11

Combination of Particle Swarm and Ant Colony Optimization Algorithms for Fuzzy Systems Design

Chia-Feng Juang
Department of Electrical Engineering, National Chung-Hsing University
Taiwan, R.O.C.

1. Introduction

Fuzzy systems (FSs) have been extensively applied to automatic control, pattern recognition, and decision analysis. However, a common bottleneck is encountered in the derivation of fuzzy rules, which is often difficult, time consuming, and relies on expert knowledge. To automate the design of FSs, many metaheuristic learning algorithms have been proposed. One major optimization category uses Swarm Intelligence (SI) model (Kennedy et al., 2001). The SI technique studies collective behavior in decentralized systems. Its development was based on mimicking the social behavior of animals or insects in an effort to find the optima in the problem space. SI models are initialized with a population of random solutions. One well-known SI model is particle swarm optimization (PSO) (Kennedy & Eberhart, 1995). Many modified PSO models have been proposed and successfully applied to different optimization problems (Clerc & Kennedy, 2002; Bergh & Engelbrecht, 2004; Ratnaweera et al., 2004; Juang, 2004; Kennedy & Mendes, 2006; Parrott & Li, 2006; Chen & Li, 2007). FS design using PSO has also been proposed in several studies (Juang, 2004; Chatterjee et al., 2005; Juang et al., 2007; Araujo & Coelho, 2008; Sharma et al., 2009).

Another well-known SI is ant colony optimization (ACO) (Dorigo & Stutzle, 2004). 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, such as the traveling salesman problem (TSP) (Dorigo et al., 1996; Dorigo & Gambardella, 1997). In the original ACO meta-heuristic, artificial ant colonies cooperate to find good solutions for difficult discrete optimization problems. Different ACO models have been applied to FS design problems (Cassillas et al., 2000; Cassillas et al., 2005; Mucientes & Casillas; 2007; Juang & Lo, 2007; Juang et al., 2008; Juang & Lu; 2009). In (Cassillas et al.,2000; Mucientes & Casillas; 2007; Juang et al., 2008; Juang & Lu; 2009), the FS input space was partitioned in grid type with antecedent part parameters of an FS manually assigned in advance. In (Juang & Lo, 2007), the FS input space was flexibly partitioned using a fuzzy clustering-like algorithm in order to reduce the total number of rules. For all of these studies, the consequent part parameters were optimized in discrete space using ACO. Since only the consequent part parameters are optimized, and the optimization space is restricted to be discrete, the designed FSs are unsuitable for problems where high accuracy is a major concern.

Source: Fuzzy Systems, Book edited by: Ahmad Taher Azar,
ISBN 978-953-7619-92-3, pp. 216, February 2010, INTECH, Croatia, downloaded from SCIYO.COM

Several studies on the combination of PSO or ACO with other optimization algorithms have been proposed in order to improve the performance of the original optimization model. In (Juang, 2004), a hybrid of GA and PSO, called HGAPSO, was proposed. In HGAPSO, new individuals are created not only by PSO but also by the crossover and mutation operations of a GA. In (Fan et al., 2004; Juang & Hsu, 2005), the simplex method was introduced into PSO. In (Ling et al., 2008), a hybrid of PSO with wavelet mutation was proposed. To apply the ACO technique to solve continuous optimization problems, several studies on the combination of ACO with other continuous optimization methods have been performed (Feng & Feng, 2004; Ge et al., 2004). An ACO followed by immune operation for optimization in a continuous space was proposed in (Feng & Feng, 2004). The incorporation of a deterministic searching algorithm (the Powell method) into ACO for continuous optimization was proposed in (Ge et al., 2004).

This chapter studies the combination of PSO and ACO for FSs design. One problem of PSO in FS design is that its performance is affected by initial particle positions, which are usually randomly generated in a continuous search space. A poor initialization may result in poor performance. Searching in the discrete-space domain by ACO helps to find good solutions. However, the search constraint in a discrete-space domain restricts learning accuracy. The motivation on the combination of ACO and PSO is to compensate the aforementioned weakness of each method in FS design problems. Two combination approaches, sequential and parallel, for PSO and ACO proposed in (Juang & Lo, 2008; Juang & Wang, 2009) are described and discussed in this Chapter.

This chapter is organized as follows. Section 2 describes the FS to be designed. The rule generation algorithm and rule initialization are also described in this section. Section 3 describes PSO and how to apply it to FS design. Section 4 describes ACO and how to apply it to FS design. Section 5 describes the sequential combination of PSO and ACO for FS design. Section 6 describes the parallel combination of PSO and ACO for FS design. Finally, Section 7 draws conclusions.

2. Fuzzy systems

2.1 Fuzzy system functions

This subsection describes the FS to be designed. The FS is of zero-order Takagi-Sugeno-Kang (TSK) type. That is, the $i\text{th}$ rule, denoted as R_i, in the FS is represented in the following form:

$$R_i: \text{If } x_1(k) \text{ is } A_{i1} \text{ And} \ldots \text{And } x_n(k) \text{ is } A_{in}, \text{Then } u(k) \text{ is } a_i \tag{1}$$

where k is the time step, $x_1(k),...,x_n(k)$ are input variables, $u(k)$ is the system output variable, A_{ij} is a fuzzy set, and a_i is a crisp value. Fuzzy set A_{ij} uses a Gaussian membership function

$$M_{ij}(x_j) = \exp\{-(\frac{x_j - m_{ij}}{b_{ij}})^2\} \tag{2}$$

where m_{ij} and b_{ij} represent the center and width of the fuzzy set A_{ij}, respectively. In the inference engine, the fuzzy AND operation is implemented by the algebraic product in

fuzzy theory. Thus, given an input data set $\bar{x} = (x_1, \ldots, x_n)$, the firing strength $\phi_i(\bar{x})$ of rule i is calculated by

$$\phi_i(\bar{x}) = \prod_{j=1}^{n} M_{ij}(x_j) = \exp\left\{-\sum_{j=1}^{n}\left(\frac{x_j - m_{ij}}{b_{ij}}\right)^2\right\} \tag{3}$$

