zenghui wang <wangzengh@gmail.com>

ICIC 2011 Paper #1565 Decision Notification

ICIC2011 <icic@iim.ac.cn> **Sun, Jun 5, 2011 at 4:47 AM**
To: "wangzengh@gmail.com" <wangzengh@gmail.com>

Dear Author(s)

Congratulations! On behalf of the ICIC 2011 International Program Committee, we are very pleased to inform you that your paper:

Paper ID: **1565**
Author(s): **Zenghui Wang and Yanxia Sun**
Title: **Fully Connected Multi-objective Particle Swarm Optimizer based on Neural Network**
Presentation: **oral**

has been accepted for presentation at the 2011 International Conference on Intelligent Computing, which has been selected into the following Springer volume:

• **Lecture Notes in Computer Sciences (LNCS)**.

This year we received a huge volume of paper submissions from 28 countries and regions, and each paper received averagely 3.63 review reports, only a very limited number of papers (less than the acceptance rate of 27%) are accepted while a large number of high-quality papers have to be rejected due to the space limitation.

IMPORTANT NOTES:

●**If you are planning to give up this paper for LNCS registration and publication, please let us know immediately so that we can pass this chance to others.**

●Please note that the papers submitted to the ICIC conference should not have been submitted to other conferences at the same time. If multiple submissions were found, the paper will be directly excluded from the conference proceedings. In the meantime, such multiple submission information will be passed to the other involved conferences.

●From previous ICIC2006 & ICIC2010 proceedings, it can be found that there are **several severe cases of plagiarism**. So we are kindly reminding all the authors to stipulate your academic behaviour, and cherish and protect your own reputation as well as our ICIC reputation. We hope not to see this phenomenon in future ICIC academic event. We shall try our best to take every measure to stop this misconduct.

To include your paper in the proceedings to be published by Springer, here are the steps you must follow:

1. Please see the attached REVIEWERS' COMMENTS for your paper, which are intended to help you to improve your paper for final publication. The listed comments should be fully addressed, as acceptance is conditional on appropriate response to the requirements and comments. Unsatisfactory final paper submission may be directly rejected by the editor.

2. Please take a look at the link Instruction for Final Paper Submission within http://www.ic-ic.org/icg/index.asp or http://www.ic-ic.org/2011/Instruction.htm to obtain the information for preparing the final version of your paper. Note that in the link Instruction for Final Paper Submission for your paper(s), you should first write your FULL given Names, and then write your FULL Family names, which is an international convention. In addition, the paper length of **8** pages for EACH paper in Springer format will be strictly enforced. If your paper is over the stipulated Maximum length, you will have to pay the over-length page fee at ¥700 RMB or

$120 USD per extra page.

When you have completed your paper and are ready to submit it, please go to the link Final Paper Submission within http://www.ic-ic.org/icg/index.asp to submit your final camera-ready paper. On this page you will need to use the following account and password:

Account: **wangzengh@gmail.com**
Password: **1236987**

Final papers MUST be submitted by June 20, 2011. Any papers submitted after this date run the risk of not being included in the Springer Lecture Notes Series. The paper must be re-submitted even if the reviewers indicated that no changes are required.

3. For each paper to be included in the Springer Lecture Notes Series, at least one of the authors for EACH paper must register for the conference. The deadline for author registration is June 20, 2011. If you register late your paper may not appear in the Springer Lecture Notes Series, which is a Mandatory step without exception for any authors. Please download the Registration Form from the link http://www.ic-ic.org/2011/ Registration Form.htm , and complete it.

Particularly, please make a note of '**ICIC2011**' plus your paper **ID Number** on the **BANK** transfer form when you are paying for a dedicated registration fee for a paper to be published in the ICIC2011 proceedings. Please mark 'Order Hard LNCS Proceedings' or 'Order Hard LNAI Proceedings' or 'Order Hard LNBI Proceedings' or 'Order Extra Meal Tickets' on the **BANK** transfer form transfer form when you are planning to purchase hard proceedings or banquet tickets.

4. Before submitting your final camera-ready paper, you are required to first upload the scanned copy of your bank transfer form (in PDF format) by http://www.ic-ic.org/icg/index.asp.

5. To be published in the conference proceedings, all accepted papers must be accompanied by a signed copyright form (download it from the link http://www.ic-ic.org/2011/Copyright Form.htm or Instruction for Final Paper Submission within http://www.ic-ic.org/icg/index.asp). The title of proceedings is **Advanced Intelligent Computing Technology and Applications-ICIC2011**. The volume editors are De-Shuang Huang, et al. Please download a copy of the copyright form from this link before submitting your final camera-ready paper, complete it and upload its scanned copy (in PDF format) together with the electronic bank transfer form (in PDF format) by http://www.ic-ic.org/icg/index.asp.

