International Conference on Control, Engineering & Information Technology (CEIT'13)
Proceedings Engineering & Technology - Vol.4, pp. 6-10, 2013
Copyright - IPCO

Online Control of Nonlinear Systems using Neuro-Fuzzy Design tuned with Cooperative Particle Sub-Swarms Optimization

Sana Bouzaida
Research Unit: Industrial Systems Study
and Renewable Energy
National School of Engineers Of Monastir
Monastir, Tunisia
bouzaida_sana@hotmail.fr

Anis Sakly
Research Unit: Industrial Systems Study
and Renewable Energy
National School of Engineers Of Monastir
Monastir, Tunisia
sakly_anis@yahoo.fr

***Abstract*— This paper proposes a TSK-type Neuro-Fuzzy system tuned with a novel learning algorithm. The proposed algorithm used an improved version of the standard Particle Swarm Optimization algorithm, it employs several sub-swarms to explore the search space more efficiently. Each particle in a sub-swarm correct her position based on the best other positions, and the useful information is exchanged among the particles during the iteration process. Simulations on Neuro-fuzzy control of two non-linear plants are conduct to verify algorithm performance and indicate that the proposed algorithm outperforms the standard Particle Swarm Optimization algorithm.**

Keywords*— Particle Swarm Optimization; Neuro-Fuzzy design; Cooperative Sub-Swarms; Online Control; Nonlinear Systems

I. Introduction

The use of conventional approaches for control problems requires an appropriate model that describes dynamic behavior of the plant to be controlled. However, models are difficult to obtain due to the complexity and nonlinearity of the controlled plant. In order to overcome this problem, other control design approaches which are not relied on nominal models are introduced. One of the successful methods is the Neuro fuzzy models.

Neuro-Fuzzy systems have seen increasing interest in last decade, they have been widely used in control problems [1]. Various methods have been applied to optimize the parameters of Neuro-fuzzy systems [2]. More recently, optimization techniques based on Evolutionary Algorithms (EAs) were proposed for tuning Neuro-Fuzzy models, such as Particle Swarm Optimization algorithm PSO [3] and genetic algorithm GA [4]. However, in complex problems, the original PSO may get trapped in a local optimum. Many variants of PSO have been proposed to improve the performance of original PSO [5,6].

In this paper, a modified version of PSO named Cooperative Particles Sub-Swarms Optimization "CPSSO" is proposed for tuning Neuro-Fuzzy systems. The proposed learning mechanism is based mainly on the cooperation between sub-swarms and exchanging information among each other to reach the global optimum.

The rest of this paper is organized as follows: in Section II TSK neuro-fuzzy model structure is reviewed. Section III describes the proposed algorithm. Then, Section IV illustrates simulation results and compares the performance of developed algorithm with original PSO and GA. Conclusions are given in the last section.

II. Structure of TSK-type fuzzy model

A TSK-type fuzzy model is one of the most efficient Neuro-fuzzy model that approximate a general class of non-linear systems [7]. Typically, the TSK model employs If-Then rules, where the rule consequents are usually constant values or linear functions of inputs. The j^{th} rule in the fuzzy model is represented in the following form given by (1)

$$\textbf{If } x_1 \text{ is } A_1^j \text{ and } \dots \text{ and } x_n \text{ est } A_n^j \textbf{ Then } y \text{ is } f_j \qquad (1)$$

where $x_1,..,x_n$ are input variables, y is the system output variable, A_i^j is a fuzzy set, and f_j is a linear function. Fuzzy set A_i^j uses a Gaussian membership function given by (2)

$$A_i^j = \exp\left\{-\frac{1}{2}\left(\frac{x_i - c_{ij}}{\sigma_{ij}}\right)^2\right\} \qquad (2)$$

The structure of a TSK model is shown in Fig. 1 where n and R are the number of input dimensions and the number of rules respectively.