If there are r rules in an FS, the output of the system calculated by the weighted average defuzzification method is

$$u = \frac{\sum_{i=1}^{r}\phi_i(\bar{x})a_i}{\sum_{i=1}^{r}\phi_i(\bar{x})}, \tag{4}$$

where a_i is the rule consequent value in (1). There are a total of $D = r(2n+1)$ free parameters in an FS, all of which are optimized using the combination of PSO and ACO algorithms.

2.2 Rule generation and initialization

Most studies on SI-based FS design algorithms determine the number of rules by trial and errors and assign the initial FS parameters randomly and uniformly in the domain of each free parameter. The subsection describes one promising rule generation and initialization algorithm based on the fuzzy clustering-like approach that has been used in an SI algorithm (Juang et al., 2007). It is assumed that there are initially no rules in the designed FS. The rule generation method generates fuzzy rules online upon receiving training data. Rules are generated in order to ensure that at least one rule is activated with a firing strength larger than a pre-defined threshold $\phi_{th} \in (0,1)$ for each input $\bar{x}$. Geometrically, as Fig. 1 shows, this

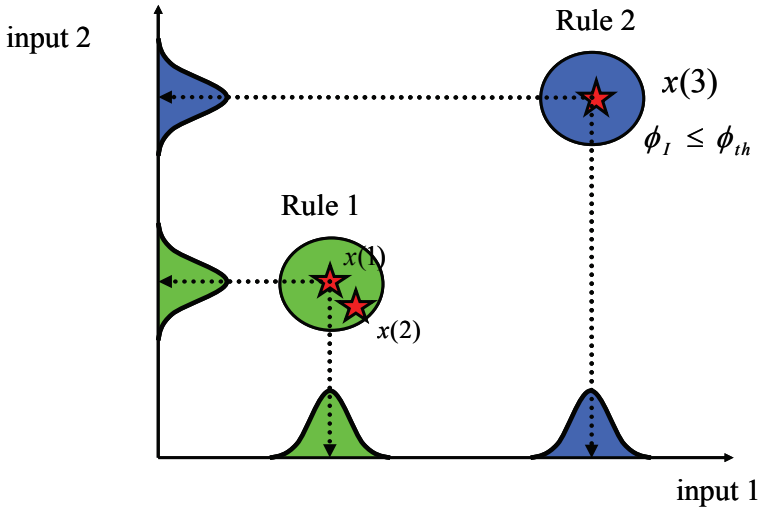

Fig. 1. Distributions of input data, generated fuzzy rules that properly cover the data, and initial shapes of the corresponding fuzzy sets in each input dimension.

threshold ensures that each input data is properly covered by a rule in the input space. According to this concept, the firing strength $\phi_i(\vec{x})$ in (3) is used as the criterion to decide if a new fuzzy rule should be generated. For each incoming piece of data $\vec{x}(k)$, find

$$I = \arg \max_{1 \le i \le r} \phi_i(\vec{x}(k)) \tag{5}$$

where r is the number of existing rules at time t. If $\phi_I \le \phi_{th}$ or r =0, then a new fuzzy rule is generated to cover $\vec{x}(t)$ and $r \leftarrow r+1$. A smaller ϕ_{th} value generates a smaller number of rules. The generation of the r th rule also generates the r th new fuzzy set in each input variable. That is, the number of fuzzy sets in each input dimension is equal to the number of fuzzy rules in the designed FS. To reduce the number of fuzzy sets in each input dimension, the fuzzy set generation criterion proposed in (Juang et al., 2007; Juang & Wang, 2009) can be further employed though it adds computation cost. For each newly generated fuzzy rule, the corresponding center and width of Gaussian fuzzy set A_{rj} in each input variable are assigned as follows:

$$m_{rj} = x_j(k),\ b_{rj} = b_{fix},\ j = 1,...,n \tag{6}$$

where b_{fix} is a pre-specified constant value. Since the centers and widths of all fuzzy sets can be further tuned by PSO, all of the initial widths are simply set to the same value of b_{fix}.

3. Particle swarm optimization (PSO) for FS design

This section first describes the basic concept of PSO. The application of PSO to optimize the generated FS in Section 2 is then is then described. The swarm in PSO is initialized with a population of random solutions (Kennedy & Eberhart, 1995). Each potential solution is called a particle. Each particle has a position, which is represented by a position vector $\vec{s}_i$. A swarm of particles moves through the problem space, with the velocity of each particle represented by a velocity vector $\vec{v}_i$. At each time step, a function f is evaluated, using $\vec{s}_i$ as an input. Each particle keeps track of its own best position, which is associated with the best fitness it has achieved so far, in a vector $\vec{p}_i$. Furthermore, each particle is defined within the context of a topological neighborhood that is made up of itself and other particles in the swarm. The best position found by any member of the neighborhood is tracked in $\vec{p}_i^g$. For a global version of PSO, $\vec{p}_i^g$ is defined as the best position in the whole population. At each iteration t, a new velocity for particle i is obtained by using the individual best position, $\vec{p}_i(t)$, and the neighborhood best position, $\vec{p}_i^g(t)$:

$$\vec{v}_i(t+1) = w\vec{v}_i(t) + c_1\vec{\phi}_1 \cdot (\vec{p}_i(t) - \vec{x}_i(t)) + c_2\vec{\phi}_2 \cdot (\vec{p}_i^g(t) - \vec{x}_i(t)), \tag{7}$$

where w is the inertia weight, c_1 and c_2 are positive acceleration coefficients, and $\vec{\phi}_1$ and $\vec{\phi}_2$ are uniformly distributed random vectors in [0,1], where a random value is sampled for each dimension. The limit of $\vec{v}_i$ in the range $[-\vec{v}_{\max}, \vec{v}_{\max}]$ is problem dependent. For some problems, if the velocity violates this limit, it is reset within its proper limits. Depending on their velocities, each particle changes its position according to the following equation:

$$\vec{s}_i(t+1) = \vec{s}_i(t) + \vec{v}_i(t+1). \tag{8}$$

Based on (7) and (8), the particle population tends to cluster around the best.
The use of PSO for FS design, i.e., optimization of all free parameters in an FS, is described as follows. For the FS in (1) that consists of n input variables and r rules, all of its free parameters can be described by the following position vector

$$\vec{s} = [m_{11}, b_{11}, \cdots, m_{1n}, b_{1n}, a_1, \cdots \cdots, m_{r1}, b_{r1}, \cdots, m_{rn}, b_{rn}, a_r] \in \Re^D \tag{9}$$

After the rule generation and initialization process described in Section 2, the initial antecedent part parameters are determined. Based on the solution vector representation in (9) and the antecedent part parameter initialization in (6), the ith solution vector $\vec{s}_i$ is generated:

$$\begin{aligned} \vec{s}_i &= [s_{i1}\ s_{i2}\ \cdots\ \cdots\ s_{iD}] \\ &= [m_{11} + \Delta m_{11}^i, b_{fix} + \Delta b_{11}^i, \cdots, m_{1n} + \Delta m_{1n}^i, b_{fix} + \Delta b_{1n}^i, a_1, \cdots\cdots, \\ &\quad m_{r1} + \Delta m_{r1}^i, b_{fix} + \Delta b_{r1}^i, \cdots, m_{rn} + \Delta m_{rn}^i, b_{fix} + \Delta b_{rn}^i, a_r] \end{aligned} \tag{10}$$

where Δm_{ij} and Δb_{ij} are small random numbers. The parameter a_i is a random number randomly and uniformly distributed in the FS output range. The evaluation function f for a particle $\vec{s}_i$ is computed according to the performance of the FS constituted of the parameters in (10).

4. Ant colony optimization (ACO) for FS design

ACO is a meta-heuristic algorithm inspired by the behavior of real ants, and in particular how they forage for food (Dorigo & Caro, 1999; Dorigo & Stutzle, 2004). It was first applied to the traveling salesman problem (TSP). In ACO, a finite size colony of artificial ants is created. Each ant then builds a solution to the problem. While building its own solution, each ant collects information based on the problem characteristics and on its own performance. The performance measure is based on a quality function $F(\cdot)$. ACO can be applied to problems that can be described by a graph, where the solutions to the optimization problem can be expressed in terms of feasible paths on the graph. Among the feasible paths, ACO is used to find an optimal one which may be a locally or globally optimal solution. The information collected by the ants during the search process is stored in the pheromone trails, τ, associated to the connection of all edges. These pheromone trails play the role of a distributed long-term memory about the whole ant search process. The ants cooperate in finding a solution by exchanging information via the pheromone trials. Edges can also have an associated heuristic value, η, representing *a priori* information about the problem instance definition or run-time information provided by a source different from the ants. Ants can act concurrently and independently, showing a cooperative behavior. Once all ants have computed their tours (i.e. at the end of the each iteration), ACO algorithms update the pheromone trail using all the solutions produced by the ant colony. Each edge belonging to one of the computed solutions is modified by the amount of pheromone that is proportional to its solution value. The pheromone trail may be updated locally while an ant builds its trail or globally when all ants have built their trails.
Let $\tau_{ij}(t)$ be the pheromone level on edge (i, j) at iteration t, and η_{ij} be the corresponding heuristic value. The probability that an ant chooses j as the next vertex when it is at the vertex i at iteration t is given by

$$p_{ij}(t)=\begin{cases}\dfrac{\tau_{ij}(t)\eta_{ij}^{\beta}(t)}{\sum_{z\in J(i)}\tau_{iz}(t)\eta_{iz}^{\beta}(t)}, & \text{if } j\in J(i)\\ 0 & \text{otherwise}\end{cases} \tag{11}$$

where $J(i)$ is the set of vertices that remain to be visited by the ant, β is a parameter that determines the relative influence of the pheromone trail and the heuristic information. After all ants have completed their tours, the pheromone level is updated by

$$\tau_{ij}(t+1)=\rho\tau_{ij}(t)+\Delta\tau_{ij}(t)\,, \tag{12}$$

where $\rho\in(0,1)$ is a parameter such that $1-\rho$ represents the evaporation coefficient. The update value $\Delta\tau_{ij}$ is related to the quality value *F*. Many updating rules for $\Delta\tau_{ij}$ have been studied (Dorigo & Stutzle, 2004), like ant system (Dorigo et al., 1996), ant colony system (Dorigo & Gambardella, 1997), MAX MIN ant system (Stutzle & Hoos, 2000), and hypercube framework ACO (Blum & Dorigo, 2004). The major differences between these ACO algorithms and AS are the probability selection techniques or pheromone update.

