Chapter 6
Applications in Machine Learning

This chapter presents the fundamental concepts regarding the application of PSO on machine learning problems. The main objective in such problems is the training of computational models for performing classification and simulation tasks. It is not our intention to provide a literature review of the numerous relative applications. Instead, we aim at providing guidelines for the application and adaptation of PSO on this problem type. To achieve this, we focus on two representative cases, namely the training of artificial neural networks, and learning in fuzzy cognitive maps. In each case, the problem is first defined in a general framework, and then an illustrative example is provided to familiarize readers with the main procedures and possible obstacles that may arise during the optimization process.

INTRODUCTION

Machine learning is the field of artificial intelligence that deals with algorithms that render computational models capable of learning and adapting to their environment. From an abstract viewpoint, machine learning is the procedure of extracting information in the form of patterns or rules from data. This purpose requires the use of computational methods. Human interaction can also be beneficial within a collaborative framework to the algorithms. However, the elimination of this necessity still remains the main challenge in the development of intelligent systems.

There are different types of machine learning procedures, based on the desired outcome as well as on the degree of human intervention:

a. **Supervised learning:** The algorithm builds a mapping between a set of presented input data and a set of desired output. This is possible by altering the parameters of the computational model so that the produced error between input and output is minimized.

DOI: 10.4018/978-1-61520-666-7.ch006

b. **Unsupervised learning:** The algorithm tunes the computational model to regularities of the available data without a task-oriented measure of quality. This is possible through competition among the modules of the computational model.
c. **Semi-supervised learning:** This is a combination of the two previous approaches that employs both learning with explicit input-output examples, as well as non-labeled examples.
d. **Reinforcement learning:** The algorithm learns an input-output mapping by continuously interacting with the environment, which in turn admits an impact from every taken step, providing feedback to the model.

Several additional learning subtypes, which are outside the scope of the book at hand, can be distinguished.

Supervised learning constitutes a very prosperous application field for evolutionary algorithms and PSO, due to the existence of explicit performance measures. These measures usually come in the form of objective functions in the parameters of the computational model. Thus, training procedures aim at the detection of parameter values that minimize the model's error in learning a set of presented examples.

In this context, PSO has been applied for training artificial neural networks and fuzzy cognitive maps. The rest of this chapter is dedicated to an overview of these applications.

TRAINING ARTIFICIAL NEURAL NETWORKS WITH PSO

In the following sections, we briefly present the problem of neural network *training* for the most common case of feedforward neural networks. We also present an illustrative example for the logical XOR classification task. Further applications are also reported.

The Multi-Layer Perceptron Model

Artificial neural networks (NNs) are computational models based on the operation of biological neural networks, which constitute the information processing mechanism of the human brain. Their structure is based on the concept of the *artificial neuron*, which resembles biological neurons, as their main processing unit. The artificial neuron constitutes a nonlinear mapping between a set of input and a set of output data. Thus, if the input data are represented as a vector, $Q = (q_1, q_2, \ldots, q_m)^{\mathrm{T}}$, the artificial neuron implements a function:

$$y = F\left(b_i + \sum_{i=1}^{m} w_i q_i \right),$$

where F is the *transfer function*; w_i, $i = 1, 2, \ldots, m$, are the *weights*; and b_i is a *bias*. In order to retain a compact notation, we will henceforth represent the bias, b_i, as a weight, w_0, with an auxiliary constant input, $q_0 = 1$.

The training of the neuron to learn an input-output pair, $\{Q, y\}$, is the procedure of detecting proper weights so that the output y is obtained if Q is presented to the neuron. Obviously, this procedure can be modeled as an error minimization problem:

$$\min_{w_i}\left(y - F\left(\sum_{i=0}^{m} w_i q_i \right)\right)^2 .$$

If more than one input vectors, $Q_1, Q_2, \ldots, Q_K$, are to be learned by the neuron, then the objective function is augmented with a separate square-error term for each input vector:

$$\min_{w_i} \sum_{k=1}^{K}\left(y_k - F\left(\sum_{i=0}^{m} w_i q_{ki} \right)\right)^2 ,$$

where q_{ki} is the i-th component of the k-th input vector, $Q_k = (q_{k1}, q_{k2}, \ldots, q_{km})^{\mathrm{T}}$ (recall that w_0 is the bias of the neuron, with $q_{k0} = 1$ for all k). This kind of training procedure is also known as *batch training*, as all input vectors are presented to the neuron prior to any change of its weights.

Obviously, the dimension of the minimization problem is equal to the number, m, of the weights and biases, which is in direct correlation with the dimension of the input vector. The transfer function, $F(x)$, is selected based on the desired properties that shall be attributed to the model. Although linear transfer functions can be used, nonlinear functions equip the model with enhanced classification capabilities. Thus, nonlinear transfer functions are most commonly used, with *sigmoid* functions defined as:

$$F(x,\lambda) = \frac{1}{1+\exp(-\lambda x)},$$

being the most common choice. In addition, alternative transfer functions have been proposed in the literature (Magoulas & Vrahatis, 2006).

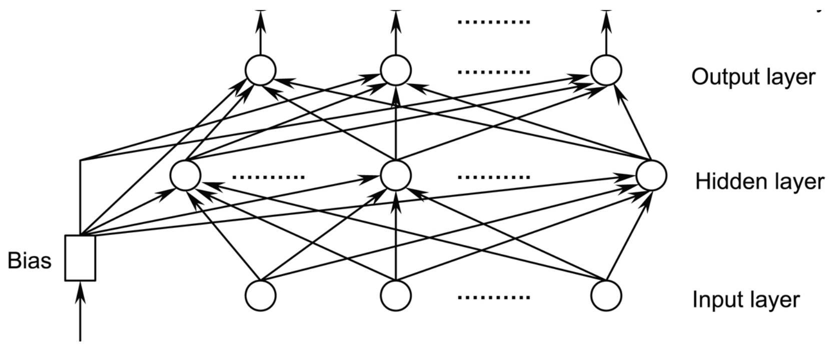
The gathering of artificial neurons, interconnected and structured in layers, produces various NN models. The most popular one is the *multi-layer perceptron*. The first layer in such a structure is called the *input layer*. Its neurons are commissioned simply to forward the input vector to the neurons of the next layer through their weighted interconnections. Thus, they do not perform any computation and, hence, do not posses biases or transfer functions. The output neurons constitute the *output layer*, while all intermediate layers are called *hidden layers*, and their neurons admit biases and transfer functions. Since each layer forwards its output strictly to the neurons of the next one, these NN models are also called *feedforward NNs*. Figure 1 illustrates the structure of such a NN architecture.