It is a five-layer network structure. In the proposed TSK fuzzy model, the firing strength of a fuzzy rule is calculated by performing the following " and " operation, as in (3)

$$\mu_j = \prod_{i=1}^{n} A_i^j \qquad (3)$$

The output is computed by (4)

$$y = \frac{\sum_{j=1}^{R} \mu_j f_j}{\sum_{j=1}^{R} \mu_j} \tag{4}$$

The fuzzy design can be considered as an optimization problem, where the parameter set of the premise part and consequent part of Takagi and Sugeno's Neuro-fuzzy model are required to be identified. In this paper, we adopt a TSK-type Neuro-fuzzy model to perform control problems. The different parameters of TSK model are optimized by a novel evolutionary algorithm that is called CPSSO.

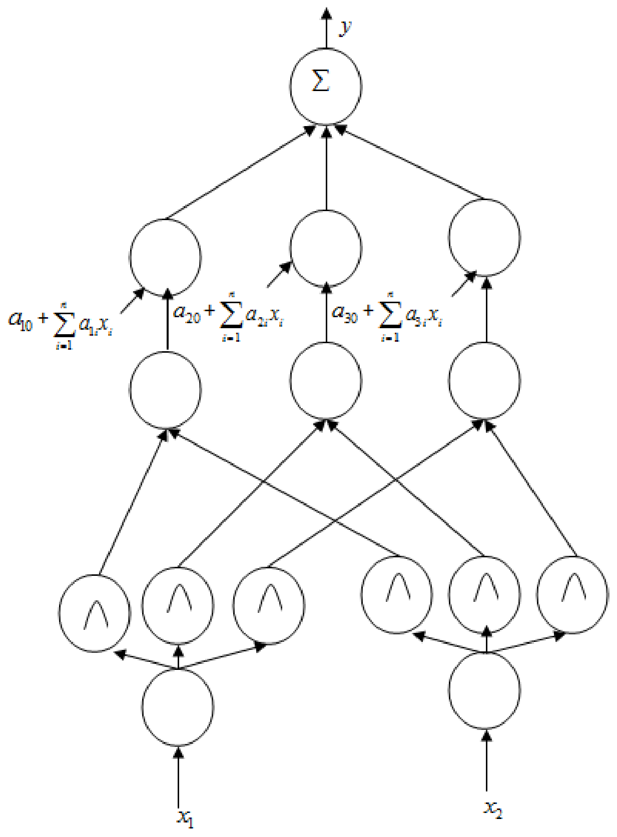

y
Σ
$a_{10}+\sum_{i=1}^{n} a_{1i}x_i$
$a_{20}+\sum_{i=1}^{n} a_{2i}x_i$
$a_{30}+\sum_{i=1}^{n} a_{3i}x_i$
x_1
x_2

Fig. 1. Structure of TSK-type fuzzy model

III. An Overview of the Novel Learning Algorithm

A. *Particle Swarm Optimization algorithm*

The PSO was developed from observation of the social behavior of animals including bird flocking and fish schooling when looking for food.

The population is referred to a swarm and the individual is called as particle in PSO. Each particle has a position and a velocity, representing the potential solution to a given problem. The position and velocity of each particle are adjusted by (5) and (6).

$$v_{k+1} \leftarrow w.v_k + c_1.r_1(gbest - x_k) + c_2.r_2(pbest - x_k) \tag{5}$$

$$x_{k+1} \leftarrow x_k + v_{k+1} \tag{6}$$

where v and x represent the velocity and the position of the particle, respectively, c_1 and c_2 are positive constants namely acceleration constants, r_1 and r_2 are random numbers between 0 and 1, *pbset* is the best position found by the particle, *gbest* is the global best position, and finally w is the inertia weight.

B. *Cooperative Particle Sub-Swarms Optimization*

In the standard PSO, each particle adjusts the velocity and position according to its own best position *pbest* and the best position found by all its neighbors *gbest*. However, in complex optimization problems the optimal solution cannot be found by simply adjusting the inertia w and parameters c_1 and c_2. In recent literatures, learning strategies used more information found by all the particles to balance the local and global search abilities [8].