ACO can be applied to design the consequent part parameters in an FS. The general design approach is described as follows. Consider the FS whose structure and antecedent part parameters are determined according to (5) and (6) in Section 2. Suppose the consequent part is selected from a discrete set $U=\{u_1,...,u_m\}$. For each rule, there are m candidate actions to be chosen. Each rule with competing consequent may be written as

$$R_i:\text{If } x_1 \text{ is } A_{i1} \text{ And } \ldots \text{ And } x_n \text{ is } A_{in} \text{ Then } u(k) \text{ is } u_1 \text{ Or } u_2 \text{ Or } \ldots. \text{ Or } u_m\,. \tag{13}$$

That is, we have to decide one from a total of m^r combinations of consequent parts. This combinatorial problem is solved by ACO. To select the consequent value of each rule by ACO, we regard a combination of selected consequent values for a whole FS as a tour of an ant. For example, in Fig. 2, there are four rules, denoted by $R_1,...,R_4$, in an FS and three candidate values, $u_1,...,u_3$, for each rule. Starting from the initial state, the nest, the ant moves through $R_1,...,R_3$ and stops at R_4, where the tour of this ant is marked by a bold line. For each rule, the node visited by the ant is selected as the consequent part of the rule. For the whole FS constructed by the ant in Fig. 2, the consequent values in R_1, R_2, R_3, and R_4 are u_2, u_3, u_1, and u_3, respectively. Selection of the consequent value is partially based on pheromone trails between each rule. The size of the pheromone matrix is $r\times m$ and each entry in the matrix is denoted by τ_{ij}, where $i=1,...,r$ and $j=1,...,m$. As shown in Fig. 2, when the ant arrives at rule R_q, then selection of the m candidate values (denoted by nodes) of R_{q+1} is partly based on τ_{q+1j}, $j=1,...,N$. By using only the pheromone matrix, the transition probability is defined by

$$p_{ij}(t)=\frac{\tau_{ij}(t)}{\sum_{z=1}^{m}\tau_{iz}(t)} \tag{14}$$

In an FS, different fuzzy rules may share the same consequent value. That is, when value a_j is selected as the consequent of rule *i*, it may also be selected as the consequent of following rules *i*+1,…,*r*. For this reason, the set *J*(*i*) in (11) is released to the whole set *U* for all *i* in (14).

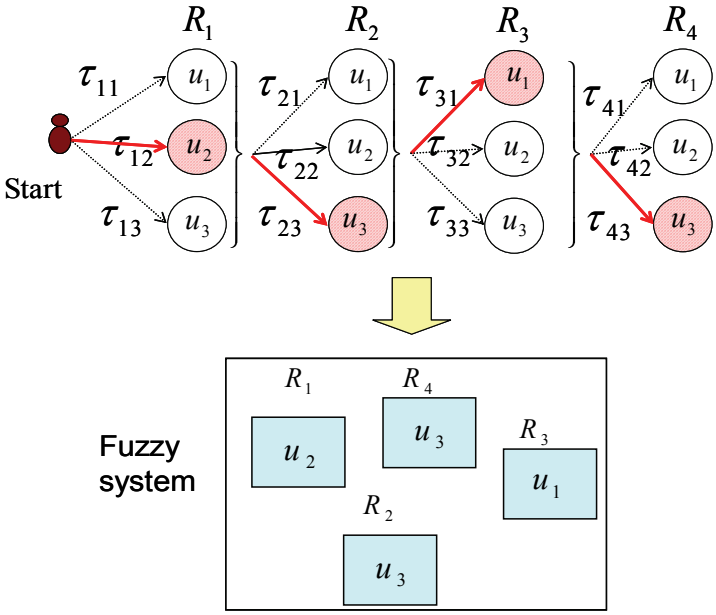

Fig. 2. The relationship between an ant path and the selected consequent values in an FS.

The ACO works without the use of heuristic values, and the consequent part can be simply selected by using (14). The use of heuristic values can be further employed for learning performance improvement. However, determination of the heuristic value η usually requires *a priori* information about the problem instance. For an unknown plant control problem using FSs, it is difficult to assign proper heuristic values in advance. For this problem, in (Juang & Lo, 2007), a new heuristic value assignment approach is proposed for controlling an unknown plant. For the control problems considered in this study, it is assumed that neither *a priori* knowledge of the plant model nor training data collected in advance are available. This study proposes an on-line heuristic value update approach according to temporal difference error between the actual output $y(k)$ and the desired output $y_d(k)$. In (Juang et al., 2008), a simple heuristic value assignment approach is proposed for controlling a plant with an unknown model except the information on the change of output direction with control input. This study assigns heuristic values to each candidate consequent value according to the corresponding fuzzy rule inputs which are control error $e(k) = y(k) - y_d(k)$ and its change with time $\Delta e(k) = e(k) - e(k-1)$. In (Juang & Lu, 2009), the q-values in a reinforcement fuzzy Q-learning algorithm are used as heuristic values for an unknown plant control. Each candidate is assigned with a q-value which is updated using success and failure reinforcement signals during the reinforcement learning process.

5. Sequential combination of ACO and PSO

This section describes the sequential combination approach of ACO and PSO proposed in (Juang & Lo, 2008). In this sequential combination approach, the rule consequent of each on-

line generated rule described in Section 2 is first learned by ACO. The advantage of using ACO for rule consequent learning is that it can help determine a good fuzzy rule base for subsequent learning. However, the search constraint in a discrete-space domain restricts learning accuracy, and the ants do not optimize antecedent part parameters. Therefore, after ACO learning, PSO is then employed to further optimize both the antecedent and consequent parameters, where initial particles in PSO are generated according to learning results from ACO.