Let L be the number of layers, including the input and output layer, and let $N_1, N_2, \ldots, N_L$, be the number of neurons per layer. Then, the number of weights between the l-th and the $(l+1)$-th layer is equal to N_l $(1+N_{l+1})$, including biases. Thus, the total number of weights and biases of the whole NN is equal to:

$$n = N_L - N_1 + \sum_{l=1}^{L-1} N_l (1 + N_{l+1}). \tag{1}$$

Let $\{Q_k, y_k\}$, $k = 1, 2, \ldots, K$, be the training patterns, i.e., input-output pair examples, with $Q_k = (q_{k1}, q_{k2}, \ldots, q_{km})^{\mathrm{T}}$ and $y_k = (y_{k1}, y_{k2}, \ldots, y_{kD})^{\mathrm{T}}$. Obviously, it must hold that $N_1 = m$ and $N_L = D$. Also, let $w_{ij}^{[l]}$ denote the weight of the interconnection between the i-th neuron of the l-th layer with the j-th neuron of the $(l+1)$-th layer, with $l = 1, 2, \ldots, L$, $i = 0, 1, 2, \ldots, N_l$, and $j = 1, 2, \ldots, N_{l+1}$ (the value $i = 0$ stands for the biases of the neurons in the l-th layer for $l = 2, \ldots, L$). Then, the training of the network is equivalent to the following minimization problem (Magoulas & Vrahatis, 2006):

Figure 1. A multi-layer perceptron model



$$\min_{w_{ij}^{[l]}} \sum_{k=1}^{K} \sum_{d=1}^{D} \left(Y_{kd} - y_{kd} \right)^2, \tag{2}$$

where Y_{kd} is the actual output of the d-th output neuron when the input vector Q_k is presented to the network, while y_{kd} is the corresponding desired output. This error function is not the only possible choice for the objective function. A variety of distance functions are available in the literature, such as the *Minkowsky*, *Mahalanobis*, *Camberra*, *Chebychev*, *quadratic*, *correlation*, *Kendall's rank corellation* and *chi-square* distance metrics; the *context-similarity measure*; the *contrast model*; *hyperrectangle* distance functions and others (Magoulas & Vrahatis, 2006; Wilson & Martinez, 1997).

The minimization problem defined in equation (2) can be straightforwardly solved using PSO by encoding the network weights as particle components. Thus, each particle constitutes an individual weight configuration, evaluated in the context of equation (2). The underlying optimization problem is high-dimensional and can be highly nonlinear under sigmoid transfer functions. All PSO concepts and properties still hold in the case of NN training, since the formulation of the problem as a problem of global optimization adheres to the general optimization framework that governs all PSO developments discussed in the first part of the book at hand.

Learning the eXclusive OR: A Simple Example

Let us now illustrate the application of PSO in NN training with a simple example, namely the eXclusive OR (XOR) problem. The main goal is to train a NN to learn the logical XOR operation, which is defined with the truth table of Table 1. A properly trained NN shall admit a two dimensional input vector from the set {(0,0), (0,1), (1,0), (1,1)} and produce the correct output reported in Table 1. A NN with a single hidden layer of two neurons and a logistic transfer function are used to achieve this (Haykin, 1999, p. 175).

Setting the problem in the framework of the previous section, there will be three layers, i.e., $L = 3$. The first (input) layer consists of $N_1 = 2$ neurons, since the input vectors are 2-dimensional. The hidden layer consists of $N_2 = 2$ neurons, and the third (output) layer consists of $N_3 = 1$ neuron, since the output can be either “0” or “1”. There is a total number of 6 weights in the interconnections among all neurons, plus 3 biases of the neurons in the hidden and output layer. Hence, there is a total of 9 parameters that need to be specified, i.e., the corresponding optimization problem is 9-dimensional, as it is also derived

Table 1. Truth table of the logical XOR operation

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

from equation (1). The corresponding objective function for the specific problem can be also given in a closed-form expression, as reported in Parsopoulos and Vrahatis (2002).

PSO can now be applied using a swarm of 9-dimensional particles. Each particle corresponds to a different weight configuration of the NN, and it is evaluated based on equation (2). However, the problem is highly nonlinear with a plethora of local minima as well as flat regions. This can impose difficulties on the detection of the global minimizer to any optimization algorithm. In this case, the stretching technique of Chapter Five can be used.

Parsopoulos and Vrahatis (2002) applied the inertia weight PSO variant on this problem, using a swarm size equal to 80, while the inertia weight was decreased from 1.0 towards 0.4; $c_1 = c_2 = 0.5$; and the weights and biases were initialized randomly and uniformly in the range $[-1,1]^9$. As the global minimum of the objective function is by definition equal to zero, one can identify whether the algorithm has detected the global minimum or not. Parsopoulos and Vrahatis (2002) performed 100 experiments and, if PSO failed to converge to a global minimizer, then stretching was applied and the algorithm continued its operation on the transformed function. The obtained results revealed that stretching was able to increase efficiency of PSO from 77% to 100%, in terms of the number of successful experiments, at the cost of an increased mean number of iterations as reported in Table 2.

Alternatively to stretching, the deflection procedure, which has a more local effect on the objective function, can be used, as presented in Chapter Five. In general, our experience shows that these two techniques can be valuable tools for tackling machine learning problems with PSO.

FURTHER APPLICATIONS

In the XOR problem, as well as in the general framework presented so far in this chapter, human intervention was needed to determine crucial properties of the NN, such as the number of hidden layers and neurons. These requirements can be reduced by encoding the corresponding parameters, L, $N_2, \ldots, N_{L-1}$, as components of the particles of PSO. The number of neurons in the input and output layers, N_1 and N_L, respectively, are excluded since they depend solely on the dimension of the input and output vectors. Thus, the number of hidden layers, neurons, weights, and biases, can all be considered as parameters to be determined by PSO. In this case, special care shall be taken since the produced particles will not necessarily be of the same size.