In this study, we propose a novel learning strategy that involves exchanging information among each other in order to explore the search space more efficiently and reach the best solutions.

Same as other population based algorithms, CPSSO starts with a population of particles that is partitioned into n sub-swarms. In the proposed algorithm, two search techniques are used: local search and global information exchange technique.

In the first technique, the particles in each sub-swarm improves their positions to reach the best local solution, they used the best local position in the group as *pbest*, and the best solution found by all the sub-systems as a *gbest*, when updating their velocity and position.

In the second technique, the cooperation between the best particles *pbest* in the different groups is modeled in order to find a global better solution. Each particle in this step chooses another position based on the best local positions *pbest* of the other particles and some random factors.

The global solution is directly applied to update the position and velocity of the next generation.

The basic steps of the proposed algorithm are as follow

- **Step1**: Initialize the positions and velocities of all the sub-systems.
- **Step2**: Calculate the fitness of all the particles and update the velocity and position vectors according to the best local positions *pbest* and the global best position *gbest*.
- **Step3**: Identify the best particle in each sub-swarm and make position correction using a probability rate (*Pr*) :

$$x_i^{j'} \begin{cases} x_i^{j'} \in \left(x_i^1, x_i^2, \ldots, x_i^M\right) & \text{with probability } Pr \\ x_i^{j'} = x_i^j + RA & \text{with probability } (1-Pr) \end{cases} \tag{7}$$

where M is the number of sub-swarms, N is the number of parameters, and RA is the Distance Radius which is defined by (8)

$$RA = RA_{\min}(\max(x_i)-\min(x_i))\exp\left(\frac{\ln\left(\frac{RA_{\min}}{RA_{\max}}\right)it}{itm}\right) \quad (8)$$

where it is the iteration number, itm is the maximum number of iterations, and $RA_{\min}$ and $RA_{\max}$ are respectively the minimum and the maximum of distance radius.

- **Step 4**: Update the *gbest,* and *pbest* of each sub-swarm.
- **Step 5**: repeat Step 3-6 for a given maximal number of iterations.

The Flowchart of the developed Algorithm is shown in Fig.2.

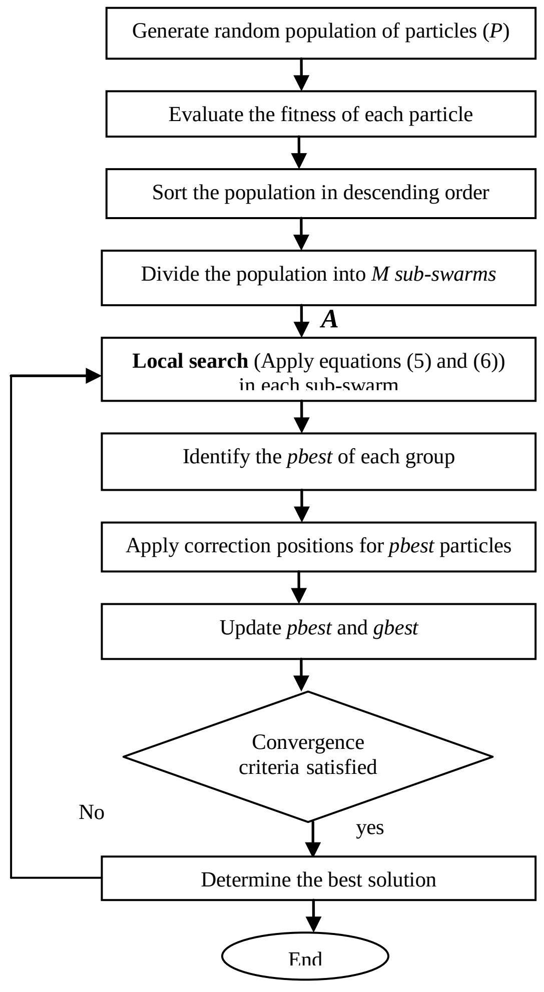

Generate random population of particles (P)
Evaluate the fitness of each particle
Sort the population in descending order
Divide the population into M sub-swarms
A
Local search (Apply equations (5) and (6)) in each sub-swarm
Identify the pbest of each group
Apply correction positions for pbest particles
Update pbest and gbest
Convergence criteria satisfied
No
yes
Determine the best solution
End

Fig. 2. Flowchart of the developed Algorithm (CPSSO)

IV. SIMULATION RESULTS

Two benchmark of non-linear plants control problems are used to validate the effectiveness of the proposed method "CPSSO. The performance of CPSSO is compared with the standard PSO and genetic algorithm GA.