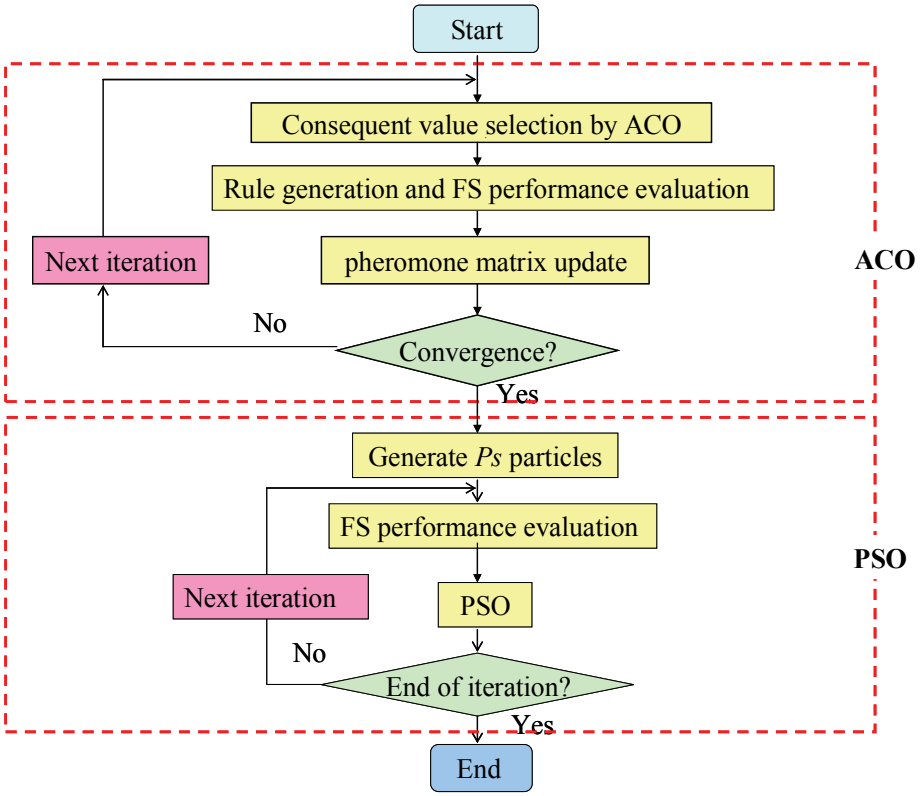

Fig. 3. Flow chart of sequential combination of ACO and PSO for FS design.

Figure 3 shows the flow of the sequential combination of ACO and PSO. The formulation of consequent part learning by ACO is described in Section 4. Detailed function evaluation and update of pheromone levels are described as follows. The pheromone trails, τ_{ij}, on the ant tour are updated according to the performance of the constructed FS. When an ant completes a tour, the corresponding FS is evaluated by a quality function F, which is defined as the inverse of learning error. A higher F value indicates better performance. Let the population size be P_s, meaning that there are P_s ants in a colony. For each iteration, after all the ants in the colony have completed their tours, i.e., after the construction of P_s FSs, select the one with the highest F from the initial iteration until now. If a new global best ant is found in this iteration, then pheromone trails on the tour traveled by the global best ant are updated; otherwise, no pheromone update is performed in this iteration. Denote the global

best ant as q^* with the corresponding quality value as F_{q^*}. The new pheromone trail $\tau_{ij}(t+1)$ is updated by

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t), \text{ if } (i,j) \in \text{global-best-tour} \tag{15}$$

where $0 < \rho < 1$ is the pheromone trail evaporation rate and

$$\Delta\tau_{ij}(t) = F_{q*} \tag{16}$$

The ACO learning iteration above repeats until the criterion for switching is met. The switching point from ACO to PSO learning is determined by the learning error convergence property of the global best ant. Let $E(t)$ denote the error index of the global best ant at iteration t. For example, $E(t)$ can be defined as root-mean-squared error (RMSE) or sum of absolute error (SAE). If

$$\frac{E(t) - E(t+50)}{E(t)} < 1\% \tag{17}$$

then ACO learning terminates and learning switches to PSO.

Using PSO releases the discrete space constraint imposed on consequent parameters when ACO is used, and searches the best consequent parameters in continuous space. In addition to the consequent parameters, PSO also searches the optimal antecedent part parameters. Like ACO, population size in the PSO is equal to P_S. The elements in position $\vec{s}$ are set as in (9). At iteration $t = 0$, the initial positions $\vec{s}_1(0), \cdots, \vec{s}_{P_s}(0)$ are generated randomly according to the best-performing FS, denoted as $\vec{s}_{ACO}$, found in ACO. Position $\vec{s}_1(0)$ is set to be the same as $\vec{s}_{ACO}$. The left $P_S - 1$ particles, $\vec{s}_2(0)$, ..., $\vec{s}_{P_s}(0)$, are generated by adding uniformly distributed random numbers to $\vec{s}_{ACO}$. That is,

$$\vec{s}_i(0) = \vec{s}_{ACO} + \vec{w}_i, \; i = 2, ..., P_s \tag{18}$$

where $\vec{w}_i$ is a random vector. The initial velocities, $\vec{v}_i(0)$, $i = 1, ..., P_s$, of all particles are randomly generated. The performance of each particle is evaluated according to the FS it represents. The evaluation function f is defined as the error index $E(t)$ described above. According to f, we can find individual best position $\vec{p}_i$ of each particle and the global best particle $\vec{p}_g^{\,i}$ in the whole population. Velocity and position of each particle are updated using (7) and (8), respectively. The whole learning process ends when a predefined criterion is met. In (Juang & Lo, 2008), the criterion is the total number of iterations. In (Juang & Lo, 2008), the sequential combination of ACO and PSO approach for FS design has been applied to different control problems, including chaotic system regulation control, nonlinear plant tracking control, and water bath temperature control. The performance of the sequential combination approach has been shown to be better than those of ACO, PSO and different existing FS design methods which were applied to the same problem.