For example, in the XOR problem, a network with 1 hidden layer and 3 hidden neurons would be modeled with a 15-dimensional particle, instead of the 9-dimensional case of 2 hidden neurons presented in the previous section. This particle would consist of the number of layers, $L = 3$, the number of hidden neurons in the hidden layer, $N_2 = 3$, and 13 weights and biases. Similarly, a network with 2 hidden layers

Table 2. Results for the XOR problem with and without stretching

Statistic	Standard PSO	Stretched cases only	PSO with stretching
Success	77%	23%	100%
Mean	1459.7	29328.6	7869.6
St.D.	1143.1	15504.2	13905.4

and 2 hidden neurons in each, would result in an 18-dimensional particle containing, L, N_2, N_3, and 15 weights and biases. Clearly, if the swarm consists of particles with different dimensions, the standard PSO update equations must be properly modified. This is possible by truncating or augmenting each particle; such approaches are reported in (Binos, 2002; Zhang *et al.*, 2000).

Moreover, instead of feedforward NN, different network types have also been also tackled with different PSO variants. For instance, Ismail and Engelbrecht (2000) used PSO to train product unit NNs, while Van den Bergh and Engelbrecht (2000) applied cooperative PSO for feedforward NNs with promising results. Al-kazemi and Mohan (2002) proposed a multi-phase PSO approach to train feedforward NNs using many swarms that combine different search criteria and hill-climbing. A hybrid approach that combines GAs with PSO for designing recurrent NNs to solve a dynamic plant control problem is proposed by Juang (2004). In this case, PSO and the genetic algorithm produce an equal portion of a population of recurrent NN designs at each iteration, while PSO additionally refines the elite individuals of the population. Finally, very interesting applications for analyzing the levels of pollution in central Hong Kong and the stage prediction of the Shing Mun river are reported in Lu *et al.* (2002) and Chau (2006), respectively.

FUZZY COGNITIVE MAPS LEARNING WITH PSO

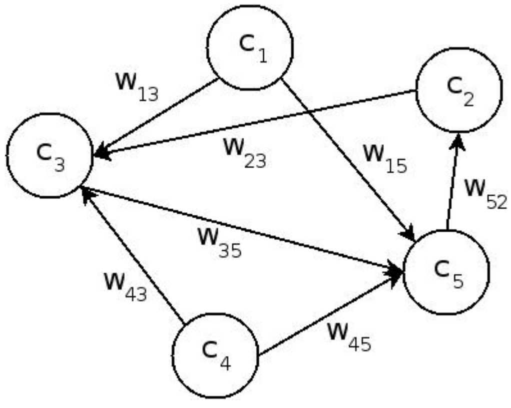
Learning in fuzzy cognitive maps constitutes an active research field with many significant applications in industry, engineering, and bioinformatics. In the following sections, we present the latest developments on the application of PSO on this machine learning task.

Fuzzy Cognitive Maps

Fuzzy cognitive maps (FCMs) are simulation models that combine concepts from artificial neural networks and fuzzy logic. The neuro-fuzzy representation equips FCMs with an inherent ability for abstraction in knowledge representation and adaptation, rendering them a very useful tool for modeling and studying complex systems. To date, FCMs have been used in a plethora of applications in diverse scientific fields, including social and organizational systems (Craiger *et al.*, 1996; Taber, 1991, 1994), circuit design (Styblinski & Meyer, 1988), industrial process control (Stylios *et al.*, 1999), supervisory control systems (Groumpos & Stylios, 2000; Stylios & Groumpos, 1998; Stylios *et al.*, 1999), and bioinformatics (Georgopoulos *et al.*, 2003; Parsopoulos *et al.*, 2004).

FCMs were originally introduced by Kosko (1986) as directional graphs with feedback. Their representation is similar to that of causal concept maps, consisting of nodes that represent key concepts

Figure 2. A simple FCM with 5 concepts (nodes) and 7 weights (arcs)



of the simulated system, and links among them that represent their causal relationships. The degree of causality between two concepts is represented with a numerical weight on their interconnecting link. A simple FCM with 5 concepts and 7 weights is depicted in Fig. 2.

To put it formally, let M be the number of concepts (nodes), C_i, $i = 1, 2,\ldots, M$, of the FCM. Each concept assumes a numerical value, $A_i \in [0,1]$, $i = 1, 2,\ldots, M$, that quantifies C_i or its effect. An edge with direction from C_i to another concept, C_j, denotes a causality relationship between them. The link is weighted with a numerical value, $w_{ij} \in [-1,1]$. Positive weights denote positive causality; hence, an increase in the value, A_i, of the concept C_i triggers an increase in the value, A_j, of C_j. On the other hand, negative causality is expressed with negative weights, and an increase in A_i results in decrease of A_j and vice versa. The weights of an FCM can be represented with a matrix:

$$W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1M} \\ w_{21} & w_{22} & \cdots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M1} & w_{M2} & \cdots & w_{MM} \end{pmatrix},$$

where the i-th row includes the causality relationships between C_i and the rest of the concepts. Zero weights, $w_{ij} = 0$, are used to represent the absence of causality (and therefore interconnection) between C_i and C_j.

The initial setting of an FCM, including both its design and initial values of weights and concepts, is determined by a group of experts with in-depth knowledge of the considered problem (Stylios *et al.*, 1999; Stylios & Groumpos, 2000). In order to avoid the error-prone procedure of assigning directly numerical values on concepts and weights, linguistic modifiers are used and then converted to fuzzy functions. Thus, numerical values are finally assigned through a fuzzification-defuzzification procedure. If needed, the experts can impose strict bounds on these values to retain their physical meaning.

After the initial configuration, the FCM behaves similarly to a discrete dynamical system. Keeping the weights fixed, the concept values are let to converge to a stable state by applying the following iterative update rule (Kosko, 1997; Stylios & Groumpos, 2004):

$$A_i(t+1) = F\left(A_i(t) + \sum_{\substack{k=1 \\ k \neq i}}^{M} w_{ki} A_k(t)\right), \tag{3}$$

where t stands for the iteration counter; $A_i(t+1)$ is the value of concept C_i at iteration $t+1$; $A_k(t)$ is the value of C_k at iteration t; and w_{ki} is the weight of the link from C_k to C_i. The function F is usually a sigmoid, similarly to the case of feedforward neural networks described in the previous sections.

After its convergence, which usually requires a small number of iterations of the equation (3), the FCM shall be capable of simulating the underlying system accurately, and desirable values shall be assumed by the concepts. Unfortunately, this is not always possible. Wide opinion variations among the experts are translated into weight values that are incapable of leading the system to desirable states. In such cases, a *learning algorithm* is needed to modify the weights further within their bounds, so that desirable steady states can be achieved.