In these examples, the plants are unknown and on-line training parameters is applied for the control purpose. The overall control structure is illustrated in Fig. 3.

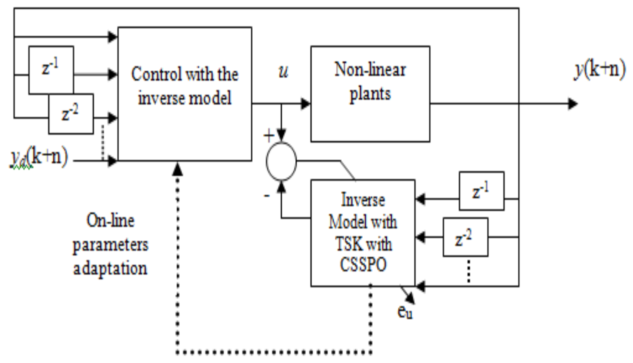

z-1
z-2
Control with the inverse model
u
Non-linear plants
y(k+n)
yd(k+n)
+
-
On-line parameters adaptation
Inverse Model with TSK with CSSPO
z-1
z-2
eu

Fig. 3. On-line strategy of control

For the different examples, the initial parameters are given in Table I.

TABLE I. Initial parameters value

Parameter	Value
Popultaion size (P)	50
Probability consideration rate (PR)	0.5
$RA_{\max}$, $RA_{\min}$	1e-2, 1e-7
Acceleration parameters c1, c2	2, 2
Mutation probability	0.2

Example 1: Water bath temperature control

The plant of water bath temperature control is described by the following difference equation

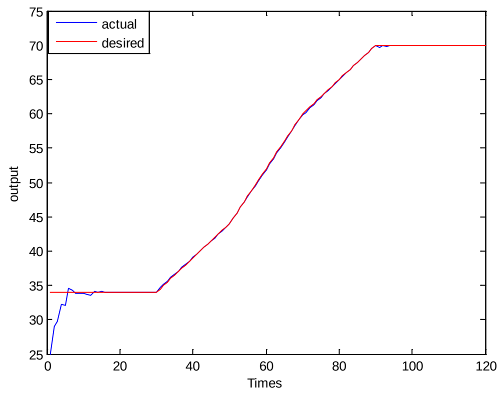
$$y(k+1) = e^{-\alpha_0 T_s}y(k) + \frac{\frac{\beta_0}{\alpha_0}(1-e^{-\alpha_0 T_s})}{1+e^{0.5y(k)-40}}u(k) + (1-e^{-\alpha_0 T_s})y_0 \quad (9)$$

where α_0=1.0015e^{-4}, β_0=8.67973e^{-3} and y_0=25(°C), which were obtained from a real water bath plant [9]. The sampling period is T_s=30s.

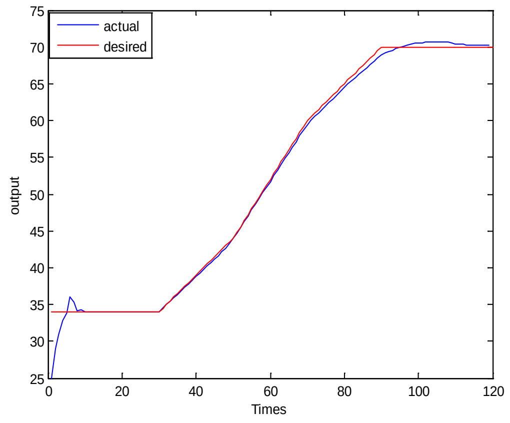
The reference temperature y_d is set as follows

$$y_d \begin{cases} 34^\circ C & \text{for } k\leq 30 \\ (34+0.5\cdot(k-30))^\circ C & \text{for } 30<k\leq 50 \\ (44+0.8\cdot(k-50))^\circ & \text{for } 50<k\leq 70 \\ (60+0.5\cdot(k-70))^\circ C & \text{for } 70<k\leq 90 \\ 70^\circ C & \end{cases} \quad (10)$$