6. Parallel combination of ACO and PSO

This section describes the parallel combination approach of ACO and PSO for FS design (Juang & Wang, 2009). Like the sequential combination approach, the parallel combination

approach uses a constant population size and is denoted as P_s =2N. Each individual in the population represents a parameter solution of the FS as described in (9). An individual may be generated by an ant path in ACO or a particle in PSO. Individuals generated by ants and particles are called ant individuals and particle individuals, respectively. Figure 4 shows a block diagram of the algorithm. Generation of population individuals are described as follows.

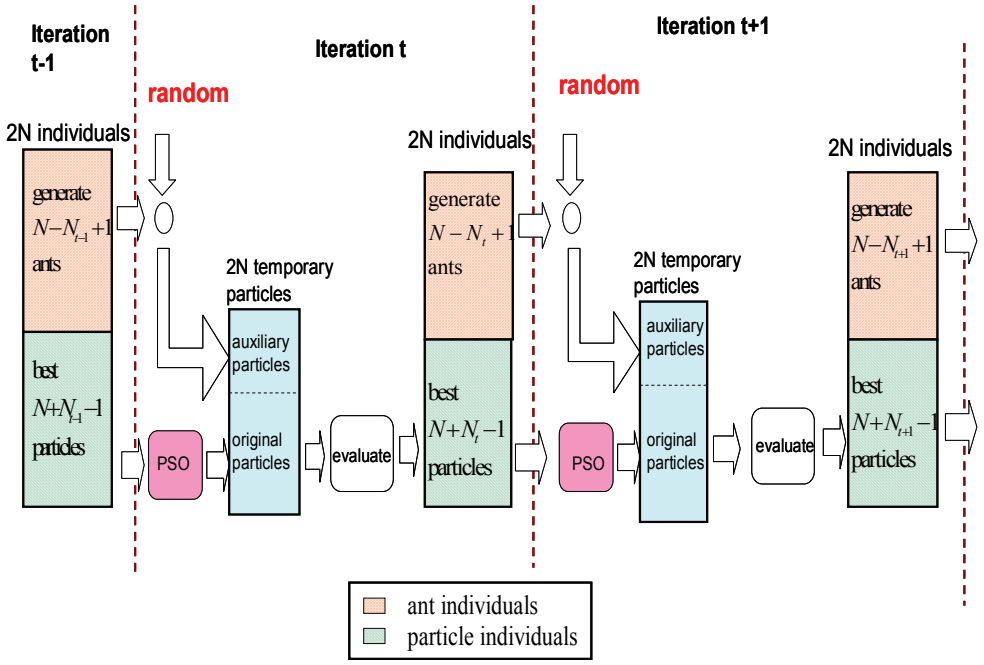

Fig. 4. Parallel combination of ACO and PSO for FS design.

In the first iteration, the rule generation algorithm in Section 2 and N different ant paths generate half of the population. The other half are generated from particle individuals. The N ant individuals contain no rules initially. New rules are generated using the criterion in (5) during the evaluation of an ant individual (FS). If the number of rules after evaluation of the N ant individuals is r_1, then the number of rules in the N particle individuals are all equal to r_1. In the parallel combination approach, the objective of using PSO is to optimize both the antecedent and consequent parameters in existing fuzzy rules; therefore, no rules are generated during the performance evaluation of a particle (FS). The initial N particle position vectors are generated using (10).

The second and subsequent iterations generate a new population with 2N new individuals. For the generation of new individuals in each iteration, the ACO-based FS design approach in Section 4 is used. In this approach, N ant paths generate N ant individuals (FSs). During the new ant individual generation process, some ants may choose the same path and generate the same individuals. This phenomenon becomes more obvious as more iterations are conducted due to pheromone matrix convergence. To consider this phenomenon, suppose that N_t ants have the same path at the tth iteration. Then only N- N_t +1 different ant

individuals are reserved in the population. The performance of these N- N_t +1 individuals is evaluated and the pheromone matrix is updated according to their performance. The remaining N- N_t +1 new individuals in the population (population size is fixed at 2N) in iteration t are partly generated by particles in the last iteration. The original N + N_{t-1} - 1 particle individuals from the last iteration are optimized by PSO. The performance of these optimized particle individuals is then evaluated. In addition to these optimized particle individuals, the N - N_{t-1} + 1 ant individuals in the previous iteration (iteration t-1) also help generate auxiliary particles to improve particle search performance. Adding these ant individuals with small random values generates auxiliary particles. The purpose of adding small random values is to distinguish auxiliary particles from existing ant individuals and to improve the algorithm's exploration ability. Suppose an original individual has the form in (10), then the auxiliary particle takes the following form

$$[m_{11}+\Delta m_{11},\sigma_{11}+\Delta\sigma_{11},\cdots,m_{1n}+\Delta m_{1n},\sigma_{1n}+\Delta\sigma_{1n},a_1+\Delta a_1,\cdots,m_{rn}+\Delta m_{rn},\sigma_{rn}+\Delta\sigma_{rn},a_r+\Delta a_r] \quad (19)$$

The performance of these N - N_{t-1} + 1 auxiliary particles is then computed. These auxiliary particles together with the original N + N_{t-1} - 1 particles constitute a total of 2N temporary particles. Only the best N + N_t - 1 particles are reserved from among these 2N particles. In the next iteration, these reserved particles cooperate to find better solutions through PSO.