There are just a few established learning algorithms for FCMs and they can be classified in two major categories. The first one consists of algorithms based on rules for unsupervised training of artificial neural networks (Kosko, 1997). The second category consists of evolutionary algorithms (Khan *et al.*, 2004; Koulouriotis *et al.*, 2001) and PSO-based approaches (Papageorgiou *et al.*, 2004, 2005; Parsopoulos *et al.*, 2003, 2004; Petalas *et al.*, 2007, 2009). The latter approaches are described in the following section.

A Learning Approach Based on PSO

The learning procedure is similar to some extent with that of neural network training. Let M be the number of concepts, C_i, and $A_i \in [0,1]$, $i = 1, 2,\ldots, M$, be their values. Let also, $M^* \leq M$, be the number of concepts whose output is of interest, i.e., their values are crucial for the operation of the simulated system. These concepts are called *output concepts*, and we denote them as $C_{o1}, C_{o2},\ldots, C_{oM^*}$. Their values, $A_{o1}, A_{o2},\ldots, A_{oM^*}$, are monitored and used as performance criterion for the learning procedure. Hence, the user is interested in detecting a weight matrix, $W = [w_{ij}]$, $i, j = 1, 2,\ldots, M$, so that the converged FCM attains a desirable steady state, while its weights retain their physical meaning.

A desirable steady state for a given weight matrix, W, shall provide output concept values that lie within prespecified bounds:

$$A_{oi}^{[\min]} \leq A_{oi} \leq A_{oi}^{[\max]}, \quad i = 1, 2,\ldots, M^*,$$

considered to be crucial for the proper operation of the simulated system. A proper objective function that guarantees this property shall be defined for the learning procedure. Papageorgiou *et al.* (2005) proposed the following objective function:

$$f(W) = \sum_{i=1}^{M^*}\left[\left|A_{oi}^{[\min]} - A_{oi}\right| \times H\left(A_{oi}^{[\min]} - A_{oi}\right)\right] + \sum_{i=1}^{M^*}\left[\left|A_{oi} - A_{oi}^{[\max]}\right| \times H\left(A_{oi} - A_{oi}^{[\max]}\right)\right], \tag{4}$$

where H is the Heaviside function:

$$H\left(A_{oi}^{[\min]} - A_{oi}\right) = \begin{cases} 0, & \text{if } A_{oi}^{[\min]} - A_{oi} < 0, \\ 1, & \text{otherwise,} \end{cases} \quad \text{and} \quad H\left(A_{oi} - A_{oi}^{[\max]}\right) = \begin{cases} 0, & \text{if } A_{oi} - A_{oi}^{[\max]} < 0, \\ 1, & \text{otherwise,} \end{cases}$$

and A_{oi}, $i = 1, 2, \ldots, M^*$, are the steady state values of output concepts obtained by applying the iterative rule of equation (3), using the weight matrix W.

The objective function of equation (4) is actually a penalty function. Its global minimizers are weight matrices that produce output concept values within the prespecified bounds. Any other weight matrix is penalized by an amount proportional to the degree of bound violation. In contrast to gradient-based approaches, the non-differentiability of $f(W)$ does not constitute an obstacle to the application of PSO. Possible further requirements implied by the problem at hand can be easily incorporated in equation (4) to achieve a desirable weight matrix.

PSO is applied straightforwardly using the objective function of equation (4). Each particle of the swarm is a weight matrix, encoded as a vector by considering its rows in turn:

$$W = [\underbrace{w_{12}, \ldots, w_{1M}}_{\text{row 1}}, \underbrace{w_{21}, \ldots, w_{2M}}_{\text{row 2}}, \ldots, \underbrace{w_{M1}, \ldots, w_{M,M-1}}_{\text{row } M}]^{\mathrm{T}}.$$

The elements, w_{11}, $w_{22}, \ldots, w_{MM}$, of the main diagonal of W are omitted in the vectorial representation, as, by definition, FCMs have no self-feedback in their nodes, and therefore the corresponding weights are all equal to zero, $w_{ii} = 0$, $i = 1, 2, \ldots, M$. The dimension of particles for an FCM with M concepts will be at maximum equal to $M\times(M-1)$, although in many applications it is significantly smaller due to sparsity of the weight matrices.

Each particle (weight matrix W) is evaluated with $f(W)$, by using the weight matrix W in the FCM and letting it converge to a steady state to obtain the required concept values involved in equation (4). Subsequently, the PSO update equations are used to produce new weight matrices (particles). Initialization of each weight is performed randomly and uniformly either in [-1,0], if it is specified as negative by the experts, or in [0,1], otherwise. Obviously, there is no restriction regarding the employed PSO variant, while the objective function can be modified accordingly to fit the framework of different problems. A flowchart of the proposed PSO-based learning procedure is depicted in Fig. 3. In the next section, an illustrative example of the learning procedure is provided for an industrial process control problem.

Industrial Process Control: An Illustrative Example

This problem was addressed by Papageorgiou *et al.* (2005), and it is ideal to illustrate the PSO-based learning procedure. The problem, previously considered by Stylios and Groumpos (1998), consists of the simulation of a simple process control problem from industry. The system, which is illustrated in Fig. 4, consists of a tank and three valves, denoted as V1, V2, and V3, which control the amount of liquid in the tank. Valves V1 and V2 pour two liquid chemicals, whose chemical reaction produces a new liquid, into the tank. A sensor (gauger) is sunk into the tank and gauges the specific gravity, G, of the produced liquid. If G attains a value within a desirable range, [$G_{\min}$, $G_{\max}$], then V3 opens and empties the tank. There is also a security limit on the height, T, of the liquid in the tank, which shall not exceed lower and upper bounds, $T_{\min}$ and $T_{\max}$, respectively. Therefore, the main goal of this simple control process is the preservation of G and T within the desirable limits:

Figure 3. Flowchart of the PSO-based learning procedure

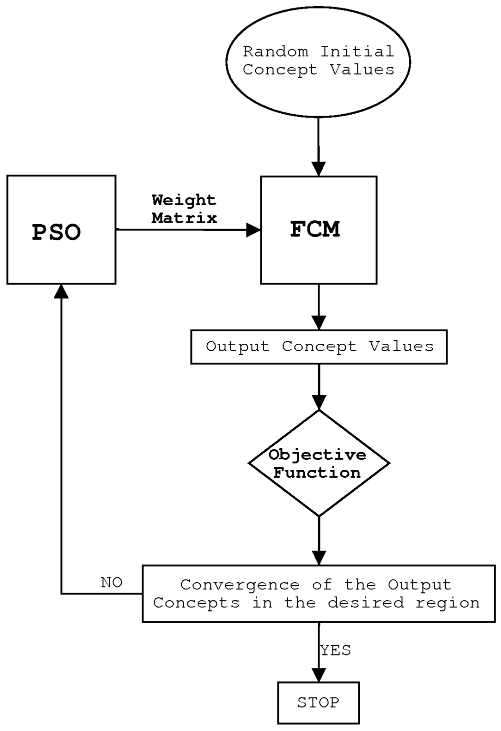
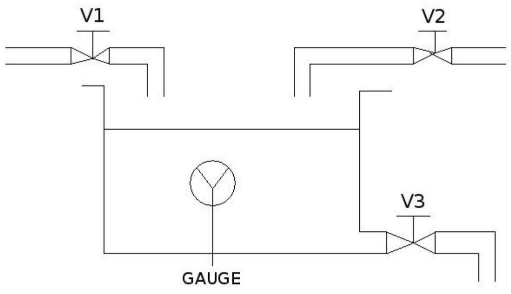

Random Initial Concept Values
PSO
Weight Matrix
FCM
Output Concept Values
Objective Function
NO
Convergence of the Output Concepts in the desired region
YES
STOP

Figure 4. The industrial process control system

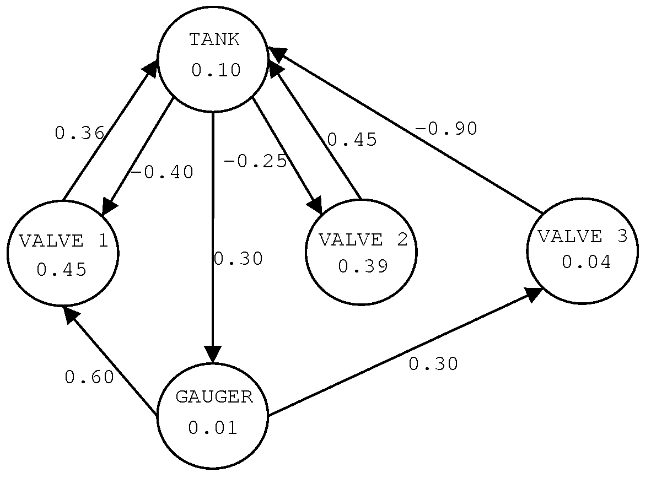


$G_{\min} \leq G \leq G_{\max}, \quad T_{\min} \leq T \leq T_{\max}.$

A group of experts designed an FCM that simulates the system, following the procedure described in the previous sections. First, they decided on the number of concepts and their interactions, which are described as "negative", "positive", and "no influence", based on the causality relationships among them. The corresponding FCM is depicted in Fig. 5 and consists of the following five concepts:

1. **Concept C_1:** Height of liquid in tank. It depends on the state of valves V1, V2, and V3.
2. **Concept C_2:** State of valve V1 (open, closed, or partially open).

Figure 5. The FCM that simulates the system of Fig. 4



3. **Concept C_3:** State of valve V2 (open, closed, or partially open).
4. **Concept C_4:** State of valve V3 (open, closed, or partially open).
5. **Concept C_5:** Specific gravity of the produced liquid.

The concepts are connected with eight links. A consensus among experts was attained regarding the directions of the links. Also, the initial weights were determined by assigning linguistic variables, such as "weak", "strong" etc., along with the corresponding fuzzy sets, as defined in Cox (1999). The linguistic variables are combined to a single linguistic weight using the widely used SUM technique (Lin & Lee, 1996). This weight is transformed to a numerical value using the *center of area* defuzzification method (Kosko, 1992; Lin & Lee, 1996).

All experts agreed on the same range for the weights w_{21}, w_{31} and w_{41}, while most of them agreed for w_{12} and w_{13} (Papageorgiou *et al.*, 2005). However, no agreement was attained for w_{15}, w_{52}, and w_{54}, where their opinions varied significantly. The ranges as implied by the fuzzy regions of the weights are reported in Table 3.

PSO was applied on the corresponding 8-dimensional optimization problem for the detection of a weight setting that retains the output concepts, C_1 and C_5, within the following bounds assigned by the experts:

$$0.68 \leq A_1 \leq 0.70, \quad 0.78 \leq A_5 \leq 0.85. \tag{5}$$

Thus, the objective function of equation (4) becomes:

$$\begin{aligned} f(W) = {} & \left|0.68 - A_1\right| H\left(0.68 - A_1\right) + \left|A_1 - 0.70\right| H\left(A_1 - 0.70\right) + \\ & \left|0.78 - A_5\right| H\left(0.78 - A_5\right) + \left|A_5 - 0.85\right| H\left(A_5 - 0.85\right). \end{aligned}$$

Table 3. Ranges of the weights as implied by their fuzzy regions for the industrial process control problem of Fig. 4

$-0.5 \le w_{12} \le -0.3$,	$0.2 \le w_{15} \le 0.4$,	$0.4 \le w_{31} \le 0.5$,	$0.5 \le w_{52} \le 0.7$,
$-0.4 \le w_{13} \le -0.2$,	$0.3 \le w_{21} \le 0.4$,	$-1.0 \le w_{41} \le -0.8$,	$0.2 \le w_{54} \le 0.4$,

Additionally, each weight was constrained in the range [-1,0] or [0,1] to avoid physically meaningless weights.

Papageorgiou *et al.* (2005) considered two different scenarios. In the first, they admitted all constraints posed by the experts. In the second, they considered only those constraints where a unanimous agreement among the experts was achieved. In both cases they used the constriction coefficient PSO variant with ring neighborhood topology of radius $r = 3$, and a swarm size equal to 20. Also, the default parameters, $\chi = 0.729$, $c_1 = c_2 = 2.05$, were used, while the desired accuracy on the objective function value was set to 10^{-8}. For each case, 100 independent experiments were performed and their results were statistically analyzed. The most important observations and conclusions are reported in the following sections.

Scenario One: All Constraints are Retained

In the first set of experiments, all constraints defined by the experts in Table 3 were used. However, as reported in Papageorgiou *et al.* (2005), no solution that fulfills the relations (5) was found in 100 experiments. This is a strong indication that the provided weight bounds are not proper and, consequently, they cannot lead the FCM to a desirable steady state. The best of the obtained weight matrices was the following:

$$W^* = \begin{pmatrix} 0.00 & -0.35 & -0.20 & 0.00 & 0.40 \\ 0.40 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.50 & 0.00 & 0.00 & 0.00 & 0.00 \\ -0.80 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.75 & 0.00 & 0.20 & 0.00 \end{pmatrix},$$

which leads the FCM to the following steady state:

$$A_1 = 0.6723, \quad A_2 = 0.7417, \quad A_3 = 0.6188, \quad A_4 = 0.6997, \quad A_5 = 0.7311.$$

This steady state clearly violates both constraints in relation (5).

