i	a_{i1}	a_{i2}	a_{i3}	a_{i4}	a_{i5}	a_{i6}	c_i	p_{i1}	p_{i2}	p_{i3}	p_{i4}	p_{i5}	p_{i6}
1	10	3	17	3.5	1.7	8	1	0.1312	0.1619	0.5569	0.0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1451	0.3522	0.2883	0.3047	0.6550
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

from FCPSO, CPSO (Liu *et al*. 2005), PSO and GA are -3.3244 ± 0.0013, -3.1953 ± 0.1352, 2.8943 ± 0.3995 and -3.0212 ± 0.4291, respectively. When comparing the results from FCPSO, CPSO (Liu *et al*. 2005), the original PSO and GA, the best result from the FCPSO is closer to the global minimum.

5.4. *The Shekel function*

The Shekel function is a multidimensional, multi-modal, continuous, deterministic function which, when $n = 4$; $q = 5, 7, 10$, is given by

$$f(X) = -\sum_{i=1}^{q} \left[c_i + \sum_{j=1}^{n} (x_j - a_{ji})^2 \right]^{-1}, \tag{13}$$

where

$$S = \{X \in R^4 | \, 0 \le x_j \le 10, \, j = 1, 2, 3, 4\}$$

and S is the range of the parameter x_j. $X_{\min} = (4, 4, 4, 4)$. For $q = 5$, $f(X_{\min}) = -10.15$; for $q = 7$, $f(X_{\min}) = -10.40$; and for $q = 10$, $f(X_{\min}) = -10.54$. When $X \in R^4$, the Shekel function has four local minima. Table 4 shows the parameter values for the Shekel function when $X \in R^4$ and $q = 5, 7, 10$ (Al-Sultan *et al*. 1997).

The best result from the proposed FCPSO for the Shekel test function when $q = 5$ is $f(X) = -10.1532$, where $X = (4.0005, 3.9999, 4.0000, 4.0003)$; when $q = 7$, it is $f(X) = -10.4029$, where $X = (4.0005; 4.0007; 4.0004; 3.9997)$; and when $q = 10$, it is $f(X) = -10.5364$, where $X = (4.0008, 4.0006, 3.9996, 3.9995)$.

5.5. *Comparison of the FCPSO with the original PSO and the constriction factor PSO*

The comparisons between these three methods are shown in Table 1. A Lenovo Thinkpad T61, Intel Core 2 Duo CPU T7300 @ 2.00 GHz with 2.97 GB RAM was used for simulation purposes

Table 4. The parameters of the Shekel function (when $n = 4$ and $q = 5, 7, 10$).

i	a_{i1}	a_{i2}	a_{i3}	a_{i4}	c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

with Matlab version 6.5.1. For the test functions, both the FCPSO and the constriction factor PSO achieved the global minimum. The simulation results obtained by the FCPSO are better than the results from the original PSO and the constriction factor PSO, which means the final solutions obtained from the FCPSO are in general closer to the global minimum than those from either the original PSO or the constriction factor PSO. As expected, the computation time is longer than the original PSO and the constriction factor PSO as it is determined by the neural network.

The MATLAB t-test function ttest2 was used to compare the means of the data samples from the original PSO and the FCPSO. Although the 95% confidence interval for the true difference in means is $[-0.0229, 0.0047]$, the FCPSO on average still performed better.

For the Goldstein test benchmark, the 95% confidence interval for the true difference in means is [0.0017, 0.1015]. The mean of FCPSO shows some improvement in Table 4. For the Hartmann function, when $n = 3$, the 95% confidence interval for the true difference in means is $[-0.0091; -0.0040]$; when $n = 6$, the 95% confidence interval for the true difference in means is $[-0.4332, -0.2359]$. For the Shekel function, when $q = 5$, the 95% confidence interval for the true difference in means is $[-7.8780, -6.2115]$; when $q = 7$, the 95% confidence interval for the true difference in means is $[-8.1074, -6.9174]$; when $q = 10$, the 95% confidence interval for the true difference in means is $[-8.3408, -7.6866]$. The means of the FCPSO for the Hartmann and Shekel functions improve more significantly than the original PSO.

The MATLAB t-test function ttest2 was also used to compare the means of the two data samples from the constriction factor PSO and the FCPSO. For the Schaffer f6, although the 95% confidence interval for the true difference in means which is $[-0.0219, 0.0044]$ does not improve significantly, the FCPSO still performed better from Table 1. For the Hartmann function, when $n = 6$, the 95% confidence interval for the true difference in means is $[-0.0937, -0.0032]$. For the Shekel function, when $q = 5$, the 95% confidence interval for the true difference in means is $[-6.1909, -0.8091]$; when $q = 7$, the 95% confidence interval for the true difference in means is $[-6.1678, -1.3699]$; when $q = 10$, the 95% confidence interval for the true difference in means is $[-5.6008, -0.1697]$. The mean of the FCPSO for the Hartmann function (when $n = 6$) and Shekel function improves more significantly than the constriction factor PSO.

It is clear that the original PSO and the constriction factor PSO are not stable and sometimes get trapped in a local optimum. The performance of the FCPSO is more stable than the original PSO and the constriction factor PSO.

6. Conclusion

In this article, a fully connected particle swarm optimization (FCPSO) model was proposed to improve the optimization performance of the PSO. In this new model, all components of each particle are directly participating in the evolutionary optimization process. The effect among different components of each particle was implemented via the back propagation (BP) neural network. Although the complexity is higher than the existing PSO algorithms, the performance of the proposed FCPSO is more stable and more accurate. In the future, the authors will focus on reducing the computational complexity of the proposed structure.

