Applied Soft Computing 11 (2011) 2687–2697

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Recurrent fuzzy system design using elite-guided continuous ant colony optimization

Chia-Feng Juang*, Po-Han Chang

Department of Electrical Engineering, National Chung Hsing University, 250 Kuo Kuang Road, Taichung 402, Taiwan, ROC

ARTICLE INFO

Article history:
Received 19 October 2009
Received in revised form 25 October 2010
Accepted 1 November 2010
Available online 13 November 2010

Keywords:
Ant colony optimization
Fuzzy system models
Swarm intelligence
Recurrent fuzzy neural networks
Dynamic plant control

ABSTRACT

This paper proposes recurrent fuzzy system design using elite-guided continuous ant colony optimization (ECACO). The designed recurrent fuzzy system is the Takagi–Sugeno–Kang (TSK)-type recurrent fuzzy network (TRFN), in which each fuzzy rule contains feedback loops to handle dynamic system processing problems. The ECACO optimizes all of the free parameters in each recurrent fuzzy rule in a TRFN. Unlike the general ant colony optimization that finds solutions in discrete space, the ECACO finds solutions in a continuous space. The ECACO is a population-based optimization algorithm. New solutions are generated by selection, Gaussian random sampling, and elite-guided movement. To verify the performance of ECACO, three examples of dynamic plant control are simulated using ECACO-optimized TRFNs. The ECACO performance is also compared with other continuous ant colony optimization, particle swarm optimization, and genetic algorithms in these simulations.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Recurrent fuzzy systems (RFSs) are fuzzy systems with feedback connections in their structure. For temporal characteristic problems, the performance of a RFS has been shown to outperform feed-forward fuzzy systems and recurrent neural networks in several studies [1–11]. Many feedback structures in RFSs have been proposed. One category of recurrent RFSs uses feedback loops from the network output(s) as a recurrence structure [1,2]. Another category of recurrent RFSs uses feedback loops from internal state variables as the recurrence structure [4–8]. The local recurrence property of the RFSs in studies [3,4] is achieved by feeding the output of each membership function locally back to itself; thus, each membership value is only influenced by its past values. Recurrent self-organizing neural fuzzy inference networks (RSONFIN) [5] and Takagi–Sugeno–Kang (TSK)-type recurrent fuzzy networks (TRFN) [6] use a global feedback structure, where the firing strengths of each rule are summed and fed back as internal network inputs. The TRFN is constructed from a series of recurrent fuzzy if–then rules with TSK-type consequent parts, and its performance is shown to be better than RSONFIN, in which a fuzzy set is used as the consequence. Compared with other RFSs, which also use TSK-type consequences [1,9], one major advantage of TRFN is that no *a priori* knowledge of the plant order is required, which eases the design process. In [6,8], the superiority of TRFN to recurrent neural networks in spatial–temporal problems, including dynamic plant identification and control, was demonstrated. Therefore, this paper selects TRFN as the designed RFS.

In addition to the difference in feedback structures, RFSs differ in their learning methods. Most RFSs are learned through gradient descent-based learning algorithms [1–9]. One disadvantage of this type of learning algorithm is the local optima problem. When there are multiple peaks in a search space, search results usually get trapped in a local solution when the gradient descent learning algorithm is used. Another problem is that the input–output training data for gradient descent learning algorithms may not be directly available. For example, for the dynamic control problem considered in this paper, the desired control outputs for a RFS controller are unknown in advance for gradient descent learning. For the problems above, a design of RFSs using population-based optimization algorithms has been proposed [11–18]. One popular approach is the use of genetic algorithms (GAs) for RFS parameter optimization [6,11,13]. In [6], an elite genetic algorithm (EGA) was proposed for TRFN parameter optimization. Another popular approach is the use of particle swarm optimization (PSO) [12,15,16]. For example, TRFN design using PSO was proposed in [12]. Different approaches to combining GA and PSO for RFS designs were proposed in [12,14,17]. For example, the hybrid of GA and PSO (HGAPSO) for TRFN design was proposed in [12]. The HGAPSO introduces the idea of crossover and mutation operations into individuals in PSO for performance improvement. In this paper, a new learning algorithm based on continuous ant colony optimization is proposed for TRFN design to further improve performance.

* Corresponding author. Tel.: +886 4 22851549x806; fax: +886 4 22851540.
E-mail address: cfjuang@dragon.nchu.edu.tw (C.-F. Juang).

1568-4946/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2010.11.001

The use of a new meta-heuristic, ant colony optimization (ACO), for solving optimization problems has been proposed recently [19]. The ACO technique is inspired by real ant colony observations. It is a multi-agent approach that was originally proposed to solve difficult discrete combinatorial optimization problems. In the original ACO meta-heuristic, artificial ant colonies cooperate to find good solutions for difficult discrete optimization problems. The ACO has been applied to feed-forward fuzzy system design problems in several studies [20–22]. Because the optimization space is restricted to be discrete, the designed FSs are unsuitable for problems where high accuracy is a major concern. Recently, some continuous ACO algorithms for optimization in continuous space have been proposed [23–25], where one promising approach is the ACO in real space ($ACO_{\mathbb{R}}$) [25]. The good performance of $ACO_{\mathbb{R}}$ for continuous function optimization has been demonstrated. Based on the $ACO_{\mathbb{R}}$ concept, this paper proposes elite-guided continuous ant colony optimization (ECACO) and applies it to TRFN design for dynamic plant control.

The major contribution of this paper is twofold. First, a new continuous ACO, the ECACO, is proposed. Like GA and PSO, the ECACO works with a population of solutions. The ECACO proposes a new approach for new solution generation at each iteration. Second, the ECACO is applied to TRFN design. To the best of our knowledge, this is the first paper that applies continuous ACO to RFS design. The superiority of ECACO in comparison with $ACO_{\mathbb{R}}$, EGA, HGAPSO, and other advanced PSO algorithms is demonstrated in three simulation examples.