For ant individual update by ACO at the end of each iteration, as in Section 4, a new ant path generates a new FS (individual) according to the pheromone levels and transition probability in (14). The pheromone levels are updated using (15) and (16). For particle individual update in iteration t, PSO updates all of the N + N_{t-1} - 1 particles generated either from auxiliary particles or original particle individuals in the previous iteration. Positions and velocities are updated based on (7) and (8) using a local version of PSO. For neighborhood best particle $\vec{p}_i^{\,g}(t)$ computation, the neighbors of a particle with r_{t-1} rules for finding $\vec{p}_i^{\,g}(t)$ are defined as the particles that also have r_{t-1} rules. As in the sequential combination approach, the learning process ends when a pre-defined number of iterations is performed.

In (Juang & Wang, 2009), the parallel combination learning approach has been applied to two control examples, nonlinear plant tracking control and reversing a truck following a circular path. These examples generate training data only when fuzzy control starts. Simulation results show that the proposed method achieves a smaller control error than ACO, PSO, and other different SI learning algorithms used for comparison in each example.

7. Conclusion

This chapter describes the design of FSs using PSO, ACO, and their sequential and parallel combination approaches. The use of on-line rule generation not only helps to determine the number of fuzzy rules, but also helps to locate the initial antecedent parameters for subsequent parameter learning using PSO. For PSO, the incorporation of ACO helps to locate a good initial fuzzy rule base for further PSO learning. For ACO, the incorporation of PSO helps to find the parameters in a continuous space. The cooperative search of ACO and PSO compensates for the searching disadvantage of each optimization method. Reported results show that the two combination approaches outperform different advanced PSO and

ACO algorithms for FS design problems. Performance of these two combination approaches on other optimization problems will be studied in the future. Other different combination approaches of ACO and PSO may also be studied for further performance improvement.

8. References

Araujo, E. & Coelho, L. d. S. (2008). Particle swarm approaches using Lozi map chaotic sequences to fuzzy modeling of an experimental thermal-vacuum system, *Applied Soft Computing*, Vol. 8, No. 4, 1354-1364.

Blum, C. & Dorigo M. (2004). The hyper-cube framework for ant colony optimization, *IEEE Trans. Syst., Man, and Cyber.-Part B: Cybernetics*, Vol. 34, No. 2, 1161-1172.

Bergh, F. van den & Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization, *IEEE Trans. Evolutionary Computation*, vol. 8, no. 3, 225-239.

Cassillas, J.; Cordon, O. & Herrera F. (2000). Learning fuzzy rules using ant colony optimization algorithms, *Proc. 2nd Workshop on Ant Algorithms – from Ant Colonies to Artificial Ants,* pp. 13-21, Brussels, Belgium.

Clerc, M. & Kennedy, J. (2002). The particle swarm - explosion, stability, and convergence in a multidimentional complex space, *IEEE Trans. Evolutionary Computation*, Vol. 6, No. 1, 58-73.

Cassillas, J.; Cordon, O.; Viana, I. F. & Herrera, F. (2005). Learning cooperative linguistic rules using the best-worst ant system algorithm, *Int. Journal of Intelligent Systems*, Vol. 20, 433-452.

Chatterjee, A.; Pulasinghe, K.; Watanabe, K. & Izumi, K. (2005). A particle-swarm-optimized fuzzy-neural network for voice-controlled robot systems, *IEEE Trans. Industrial Electronics*, Vol. 52, No. 6, 1478-1489.

Chen, X. & Li, Y. (2007). A modified PSO structure resulting in high exploration ability with convergence guaranteed, *IEEE Trans. Syst., Man, and Cyber., Part B: Cybernetics*, Vol. 37, No. 5, 1271-1289.

Dorigo, M.; Maniezzo, V. & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents, *IEEE Trans. on Syst., Man, and Cybe., Part B: Cybernetics*, Vol. 26, No. 1, 29-41.

Dorigo, M. & Gambardella, L.M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem, *IEEE Trans. on Evolutionary Computation*, Vol. 1, No. 1, 53-66.

Dorigo, M. & Caro, G. D. (1999). The ant colony optimization meta-heuristic, *New Ideas In Optimization*, D. Corne, M. Dorigo, & F. Glover Eds., pp. 11-32, London, McGraw-Hill.

Dorigo, M. & Stutzle, T. (2004). Ant Colony Optimization, MIT.

Feng, Y. J. & Feng, Z. R. (2004). An immunity-based ant system for continuous space multi-modal function optimization, *Proc. Int. Conf. Machine Learning and Cybernetics*, Vol. 2, pp. 1050-1054.

Fan, S. K. S. ; Liang, Y. C. & Zahara, E. (2004). Hybrid simplex search and particle swarm optimization for the global optimization of multimodal functions, *Engineering Optimization*, Vol. 36, No. 4, 401-418.

Ge, Y.; Meng, Q. C.; Yan, C. J. & Xu, J. (2004). A hybrid ant colony algorithm for global optimization of continuous multi-extreme functions, *Proc. Int. Conf. Machine Learning and Cybernetics*, Vol. 4, pp. 2427-2432.