6. Hotel reservation information is available on the ICIC 2011 website very soon. Please keep an eye on http://www.ic-ic.org/2011/index.htm and book your room as early as possible!

If you require a visa to enter China please send a request for an invitation letter to Prashan Premaratne, University of Wollongong, Australia at prashan@uow.edu.au. Thank you for participating in ICIC 2011. We look forward to seeing you in Zhengzhou, China, August 11-14, 2011.

Sincerely,

ICIC2011 Program Committee

REVIEWERS' COMMENTS

REVIEW NO. 1

Originality: 2.5
Significance of topic: 3
Technical quality: 2.5
Relevance to ICIC 2011: 3.5

Presentation: 3
Overall rating: 2.9
Reviewer's expertise on the topic: High

Comments to the authors:
This paper presents a Fully Connected Multi-objective Particle Swarm Optimizer (FCMOPSO) which is based on Back Propagation (BP) Neural Network and the BP algorithm is applied to predict the 'gradient descent' direction vector to less objective value. It makes full use of the past information of particles. The FCMOPSO is compared with MOPSO proposed by Moore et al. to solve multi- objective problems. However, there are only three test problems and the results of the experiments are not enough to identify the efficiency of the proposed algorithm. The author should do more experiments to make it more persuasive.

REVIEWERS' COMMENTS

REVIEW NO. 2

Originality: 2.5
Significance of topic: 4
Technical quality: 2.5
Relevance to ICIC 2011: 4
Presentation: 3
Overall rating: 3.2
Reviewer's expertise on the topic: High

Comments to the authors:
A new structured MOPSO model named fully connected Multi-objective Particle Swarm Optimizer (FCMOPSO) is proposed in this paper. The efficiency and effectiveness of the new model is demonstrated by simulation results and the comparative experiments with exiting MOPSO by three Test Problems.
However, there are plenty of particle swarm optimization or hybrid PSO algorithm used for solving multiple problems for the present. So the author should tell the difference between his/her ideal and prior ones, especially the differing from the hybrid PSO-BP algorithms. What is more, strongly recommend that the author should better add some more test problems to improve the persuasion of new algorithm.

REVIEWERS' COMMENTS

REVIEW NO. 3

Originality: 3
Significance of topic: 3
Technical quality: 2.5
Relevance to ICIC 2011: 3.5
Presentation: 2.5
Overall rating: 2.9
Reviewer's expertise on the topic: High

Comments to the authors:
This paper proposed a fully connected multi-objective particle swarm optimizer (FCMOPSO). But the results of the experiments are not enough to identify the efficiency of the research. To demonstrate the capability of the presented algorithm, more benchmark functions need to be performed and compare the results with other algorithm. Moreover, I recommend the author should better adjust the article format and modify the details of the paper.

Comments of PC Member:

Program Committee Decision: This paper must be much strengthened and revised according to the reviewers' comments before it can be officially published.

Fully Connected Multi-Objective Particle Swarm Optimizer Based on Neural Network

Zenghui Wang[1] and Yanxia Sun[2]

[1]School of Engineering, University of South Africa, Florida 1710, South Africa
[2]Department of Electrical Engineering &French South African Institute of Technology (F'SATI), Tshwane University of Technology, Pretoria 0001, South Africa
{wangzengh,sunyanxia}gmail.com

Abstract. In this paper, a new model for multi-objective particle swarm optimization (MOPSO) is proposed. In this model, each particle's behavior is influenced by the best experience among its neighbors, its own best experience and all its components. The influence among different components of particles is implemented by the on-line training of a multi-input Multi-output back propagation (BP) neural network. The inputs and outputs of the BP neural network are the particle position and its the 'gradient descent' direction vector to the less objective value according to the definition of no-domination, respectively. Therefore, the new structured MOPSO model is called a fully connected multi-objective particle swarm optimizer (FCMOPSO). Simulation results and comparisons with exiting MOPSOs demonstrate that the proposed FCMOPSO is more stable and can improve the optimization performance.

Keywords: Multi-objective Optimization, Particle Swarm Optimization, Neural Network, Pareto Front, non-domination.