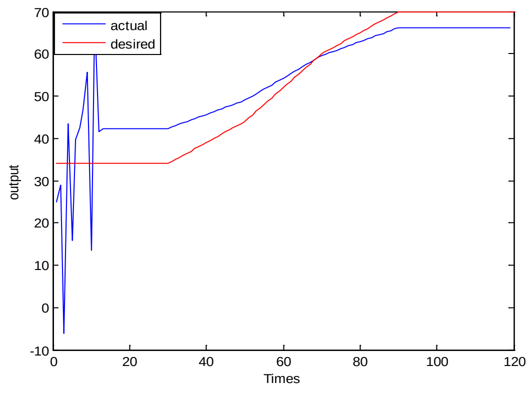
Fig. 4 illustrates the tracking performance of CPSSO method. In this example, y(k) and y_d(k+1) are used as inputs, and $\tilde{u}(k)$ as output. The number of rules is fixed to 4, thus resulting in 28 parameters. The evolution progressed for 100 generations. Simulation results confirm the effectiveness of CPSSO in control problems.



(a)



(b)



(c)

Fig. 4. Results of the desired output and CPSSO output (a), PSO output (b), and GA output (c).

Example 2: Dynamic plant control

The dynamic plant to be controlled is given by [10]:

$$y(k+1)=\frac{y(k)y(k-1)(y(k)+2.5)}{1+y^2(k)+y^2(k-1)}+u(k) \qquad (11)$$

The desired output y_d is given by the following 250 samples of data:

$$y_d(k+1)=0.6y_d(k)+0.2y_d(k-1)+r(k) \qquad (12)$$

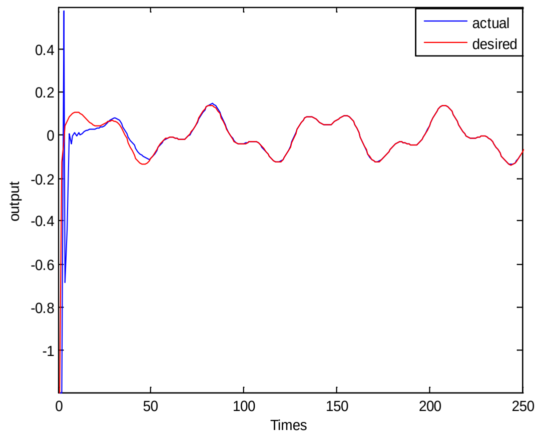
where

$$r(k)=0.2\sin(\frac{2\pi k}{25})+0.4\sin(\frac{\pi k}{32})+0.3\cos(\frac{\pi k}{25}) \qquad (13)$$

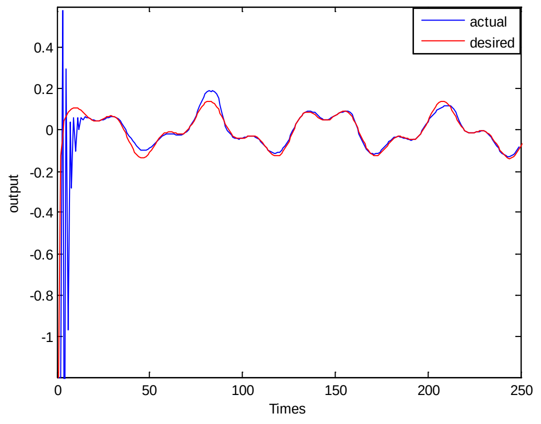
By using TSK type Neuro-fuzzy controller with CPSSO y(k-1), y(k) and y_d(k+1) are used as inputs, and $\tilde{u}(k)$ as output. The evolution progressed for 100 generations. The tracking performance of CPSSO method is shown in Fig. 5. The results were compared with PSO and GA with the same initial parameters. The comparison results are tabulated in Table 3. To test the performance of different methods, we considered the sum of absolute error "*SAE*" which is defined by (14)