Juang, C. F. (2004). A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, *IEEE Trans. Syst., Man, and Cyber., Part B: Cybernetics*, Vol. 34, No. 2, 997-1006.

Juang, C. F. & Hsu, C. H. (2005). Temperature control by chip-implemented adaptive recurrent fuzzy controller designed by evolutionary algorithm, *IEEE Trans. Circuits and Systems- I: Regular Papers*, Vol. 52, No. 11, 2376-2384.

Juang, C. F.; Chung, I. F. & Hsu, C. H. (2007). Automatic construction of feedforward /recurrent fuzzy systems by clustering-aided simplex particle swarm optimization, *Fuzzy Sets and Systems*, Vol. 158, No. 18, 1979-1996.

Juang, C. F. & Lo, C. (2007). Fuzzy systems design by clustering-aided ant colony optimization for plant control, *Int. Journal of General Systems*, Vol. 36, No. 6, 623-641.

Juang, C. F.; Lu, C. M.; Lo, C. & Wang, C. Y. (2008). Ant colony optimization algorithm for fuzzy controller design and its FPGA implementation, *IEEE Trans. Industrial Electronics*, Vol. 55, No. 3, 1453-1462.

Juang, C. F. & Lo, C. (2008). Zero-order TSK-type fuzzy system learning using a two-phase swarm intelligence, *Fuzzy Sets and Systems*, Vol. 159, No. 21, 2910-2926.

Juang, C. F. & Lu, C. M. (2009). Ant colony optimization incorporated with fuzzy Q-learning for reinforcement fuzzy control, *IEEE Trans. Syst., Man, and Cyber., Part A: Systems and Humans*, Vol. 39, No. 3, 597-608.

Juang, C. F. & Wang, C. Y. (2009). A self-generating fuzzy system with ant and particle swarm cooperative optimization, *Expert Systems with Applications*, Vol. 36, No. 3P1, 5362-5370.

Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization, *Proc. of IEEE Int. Conf. on Neural Networks*, Perth, Australia, pp. 1942-1948.

Kennedy, J.; Eberhart, R. & Shi, Y. (2001). Swarm Intelligence, Morgan Kaufmann Publisher.

Kennedy, J & Mendes, R. (2006). Neighborhood topologies in fully informed and best-of-neighborhood particle swarms, *IEEE Trans. Syst., Man, and Cyber., Part C: Applications and Reviews*, Vol. 36, No. 4, 515 – 519.

Ling, S. H. ; Iu, H. H. C. ; Chan, K. Y. ; Lam, H. K. ; Yeung, B. C. W. & Leung, F. H. (2008). Hybrid particle swarm optimization with wavelet mutation and its industrial applications, *IEEE Trans. Syst., Man, and Cyber., - Part B: Cybernetics*, Vol. 38, No. 3, 743-763.

Mucientes, M. & Casillas, J. (2007). Quick design of fuzzy controllers with good interpretability in mobile robotics, *IEEE Trans. Fuzzy Systems*, Vol. 15, No. 4, 636-651.

Parrott, D. & Li, X. (2006). Locating and tracking multiple dynamic optima by a particle swarm model using speciation, *IEEE Trans. Evolutionary Computation*, Vol.10, No.4, 440-458.

Ratnaweera, A.; Halgamuge, S. K. & Watson, H. C. (2004), Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, *IEEE Trans. Evolutionary Computation*, Vol. 8, No. 3, 240-255.

Stutzle, T. & Hoos, H. H. (2000). MAX MIN ant system, *Journal of Future Generation Computer Systems*, Vol. 8, No. 16, 889-914.

Sharma, K. D.; Chatterjee, A. & Rakshit, A. (2009). A hybrid approach for design of stable adaptive fuzzy controllers employing Lyapunov theory and particle swarm optimization, *IEEE Trans. Fuzzy Systems*, Vol. 17, No. 2, 329-342.

Fuzzy Systems

Edited by Ahmad Taher Azar

ISBN 978-953-7619-92-3
Hard cover, 216 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

While several books are available today that address the mathematical and philosophical foundations of fuzzy logic, none, unfortunately, provides the practicing knowledge engineer, system analyst, and project manager with specific, practical information about fuzzy system modeling. Those few books that include applications and case studies concentrate almost exclusively on engineering problems: pendulum balancing, truck backeruppers, cement kilns, antilock braking systems, image pattern recognition, and digital signal processing. Yet the application of fuzzy logic to engineering problems represents only a fraction of its real potential. As a method of encoding and using human knowledge in a form that is very close to the way experts think about difficult, complex problems, fuzzy systems provide the facilities necessary to break through the computational bottlenecks associated with traditional decision support and expert systems. Additionally, fuzzy systems provide a rich and robust method of building systems that include multiple conflicting, cooperating, and collaborating experts (a capability that generally eludes not only symbolic expert system users but analysts who have turned to such related technologies as neural networks and genetic algorithms). Yet the application of fuzzy logic in the areas of decision support, medical systems, database analysis and mining has been largely ignored by both the commercial vendors of decision support products and the knowledge engineers who use them.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Chia-Feng Juang (2010). Combination of Particle Swarm and Ant Colony Optimization Algorithms for Fuzzy Systems Design, Fuzzy Systems, Ahmad Taher Azar (Ed.), ISBN: 978-953-7619-92-3, InTech, Available from: http://www.intechopen.com/books/fuzzy-systems/combination-of-particle-swarm-and-ant-colony-optimization-algorithms-for-fuzzy-systems-design

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821