1 Introduction

The single-objective Particle Swarm Optimization (SOPSO), first introduced by [1], is a stochastic optimization technique that can be likened to the behavior of a flock of birds or the sociological behavior of a group of people. It has been used to solve a range of optimization problems, including neural network training [2] and function minimization [3]. 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 rapidly reach optimum position. Particle swarm algorithms have been used to successfully optimize a wide range of problems [4]. Later, the SOPSO was extended to deal with multi-objective Optimization [5] [6]. The searching processes of the multi-objective PSO (MOPSO) are similar with SOPSO. 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

D.-S. Huang et al. (Eds.): ICIC 2011, LNCS 6838, pp. 170–177, 2011.

the social part of the particle update formula. Both play crucial roles to guide the particle's search.

There are no connections among particle components for most PSOs. But the relationships among the variables of optimization problems can affect the optimization performance as the variables are coupled for most of the optimization problems. There are many classic optimization algorithms in the literature such as Quasi-Newton Methods [7], Gauss-Newton Method, Levenberg–Marquardt Method [8], 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 [9].

Moreover, the PSOs only use little information of past experience, such as the individual best position, the group's best position and so on. Most of the past information will be given up. The neural network is very powerful as it can memorize the information or characteristics of a complicated system. To improve the optimization performance of SOPSO, Sun et al. proposed the full connected PSO for single-objective optimization problems [10] based on neural network which was used to memorize the tendency to a less value of fitness function at different positions which is the most important difference between the full connected PSO and other hybrid PSO-BP algorithms such as references [11], [12], and so on. It showed a good optimization performance. However, the full connected single-objective PSO cannot be directly used for multi-objective optimization problems as there are several optimization objectives, and the choices of the own experience and the swarm experience are different. The present paper aims at proposing a multi-objective version of full connected particle swarm optimization.

The remainder of this paper is organized as follows: Section 2 is an introduction of the multi-objective particle swarm optimization algorithm and BP neural network. Section 3 proposes the multi-objective version of full connected particle swarm optimization. Numerical results and comparisons are provided in Section 4. Finally, the concluding remarks appear in Section 5.

2 A Brief Introduction of Multi-Objective Particle Swarm Optimization

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 [1]. 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 following equations.

$$V_i(t+1) = \omega V_i(t) + c_1 R_1 (P_i - X_i(t)) + c_2 R_2 (P_g - X_i(t)), \quad (1)$$

$$X_i(t+1) = X_i(t) + V_i(t+1), \quad (2)$$

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.

The difference between single-objective optimization and multi-objective optimization is that there are more than one objectives to be optimized for multi-objective optimization, that is, a multi-objective optimization problem can be described as:

$$\begin{aligned} \text{Find: } X^T &= [x_1, x_2, \cdots, x_n], \\ \text{Minimize: } F(X) &= (f_1(X), \cdots, f_m(X), \\ \text{subject to: } X &\in \Omega. \end{aligned} \tag{3}$$

Here, R^m is the objective space, and $F : \Omega \longrightarrow R^m$ consists of m real-valued objective functions. The analogy of PSO with evolutionary algorithms makes evident the notion that using a Pareto ranking scheme [6] could be the straightforward way to extend the approach to handle multiobjective optimization problems. The solution to the multi-objective optimization problem exists in the form of an alternate tradeoff known as a Pareto optimal set. Each objective component of any non-dominated solution in the Pareto optimal set can only be improved by degrading at least one of its other objective components. A vector F_a is said to dominate another vector F_b, denoted as

$$F_a < F_b, \text{iff} f_{a,i} < f_{b,i} \ \forall i = \{1, 2, \cdots, m\} \text{ and } \exists j \in \{1, 2, \cdots, m\} \text{ where } f_{a,j} < f_{b,j} \tag{4}$$

For the more details about MMPSO, please refer to references [6] and [13].

3 Full Connected Multi-Objective Particle Swarm Optimization

As can be seen from equations (1) and (2), the position of a particle, the best experience among its neighbors 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 (1)–(2), the position components of particle i can be thought of as the output of a neural network as shown in Fig. 1(a) [10].