Assuming that the constraints for the weights w_{15}, w_{52}, and w_{54}, for which the opinions of experts varied significantly, were the reason for the disability of the algorithm to achieve a solution, Papageorgiou *et al.* (2005) attempted to solve the problem by omitting them. Thus, they experimented by omitting the constraints, initially one by one, and subsequently in pairs. Despite their effort, again, no solution was detected.

Finally, they omitted all three constraints, allowing the three weights to assume values in the whole range [0,1] (they all have positive signs). In this case, proper weight matrices were obtained, although in substantially different ranges than those determined by the experts (Papageorgiou *et al.*, 2005). Thus, the algorithm was able to correct the inconsistencies of the experts in a very efficient manner, requiring at most 620 function evaluations to find a solution with the desirable accuracy. Of course, due to the interactions among weights and concepts, there is a multitude of different optimal matrices. Indicatively, we report one of the obtained optimal matrices:

$$W^* = \begin{pmatrix} 0.00 & -0.45 & -0.20 & 0.00 & 0.84 \\ 0.40 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.50 & 0.00 & 0.00 & 0.00 & 0.00 \\ -0.80 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.99 & 0.00 & 0.10 & 0.00 \end{pmatrix},$$

which leads the FCM to the desirable steady state:

$A_1 = 0.6805, \quad A_2 = 0.7798, \quad A_3 = 0.6176, \quad A_4 = 0.6816, \quad A_5 = 0.7967.$

Scenario Two: Only Unanimously Agreed Constraints are Retained

In the second scenario, Papageorgiou *et al.* (2005) retained only the unanimously agreed constraints, i.e., those of the weights w_{21}, w_{31}, and w_{41}. The rest of the weights were let to move unrestricted in [-1,0] or [0,1], depending on their sign.

In this case, it was observed that the three constrained weights assumed values in remarkably narrower ranges than those proposed by the experts. This implies that PSO can be also used for the further refinement of the assigned bounds. Numerous experiments were performed with the three weights being fixed or moving in their corresponding ranges (Papageorgiou *et al.*, 2005). Some of the obtained optimal weight matrices were the following:

$$W^* = \begin{pmatrix} 0.00 & -0.44 & -0.10 & 0.00 & 1.00 \\ 0.40 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.50 & 0.00 & 0.00 & 0.00 & 0.00 \\ -0.81 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.13 & 0.00 \end{pmatrix},$$

which leads the FCM to the steady state:

$A_1 = 0.6805, \quad A_2 = 0.7872, \quad A_3 = 0.6390, \quad A_4 = 0.6898, \quad A_5 = 0.8172;$

$$W^* = \begin{pmatrix} 0.00 & -0.27 & -0.20 & 0.00 & 1.00 \\ 0.40 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.50 & 0.00 & 0.00 & 0.00 & 0.00 \\ -0.81 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.10 & 0.00 \end{pmatrix},$$

which leads the FCM to the steady state:

$A_1 = 0.6816, \quad A_2 = 0.8090, \quad A_3 = 0.6174, \quad A_4 = 0.6822, \quad A_5 = 0.8174;$

and

$$W^* = \begin{pmatrix} 0.00 & -0.23 & -0.13 & 0.00 & 0.86 \\ 0.40 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.50 & 0.00 & 0.00 & 0.00 & 0.00 \\ -0.81 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.92 & 0.00 & 0.12 & 0.00 \end{pmatrix},$$

which leads the FCM to the steady state:

$A_1 = 0.6817, \quad A_2 = 0.7985, \quad A_3 = 0.6323, \quad A_4 = 0.6860, \quad A_5 = 0.8007.$

Although different, all the obtained weight matrices lead the FCM to desirably steady states. The weights w_{13}, w_{15}, w_{52}, and w_{54}, converged to regions significantly different than those suggested by the experts, while w_{21}, w_{31}, and w_{41}, remained almost fixed at values close to the initial ones suggested by the experts. Also, the weight w_{12} deviates slightly from its initial region.

These interesting observations constitute a small fraction of the information derived from the application of the PSO-based learning procedure on the industrial process control problem. However it suffices to reveal the potential of the PSO-based learning algorithm to significantly increase our intuition on the problem dynamics, and help towards the further refinement of expert knowledge. Moreover, it can provide robust solutions in cases where the experts disagree or doubt on their decisions. The next section briefly reports further developments and applications of PSO-based learning procedures.

Further Developments on PSO-Based Learning Algorithms

The PSO-based learning procedure was introduced in Parsopoulos *et al.* (2003) and further evaluated in Papageorgiou *et al.* (2005) for the industrial process control problem presented in the previous section. Its efficiency and apparent practical value prompted its further use in different applications, such as modeling of radiation therapy systems (Parsopoulos *et al.*, 2004), and more complex industrial process control problems (Papageorgiou *et al.*, 2004).

Concurrently, different PSO variants were assessed on the FCM learning problem. One of the most efficient approaches was recently proposed by Petalas *et al.* (2007, 2009). This approach utilizes the memetic PSO (MPSO) algorithm presented in Chapter Four. More specifically, MPSO approaches, equipped with either the Hook and Jeeves or the Solis and Wets local search, were investigated on interesting FCM learning problems, including a complex industrial process control problem, the simulation of a radiation therapy system, a heat exchanger problem, and the simulation of an ecological industrial park problem (Petalas *et al.*, 2009). Moreover, MPSO was compared favorably against the DE and GA algorithm on these problems. The reader is referred to the original paper for the complete presentation of these applications. Besides the form of the equation (4), different objective functions can be used, depending on the problem at hand. In the following, we briefly describe the radiation therapy problem, where such an objective function is used.

Radiotherapy is a popular means of cancer treatment. It is a complex process that involves a large number of treatment variables. The main objective of radiotherapy is the delivery of the highest possible amount of radiation to the tumor, while minimizing the exposure of healthy tissue and critical organs to the radiation. Hence, treatment planning and doctor-computer interaction is required prior to the actual final treatment (Parsopoulos *et al.*, 2004; Petalas *et al.*, 2009).