This paper is organized as follows. Section 2 introduces the TSK-type recurrent fuzzy network (TRFN). Section 3 introduces the TRFN design by ECACO. Section 4 analyzes the ECACO algorithm. Section 5 presents simulation results of the ECACO-designed TRFN for dynamic plant control. This section also compares the ECACO performance with other optimization algorithms. Section 6 discusses the similarity and difference between the ECACO and discrete ACO. This section also discusses the major factors that the ECACO outperforms GAs and PSO in learning performance. Finally, Section 7 presents the conclusions.

2. TSK-type recurrent fuzzy network (TRFN)

Fig. 1 shows the TRFN structure [6]. Each recurrent fuzzy if–then rule in TRFN, consisting of r rules and n external inputs, $x_1,\ldots,x_n$, is in the following form:

$$\begin{aligned}
&\text{Rule } i: \text{If } x_1(t) \text{ is } A_{i1} \text{ and } x_2(t) \text{ is } A_{i2} \text{ and } \ldots \\
&\qquad \text{and } x_n(t) \text{ is } A_{in} \text{ and } h_i(t) \text{ is } G \\
&\quad \text{Then } y(t+1) \text{ is } a_{i0} + \sum_{j=1}^{n} a_{ij}x_j(t) + a_{in+1}h_i(t) \\
&\text{And } h_1(t+1) \text{ is } v_{i1} \text{ and } h_2(t+1) \text{ is } v_{i2} \text{ and } \ldots \\
&\qquad \text{and } h_r(t+1) \text{ is } v_{ir}
\end{aligned} \tag{1}$$

where A_{ij} and G are fuzzy sets, and v_{ij} and a_{ij} are the consequent parameters for inference output h_i and y, respectively. The consequent part for the external output y is a TSK type and is a linear combination of the external input variables x_j and internal variables h_i, plus a constant. In TRFN, the recurrent property comes from feeding the internal variables, which are derived from fuzzy firing strengths, back to both the network input and output layers. In this configuration, each internal variable is responsible for memorizing the temporal history of its corresponding fuzzy rule.

To give a clear understanding of the mathematical function of each node, we will describe the TRFN functions layer by layer. For notation convenience, the net input to the ith node in layer k is denoted by $u_i^{(k)}$ and the output value by $O_i^{(k)}$.

Layer 1: No function is performed in this layer. The node only transmits input values to layer 2.

Layer 2: Nodes in layer 2 act as membership functions. Two types of membership functions are used in this layer. For external input, $u_j^{(2)} = x_j$, the following Gaussian membership function is used:

$$O_j^{(2)} = \exp\left\{-\frac{(u_j^{(2)} - m_{ij})^2}{b_{ij}^2}\right\} \quad \text{and} \quad u_j^{(2)} = O_j^{(1)}, \tag{2}$$

where m_{ij} and b_{ij} are, respectively, the center and the width of the Gaussian membership function of the ith term in the jth input variable x_j. For internal variable, $u_i^{(2)} = h_i$, the following sigmoid membership function is used:

$$O_i^{(2)} = \frac{1}{1+\exp\{-u_i^{(2)}\}} \quad \text{and} \quad u_i^{(2)} = O_i^{(5)}. \tag{3}$$

Layer 3: Each node in this layer calculates the firing strength of a rule by product operation. The function of each rule is

$$O_i^{(3)} = \prod_{j=1}^{n+1} O_j^{(2)} = \frac{1}{1+\exp\left(-O_i^{(5)}\right)} \cdot \exp\left\{-\sum_{j=1}^{n}\left(\frac{O_j^{(1)} - m_{ij}}{b_{ij}}\right)^2\right\}. \tag{4}$$

Layer 4: Nodes in this layer perform a linear summation of the TSK-type consequent. The mathematical function of each node i is

$$O_i^{(4)} = \sum_{j=0}^{n+1} a_{ij}u_j^{(4)} = a_{io} + \sum_{j=1}^{n} a_{ij}x_j + a_{in+1}h_i. \tag{5}$$

Layer 5: The context node functions as a defuzzifier for the fuzzy rules with inference output h. The link weights represent the singleton values in the consequent part of the internal rules. The simple weighted sum is calculated in each node:

$$h_i = O_i^{(5)} = \sum_{j=1}^{r} O_j^{(3)} v_{ij}. \tag{6}$$

As in Fig. 1, the delayed value of h_i is fed back to layer 1 and acts as an input variable to the antecedent part of a rule. Each rule has a corresponding internal variable h and is used to determine the degree of influence of the temporal history to the current rule.

Layer 6: The node in this layer computes the output y of the TRFN. The output node, together with links connected to it, acts as a defuzzifier. The mathematical function is

$$y = O^{(6)} = \frac{\sum_{j=1}^{r} O_j^{(3)} O_j^{(4)}}{\sum_{j=1}^{r} O_j^{(3)}} \tag{7}$$

For TRFN design using the ECACO, the number of rules r in a TRFN is assigned in advance. The ECACO optimizes all of the free parameters, including m_{ij} and b_{ij} in Layer 2, v_{ij} in Layer 5, and a_{ij} in Layer 4, in the TRFN.

3. Elite-guided continuous ant colony optimization (ECACO) for TRFN design

In discrete ACO [19], the value of a variable S is selected from a discrete set according to sampling from a discrete probability density function (PDF). The fundamental idea underlying continuous ACO is the extension of ACO from the discrete solution domain and discrete PDF to the continuous solution domain and continuous PDF, respectively. Because the most popular continuous PDF is the Gaussian function, it has been used in several studies [24,25].