However, the relationship and influence only relies on the corresponding dimensional components of the particle swarm which is easily seen in Fig. 1(a). 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 gradient descent direction of the optimization problem. The inputs and outputs of the neural network are the positions of the particles and the gradient descent direction vector of the better experience of the particles, respectively. Here the back propagation neural network is used to learn the dynamics at different positions and the "Levenberg–Marquardt Method" [8] is chosen to train back propagation neural network.. The fully connected particle swarm structure based on multi-input multi-output BP neural network is therefore proposed which is shown in Fig. 1(b). 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, $\bigtriangledown X_i = [\bigtriangledown x_i^1, \bigtriangledown x_i^2, \ldots, \bigtriangledown x_n^1,]$, reflects the gradient descent unit vector of the particle i. In order to take advantage of each component of the particle itself, the gradient descent unit vector $\bigtriangledown X_i(t)$ is added to equation (2). In our proposed fully connected particle swarm optimization, when the previous position was dominated by a position, a back propagation neural network is used to get the gradient descent unit vector at different position of the particle. 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

$$\bigtriangledown x_i^j = \frac{X_i^j(t_1) - X_i^j(t_0)}{||X_i(t_1) - X_i(t_0)||}. \tag{5}$$

Here, $X_i^j(t_1)$ is the current position in problem space if the current position dominates the objective values related to $X_i(t_0)$ in objective space; $X_i(t_0)$ is updated with $X_i^j(t_0)$ when $X_i^j(t_1)$ is changed. For the improvement index $\Delta X_i^j(t)$, a random part is introduced to improve the search ability. Based on trial and error, the parameter α can be chosen in the range $[0.01, 0.2]$, in this paper, 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 j^{th} component of particle i is described as

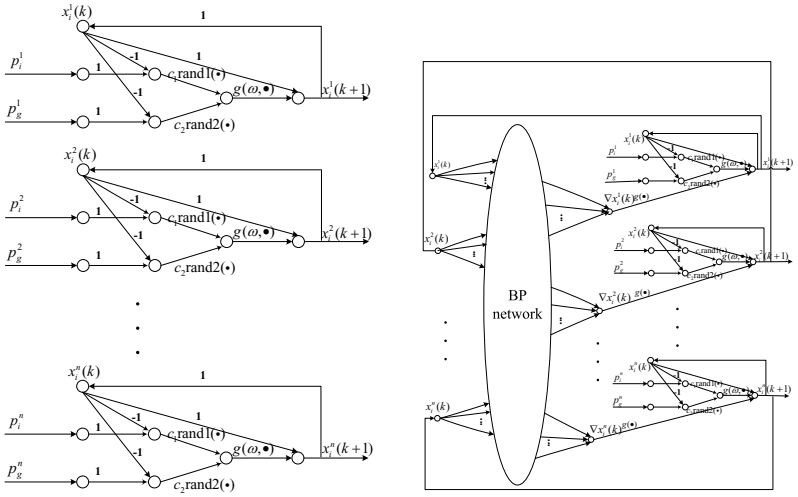
$$\Delta X_i^j(t) = g(\bigtriangledown x_i^j) = \frac{\alpha * rand(1) \bigtriangledown x_i^j}{generations}. \tag{6}$$

Equations (2) is therefore changed into the following equations:

$$X_i(t+1) = X_i(t) + V_i(t+1) + \Delta X_i(t), \tag{7}$$

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

– 1) Initialize the BP neural network (the BP neural network can be trained using the existing data firstly), the swarm and assign a random position in



(a) The structure of the classical PSO

(b) The structure of the fully connected PSO based on the multi-input multi-output BP neural network

Fig. 1. Structures of Particle swarm optimization

the problem hyperspace to each particle and calculate the fitness function which is given by the optimization problem whose variables are corresponding to the elements of particle position coordinates.

- 2) Synchronously update the positions of all the particles according to equations (2), (8), and change the two states every iteration.
- 3) Evaluate the fitness function for each particle.
- 4) Find the non-dominated Pareto front and store it in the repository set. (The data from the first 50 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 5 epochs).
- 5) Repeat steps 2)–4) until a stopping criterion is met (e.g., maximum number of iterations).