The radiation therapy process is modeled by a supervisor-FCM, which is constructed by experts and consists of the following six concepts:

1. **Concept C_1:** Tumor localization.
2. **Concept C_2:** Dose prescribed for the treatment planning.
3. **Concept C_3:** Machine-related factors.
4. **Concept C_4:** Human-related factors.
5. **Concept C_5:** Patient positioning and immobilization.
6. **Concept C_6:** Final dose received by the targeted tumor.

The corresponding FCM is depicted in Fig. 6.

The main objective in this problem is the maximization of the final dose received by the tumor, which is described by C_6, as well as the maximization of the dose, C_2, prescribed by the treatment planning as defined by the AAPM and ICRP protocols for the determination of acceptable dose per organ and part of the human body (Khan, 1994; Wells & Niederer, 1998; Willoughby *et al.*, 1996). Thus, the objective function is modeled so that the values of the two aforementioned concepts are maximized, instead of restricting them in bounds as in equation (4), and it is defined as:

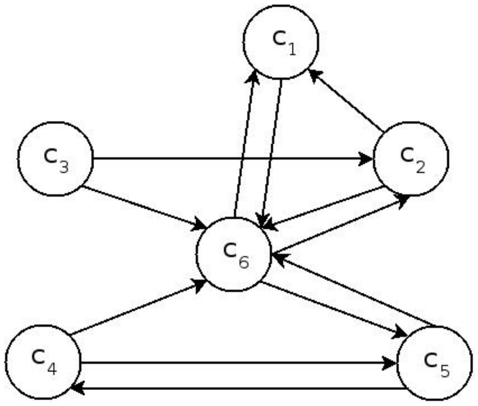
$$f(W) = -A_2 - A_6,$$

where the negative sign is used to transform the maximization of the positive values A_2 and A_6 to an equivalent minimization problem.

This objective function was easily addressed with PSO as reported in Parsopoulos *et al.* (2004). In their experiments, both the inertia weight and the constriction coefficient PSO variant were investigated and compared against two DE variants. The standard parameter setting, $\chi = 0.729$, $c_1 = c_2 = 2.05$, and a decreasing inertia weight from 1.2 to 0.1 were employed, while the desirable bounds for the output concepts were determined by the experts as follows:

$$0.80 \leq A_2 \leq 0.95,\ 0.90 \leq A_6 \leq 0.95.$$

Figure 6. The supervisor-FCM for the radiotherapy problem



All algorithms detected the same solution matrix (Parsopoulos *et al.*, 2004):

$$W^* = \begin{pmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 \\ 0.4 & 0.0 & 0.0 & 0.0 & 0.0 & 0.7 \\ 0.0 & -0.2 & 0.0 & 0.0 & 0.0 & -0.1 \\ 0.0 & 0.0 & 0.0 & 0.0 & -0.2 & -0.2 \\ 0.0 & 0.0 & 0.0 & -0.6 & 0.0 & 0.8 \\ 0.4 & 0.9 & 0.0 & 0.0 & 0.9 & 0.0 \end{pmatrix},$$

which leads the FCM to the steady state:

$A_1 = 0.819643$, $A_2 = 0.819398$, $A_3 = 0.659046$, $A_4 = 0.501709$, $A_5 = 0.824788$, $A_6 = 0.916315$.

An interesting point is the high positive influence of concept C_6 (final dose) on concept C_2 (dose prescribed from the treatment planning), as well as on concept C_5 (patient positioning). This means that if we succeed in delivering the maximum dose to the target volume, then the initial calculated dose from treatment planning is the desired; the same happens with patient positioning (Parsopoulos *et al.*, 2004).

The convenient manipulation and easy adaptation of the PSO-based learning scheme to different problems renders it a valuable tool in machine learning tasks. Using these developments as a starting point, new simulation models were also introduced. Such a model is the *interval cognitive map* introduced by Petalas *et al.* (2005), which is based on modeling the concepts and weights with interval numbers. The presentation of the interval arithmetic methodology for learning in interval FCM models is outside the scope of the book at hand, and thus omitted.

CHAPTER SYNOPSIS

Fundamental concepts of applying PSO on machine learning problems were presented and analyzed. The problems were restricted to the representative cases of training feedforward artificial neural networks and learning in fuzzy cognitive maps. For each case, an illustrative example was used to introduce the reader to the workings of the corresponding PSO-based approach. Thus, the training of a feedforward neural network with one hidden layer for learning the XOR logical operation, as well as the simulation of an industrial process control problem with a fuzzy cognitive map, were analyzed. Moreover, elements for further acquisition were provided in the reported literature, along with brief references to relative applications.

REFERENCES

Al-kazemi, B., & Mohan, C. K. (2002). Training feedforward neural networks using multi-phase particle swarm optimization. In L. Wang, J.C. Rajapakse, K. Fukushima, S.-Y. Lee & X. Yao (Eds.), *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'02), Singapore* (pp. 2615-2619).

Binos, T. (2002). *Evolving neural network architecture and weights using an evolutionary algorithm*. MSc thesis, RMIT University, Melbourne, Australia.

Chau, K. W. (2006). Particle swarm optimization training algorithm for ANNs in stage prediction of Shing Mun river. *Journal of Hydrology (Amsterdam)*, *329*, 363–367. doi:10.1016/j.jhydrol.2006.02.025

Cox, E. (1999). *The fuzzy systems handbook*. Cambridge, MA: Academic Press.

Craiger, J. P., Goodman, D. F., Weiss, R. J., & Butler, A. (1996). Modeling organizational behavior with fuzzy cognitive maps. *International Journal of Computation Intelligent Organization*, *1*, 120–123.

Georgopoulos, V., Malandraki, G., & Stylios, C. (2003). A fuzzy cognitive map approach to differential diagnosis of specific language impairment. *Journal of Artificial Intelligence in Medicine*, *29*(3), 261–278. doi:10.1016/S0933-3657(02)00076-3

Groumpos, P. P., & Stylios, C. D. (2000). Modelling supervisory control systems using fuzzy cognitive maps. *Chaos, Solitons, and Fractals*, *11*, 329–336. doi:10.1016/S0960-0779(98)00303-8

Haykin, S. (1999). *Neural networks: a comprehensive foundation*. Upper Saddle River, NJ: Prentice Hall.

Ismail, A., & Engelbrecht, A. P. (2000). Global optimization algorithms for training product unit neural networks. In *Proceedings of the 2000 IEEE International Joint Conference on Neural Networks (IJCNN'00), Como, Italy* (pp. 132-137).