$$\boldsymbol{SAE}=\sum_k \left|y_{ref}(k)-y(k)\right| \qquad (14)$$

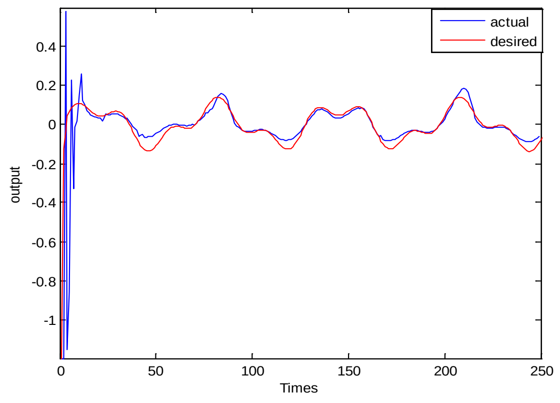
Table II shows that CPSSO algorithm obtains smaller errors then the standard PSO.



(a)



(b)



(c)

Fig. 5. Results of the desired output and CPSSO output (a), PSO output (b), and GA output (c).

Table II. Performance comparison of various existing models

Sum of absolute error	**CPSSO**	PSO	GA
Example 1	**28**	57.56	102
Example 2	**5.23**	7.5	9.8

V. CONCLUSION

In this paper, a new meta-heuristic search technique, called as CPSSO is applied for tuning parameters of TSK-type neuro-fuzzy model. By comparing with PSO and GA in the aspects of optimal solution, results show that CPSSO has much better optimizing accuracy in the different examples of control problems.

REFERENCES

[1] F.J. Lin, C.H. Lin, P.H. Shen, "Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive", IEEE Transactions on Fuzzy Systems, vol. 5, pp. 751–759, 2001.

[2] D. Simon, "Training fuzzy systems with the extended Kalman filter", Fuzzy Sets and Systems, vol. 132, pp. 189–199, 2002.

[3] A. Chatterjee, K. Watanabe, "An optimized Takagi–Sugeno type neuro-fuzzy system for modeling robot manipulators", Neural Comput. Appl., vol. 1, pp. 55–61, 2006.

[4] G. Leng, T.M. McGinnity, G. Prasad, "Design for self-organizing fuzzy neural networks based on genetic algorithms", IEEE Transactions on Fuzzy Systems, vol. 14, pp. 755–766, 2006.

[5] C.J. Lin, "An efficient immune-based symbiotic particle swarm optimization learning algorithm for TSK-type neuro-fuzzy network design", Fuzzy Sets Sys, vol. 159, pp. 2890-2909, 2008.

[6] R. Mendes, J. Kennedy, J. Neves, "The fully informed particle swarm: simpler, maybe better", IEE Trans.Evol.Comput., vol. 8, pp. 204-210, 2004.

[7] C.F. Juang, C.T. Lin, "An self-constructing neural fuzzy inference network and its applications", IEEE Transactions on Fuzzy Systems, vol. 1, pp. 12–31, 1998.

[8] J.j. Liang, A.K Qin, P.N. Suganthan, S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodel functions", IEEE Trans.Evol.Comput , vol. 3, pp. 281-285, 2006.

[9] J. Tanomaru, S. Omatu, "Process control by self-constructing trained neural controllers", IEEE Transactions on Industriels Electronics, pp. 511-521, 1992.

[10] C.F. Juang, "A TSK-type recurrent fuzzy network for dynamic systems processing by neural networks and genetic algorithms", IEEE Trans. Fuzzy Syst, vol. 2, pp. 155-170, 2002.