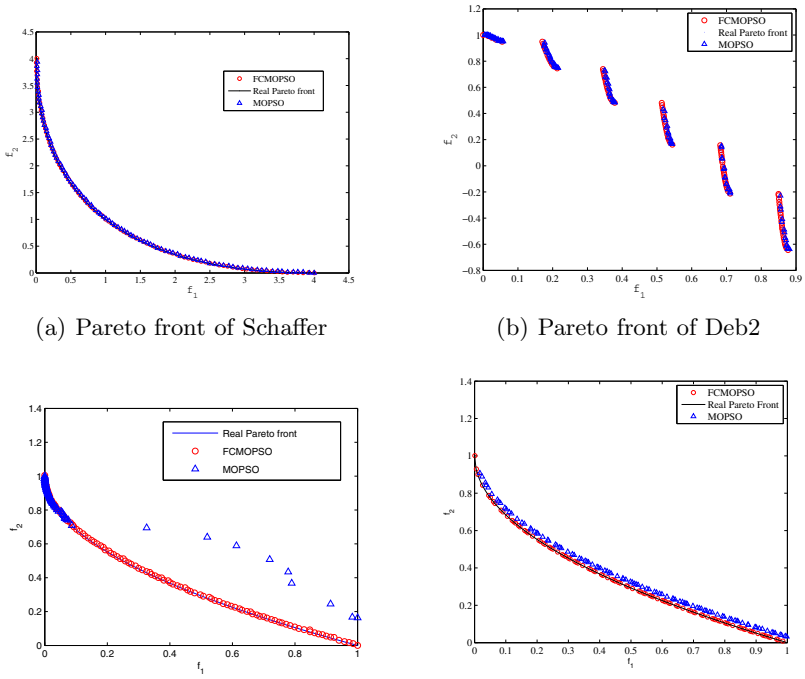
4 Comparison between FCMOPSO and MOPSO

4.1 Test Problems

The test problems are Shaffer, Deb 2, ZDT1 and ZDT 4. For the Schaffer function [14][13], the Pareto front is convex and connected. The Pareto front of Deb 2 [13] is disconnected and convex. The Pareto front of ZDT1 [14][13] is convex. The Pareto front of ZDT4 [14][13] is connected and non-convex. All of them are typical benchmark functions.

In this section, the performance of this proposed method is compared with multi-objective PSO [6]. In these examples, the total number of fitness function evaluations was set to 20 000. The particle number of particles is 100. A random initial population was created for each of the 20 runs on each test problem. The maximum number of external repository particles is 100.

Using the full connected multi-objective PSO and the multi-objective PSO in reference [6], the Pareto fronts are the 'o' symbols and the '△' symbols in Figs. 2(a)-2(d), respectively. From Figs. 2(a)-2(d), it can be seen that the proposed



(a) Pareto front of Schaffer

(b) Pareto front of Deb2

(c) Pareto front of ZDT1

(d) Pareto front of ZDT4

Fig. 2. Pareto fronts

method performs better than the algorithm in reference [6]. It would be better to use some performance metrics to find the comparison results other than just the figures as it is more convincing.

4.2 Performance Metrics

In order to provide a quantitative assessment for the performance of MO optimizer, two metrics are often taken into consideration, i.e., Generational Distance and Spacing metric [6] [13].

1)The metric of generational distance (GD) gives a good indication of the gap between the discovered Pareto front and the true Pareto front. The GD comparison of FCMOPSO and MOPSO [6] is shown in Table 1.

Table 1. GD comparison of FCMOPSO and MOPSO [6]

Problem	Method	Max	Mean	Min	std.dev
Schaffer	MOPSO	$2.194 * 10^{-4}$	$7.117 * 10^{-5}$	$3.244 * 10^{-5}$	$8.257 * 10^{-5}$
Schaffer	FCPSO	$2.210 * 10^{-5}$	$1.022 * 10^{-5}$	$2.792 * 10^{-6}$	$4.952 * 10^{-6}$
Deb2	MOPSO	$1.102 * 10^{-4}$	$9.817 * 10^{-5}$	$5.440 * 10^{-5}$	$9.510 * 10^{-6}$
Deb2	FCMOPSO	$9.412e-05$	$6.700 * 10^{-5}$	$4.170 * 10^{-5}$	$7.243 * 10^{-6}$
ZDT1	MOPSO	0.0048	0.0023	0(Pareto front converges to a point)	0.0023
ZDT1	FCMOPSO	$1.927 * 10^{-4}$	$2.001 * 10^{-4}$	$2.054 * 10^{-4}$	$7.131 * 10^{-6}$
ZDT4	MOPSO	$2.177 * 10^{-4}$	$1.010 * 10^{-3}$	$2.177 * 10^{-4}$	$9.138 * 10^{-4}$
ZDT4	FCMOPSO	$3.127 * 10^{-4}$	$1.017 * 10^{-4}$	$4.950 * 10^{-5}$	$5.831 * 10^{-5}$

2)To measure the distribution of vectors throughout the non-dominated vectors found so far, the spacing metric is often used [6]. This metric can show how well the Pareto front found is if all the points are on or very close to the real Pareto front. At this situation, the smaller the spacing metric is, the better the particles are spread along the Pareto front. It would be better to use the spacing metric together with the Pareto front figure; otherwise it would be difficult to conclude the performance just according to the spacing metric. For example, all the Pareto front points converged to one point and the space metric is 0 in one simulation. The spacing metric comparison of FCMOPSO and MOPSO [6] is shown in Table 2.