Juang, C.-F. (2004). A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics*, *34*(2), 997–1006. doi:10.1109/TSMCB.2003.818557

Khan, F. (1994). *The physics of radiation therapy*. Baltimore: Williams & Wilkins.

Khan, M. S., Khor, S., & Chong, A. (2004). Fuzzy cognitive maps with genetic algorithm for goal-oriented decision support. *International Journal of Uncertainty . Fuzziness and Knowledge-Based Systems*, *12*, 31–42. doi:10.1142/S0218488504003028

Kosko, B. (1986). Fuzzy cognitive maps. *International Journal of Man-Machine Studies*, *24*, 65–75. doi:10.1016/S0020-7373(86)80040-2

Kosko, B. (1992). *Neural networks and fuzzy systems*. Upper Saddle River, NJ: Prentice Hall.

Kosko, B. (1997). *Fuzzy engineering*. Upper Saddle River, NJ: Prentice Hall.

Koulouriotis, D. E., Diakoulakis, I. E., & Emiris, D. M. (2001). Learning fuzzy cognitive maps using evolution strategies: a novel schema for modeling and simulating high-level behavior. In *Proceedings of the 2001 IEEE Congress on Evolutionary Computation (CEC'01), Seoul, Korea* (pp. 364-371).

Lin, C. T., & Lee, C. S. (1996). *Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems*. Upper Saddle River, NJ: Prentice Hall.

Lu, W. Z., Fan, H. Y., Leung, A. Y. T., & Wong, J. C. K. (2002). Analysis of pollutant levels in central Hong Kong applying neural network method with particle swarm optimization. *Environmental Monitoring and Assessment*, *79*, 217–230. doi:10.1023/A:1020274409612

Magoulas, G. D., & Vrahatis, M. N. (2006). Adaptive algorithms for neural network supervised learning: a deterministic optimization approach. *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, *16*(7), 1929–1950. doi:10.1142/S0218127406015805

Papageorgiou, E. I., Parsopoulos, K. E., Groumpos, P. P., & Vrahatis, M. N. (2004). Fuzzy cognitive maps learning through swarm intelligence. In *Lecture Notes in Artificial Intelligence (LNAI), Vol. 3070* (pp. 344-349). Berlin: Springer.

Papageorgiou, E. I., Parsopoulos, K. E., Stylios, C. D., Groumpos, P. P., & Vrahatis, M. N. (2005). Fuzzy cognitive maps learning using particle swarm optimization. *Journal of Intelligent Information Systems*, *25*(1), 95–121. doi:10.1007/s10844-005-0864-9

Parsopoulos, K. E., Papageorgiou, E. I., Groumpos, P. P., & Vrahatis, M. N. (2003). A first study of fuzzy cognitive maps learning using particle swarm optimization. In *Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC'03), Canberra, Australia* (pp. 1440-1447).

Parsopoulos, K. E., Papageorgiou, E. I., Groumpos, P. P., & Vrahatis, M. N. (2004). Evolutionary computation techniques for optimizing fuzzy cognitive maps in radiation therapy systems. In [LNCS]. *Lecture Notes in Computer Science*, *3102*, 402–413.

Parsopoulos, K. E., & Vrahatis, M. N. (2002). Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, *1*(2-3), 235–306. doi:10.1023/A:1016568309421

Petalas, Y. G., Papageorgiou, E. I., Parsopoulos, K. E., Groumpos, P. P., & Vrahatis, M. N. (2005). Interval cognitive maps. In *Proceedings of the International Conference of Numerical Analysis and Applied Mathematics (ICNAAM 2005), Rhodes, Greece* (pp. 882-885).

Petalas, Y. G., Parsopoulos, K. E., Papageorgiou, E. I., Groumpos, P. P., & Vrahatis, M. N. (2007). Enhanced learning in fuzzy simulation models using memetic particle swarm optimization. In *Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS'07)* (pp. 16-22).:Washington, D.C.: IEEE.

Petalas, Y. G., Parsopoulos, K. E., & Vrahatis, M. N. (2009). Improving fuzzy cognitive maps learning through memetic particle swarm optimization. *Soft Computing*, *13*(1), 77–94. doi:10.1007/s00500-008-0311-2

Styblinski, M. A., & Meyer, B. D. (1988). Fuzzy cognitive maps, signal flow graphs, and qualitative circuit analysis. In *Proceedings of 2nd IEEE International Conference on Neural Networks, San Diego (CA), USA* (pp. 549-556).

Stylios, C. D., Georgopoulos, V., & Groumpos, P. P. (1999). Fuzzy cognitive map approach to process control systems. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, *3*(5), 409–417.

Stylios, C. D., & Groumpos, P. P. (1998). The challenge of modelling supervisory systems using fuzzy cognitive maps. *Journal of Intelligent Manufacturing*, *9*, 339–345. doi:10.1023/A:1008978809938

Stylios, C. D., & Groumpos, P. P. (2000). Fuzzy cognitive maps in modeling supervisory control systems. *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology*, *8*(2), 83–98.

Stylios, C. D., & Groumpos, P. P. (2004). Modeling complex systems using fuzzy cognitive maps. *IEEE Transactions on Systems . Man and Cybernetics: Part A*, *34*(1), 159–165.

Taber, R. (1991). Knowledge processing with fuzzy cognitive maps. *Expert Systems with Applications*, *2*, 83–87. doi:10.1016/0957-4174(91)90136-3

Taber, R. (1994). Fuzzy cognitive maps model social systems. *AI Expert*, *9*, 8–23.

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

Wells, D., & Niederer, J. (1998). A medical expert system approach using artificial neural networks for standardized treatment planning. *International Journal of Radiation Oncology, Biology, Physics*, *41*(1), 173–182. doi:10.1016/S0360-3016(98)00035-2

Willoughby, T., Starkschall, G., Janjan, N., & Rosen, I. (1996). Evaluation and scoring of radiotherapy treatment plans using an artificial neural network. *International Journal of Radiation Oncology, Biology, Physics*, *34*(4), 923–930. doi:10.1016/0360-3016(95)02120-5

Wilson, D. R., & Martinez, T. R. (1997). Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, *6*, 1–34.

Zhang, C., Shao, H., & Li, Y. (2000). Particle swarm optimisation for evolving artificial neural network. In *Proceedings of the 2000 IEEE International Conference on Systems, Man, Cybernetics, Nashville (TN), USA* (pp. 2487-2490).