Acknowledgements

This work is supported by Research Funding of the Tshwane University of Technology (TUT), Incentive Funding of the National Research Foundation of South Africa (IFR2009090800049) and the National Scientific Foundation of China (10772135).

References

Al-Sultan, K.S. and Al-Fawzan, M.A., 1997. A tabu search Hooke and Jeeves algorithm for unconstrained optimization. *European Journal of Operation Research*, 103 (1), 198–208.

Clerc, M. and Kennedy, J., 2002. The particle swarm: explosion, stability, and convergence in multidimension complex space. *IEEE Transactions on Evolutionary Computation*, 6 (1), 58–73.

del Valle, Y., *et al.*, 2005. Training MLP neural networks for identification of a small power system: comparison of PSO and backpropagation. *In*: *Proceedings of the 6th international conference on power systems, operation and planning*, Univerdale Jean Pieget, Praia, Cape Verde, 22–26 May. Cape Town: UCT Press, 162–166.

del Valle, Y., *et al.*, 2008. Particle swarm optimization: basic concepts, variants and applications in power systems. *IEEE Transactions on Evolutionary Comptation*, 12 (2), 171–195.

Doctor, S., Venayagamoorthy, G., and Gudise, V., 2004. Optimal PSO for collective robotic search applications. *In*: *Proceedings of the IEEE congress on evolutionary computation*, Portland, OR, 19–23 June, Vol. 2. New York: IEEE Press, 1390–1395.

Fletcher, R. and Powell, M.J.D., 1963. A rapidly convergent descent method for minimization. *Computer Journal*, 6 (2), 163–168.

Gholizadeh, S. and Salajegheh, E., 2009. Optimal design of structures subjected to time history loading by swarm intelligence and an advanced metamodel. *Computer Methods in Applied Mechanics and Engineering*, 198 (37–40), 2936–2949.

Hamidian, D. and Seyedpoor, S.M., 2010. Shape optimal design of arch dams using an adaptive neuro-fuzzy inference system and improved particle swarm optimization. *Applied Mathematical Modelling*, 34 (6), 1574–1585.

Hertz, J., Krogh, A., and Palmer, R.G. 1991. *Introduction to the theory of neural computation*. Boston, MA: Addison Wesley.

Hopfield, J.J., 2007. Hopfield network. *Scholarpedia*, 2 (5), 1977–1981.

Kennedy, J. and Eberhart, R., 1995. Particle swarm optimization. *In*: *Proceedings of the IEEE international conference neural networks*, Perth, Australia, New York: IEEE Press, IV, 1942–1948.

Kotinis, M., 2010. A particle swarm optimizer for constrained multi-objective engineering design problems. *Engineering Optimization*, 42 (10), 907–926.

Liu, B., *et al.*, 2005. Improved particle swarm optimization combined with chaos. *Chaos, Solitons and Fractals*, 25 (5), 1261–1271.

Mohagheghi, S., *et al.*, 2005. A comparison of PSO and backpropagation for training RBF neural networks for identification of a power system with STATCO. *In*: *Proceedings of the IEEE swarm intelligence symposium*, 8–10 June, Atlanta, GA: New York, IEEE Press, 381–384.

More, J.J., 1978. The Levenberg–Marquardt algorithm: implementation and theory. *In*: G.A. Watson, ed. *Proceedings of the 1977 Dundee conference on numerical analysis*. Lecture notes in mathematics 630. Berlin: Springer-Verlag, 105–116.

Perez, R.E. and Behdinan, K., 2007. Particle swarm approach for structural design optimization. *Computers and Structures*, 85 (19–20), 1579–1588.

Rojas, R., 1996. *Neural networks*. Berlin: Springer-Verlag.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J., 1986. Learning internal representations by error propagation. *Parallel Distributed Processing*, 1, 318–362.

Seyedpoor, S.M., Gholizadeh, S., and Talebian, S.R., 2009. An efficient structural optimisation algorithm using a hybrid version of particle swarm optimisation with simultaneous perturbation stochastic approximation. *Civil Engineering and Environmental Systems*, 1 (1), 1–19.

Shi, Y. and Eberhart, R., 1998. A modified particle swarm optimizer. *In*: *IEEE international conference on evolutionary computation*, 4–9 May, Anchorage, Alaska, Piscataway, NJ: IEEE Press, 69–73.

Shi, Y. and Eberhart, R., 2001. Fuzzy adaptive particle swarm optimization. *In*: *Proceedings of the 2001 congress on evolutionary computation*, 27–30 May, Seoul, South Korea, New York: IEEE Press, 101–106.

Toscano, G. and Coello Coello, C.A., 2004. A constraint-handling mechanism for particle swarm optimization. *In*: *Proceedings of the 2004 congress on evolutionary computation*, 19–23 June, Portland, OR, New York: IEEE Service Center, 1396–1403.

van den Bergh, F. and Engelbrecht, A.P., 2000. Cooperative learning in neural networks using particle swarm optimizers. *South African Computer Journal*, 26 (1), 84–90.

Venayagamoorthy, G., 2004. Adaptive critics for dynamic particle swarm optimization. *In*: *Proceedings of the IEEE international symposium on intelligence control*, 2–4 September, Taipei, Taiwan, New York: IEEE Press. 380–384.