Table 2. Spacing metric comparison of FCMOPSO and MOPSO [6]

Problem	Method	Max	Mean	Min	std.dev
Schaffer	MOPSO	0.0124	0.0071	0.0009	0.0044
Schaffer	FCPSO	$7.221 * 10^{-4}$	$3.267 * 10^{-4}$	$2.961 * 10^{-4}$	$1.249 * 10^{-4}$
Deb2	MOPSO	0.0175	0.0096	0.0044	0.0021
Deb2	FCMOPSO	0.0062	0.0045	0.0037	0.0016
ZDT1	MOPSO	0.0472	0.0230	0 (Pareto front converges to a point)	0.0211
ZDT1	FCMOPSO	0.0034	0.0033	0.0030	$3.1426 * 10^{-5}$
ZDT4	MOPSO	0.0349	0.0071	0.0068	0.0163
ZDT4	FCMOPSO	0.0141	0.0124	0.0112	0.0015

As can be seen from the statistic Tables 1 and 2, the proposed method can achieve better Pareto front than the classic MOPSO [6] for Schaffer, Deb2 and ZDT1 optimization problems. Although according to Table 2 the spacing metric of FCMOPSO is not better than the one of MOPSO, it can be shown that the obtained Pareto front is better than the one of MOPSO as shown in Fig. 2(d). The reason for the spacing metric of ZDT4 is that the Pareto front of MOPSO shrinks as it is away from the true Pareto front.

5 Conclusion

A fully connected multi-objective particle swarm optimization (FCMOPSO) model was proposed to improve the optimization performance of the MOPSO. For 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 MOPSO algorithms, the performance of the proposed FCMOPSO is more stable and more accurate than the classic MOPSO.

References

1. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proceeding of IEEE International Conference Neural Networks, Perth, Australia, pp. 1942–1948. IEEE Press, New York (1995)
2. van den Bergh, F., Engelbrecht, A.P.: Cooperative Learning in Neural Networks Using Particle Swarm Optimizers. South African Computer Journal 26, 84–90 (2000)
3. Shi, Y., Eberhart, R.: A Modified Particle Swarm Optimizer. In: IEEE International Conference on Evolutionary Computation, Piscataway, Alaska, USA, pp. 69–73. IEEE Press, New York (1998)
4. Kotinis, M.: A Particle Swarm Optimizer for Constrained Multiobjective Engineering Design Problems. Engineering Optimization (2010), doi:10.1080/03052150903505877
5. Moore, J., Chapman, R.: Application of Particle Swarm to Multiobjective Optimization. Department of Computer Science and Software Engineering, Auburn University (1999)
6. Coello, C.A., Pulido, G.T., Lechuga, M.S.: Handling Multiple Objectives With Particle Swarm Optimization. IEEE Trans. on Evolu. Comp. 8, 256–279 (2004)
7. Fletcher, R., Powell, M.J.D.: A Rapidly Convergent Descent Method for Minimization. Computer Journal 6, 163–168 (1963)
8. More, J.J.: The Levenberg-Marquardt Algorithm: Implementation and Theory. In: Watson, G.A. (ed.) Numerical Analysis. Lecture Notes in Mathematics, pp. 105–116. Springer, Heidelberg (1977)
9. Hertz, J., Krogh, A., Palmer, R.G.: Introduction to the Theory of Neural Computation. Addison-Wesley, Reading (1991)
10. Sun, Y., Djouani, K., Qi, G., van Wyk, B.J., Wang, Z.: Fully Connected Particle Swarm Optimizer. Engineering Optimization 43, 801–812 (2011)
11. Liu, J., Qiu, X.: A Novel Hybrid PSO-BP Algorithm for Neural Network Training. In: Proceedings of the Second International Joint Conference on Computational Sciences and Optimization, pp. 300–303. IEEE Press, New York (2009)
12. Hu, J., Zeng, X.: A Hybrid PSO-BP Algorithm and Its Application. In: 2010 Sixth International Conference on Natural Computation, pp. 2520–2523. IEEE Press, New York (2010)
13. Coello Coello, C.A., Lamont, B.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York (2007)
14. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Trans. on Evolu. Comp. 6, 182–197 (2002)