Chapter 9

Heuristic Search Applied to Fuzzy Cognitive Maps Learning

Bruno Augusto Angélico, Márcio Mendonça,
Lúcia Valéria R. de Arruda and Taufik Abrão

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/52918

1. Introduction

Fuzzy Cognitive Maps were initially proposed by Kosko [1–3], as an extension of cognitive maps proposed by Axelrod [4]. FCM is a graph used for representing causal relationships among concepts that stand for the states and variables of the system, emulating the cognitive knowledge of experts on a specific area. FCM can be interpreted as a combination of Fuzzy Logic and Neural Networks, because it combines the sense rules of Fuzzy Logic with the learning of the Neural Networks. A FCM describes the behavior of a knowledge based system in terms of concepts, where each concept represents an entity, a state, a variable, or a characteristic of the system. The human knowledge and experience about the system determines the type and the number of the nodes as well as the initial conditions of the FCM.

FCM has been considered with great research interest in many scientific fields, such as political decision [5], medical decision [6, 7], industrial process control [8, 9], artificial life [10], social systems [11], corporative decision, policy analysis [12], among others.

The knowledge of the experts firstly defines the influence of a concept on the other, determining the causality relationships. Then, the concept values are qualitatively obtained by linguistic terms, such as strong, weak, null, and so on. These linguistic variables are transformed in numerical values using a *defuzzification* method, for instance, the center of gravity scheme described in [2].

Hence, in general, experts develop a FCM identifying key concepts, defining the causal relationships among the concepts, and estimating the strength of these relationships. However, when the experts are not able to express the causal relationships or they substantially diverge in opinion about it, data driven methods for learning FCMs may be necessary.

Particularly, this Chapter focuses on the FCM learning using three different population based metaheuristics: particle swarm optimization (PSO), genetic algorithm (GA) and differential evolution (DE). Two process control problems described in [8] and [9] are considered in this work. A complete convergence analysis of the PSO, GA and DE is carried out considering 10000 realizations of each algorithm in every scenario of the studied processes.

The rest of the Chapter has the following organization: Section 2 briefly describes the FCM modeling and the processes to be controlled. Section 3 considers the PSO, GA and the DE approaching for FCM learning, while Section 4 shows the simulation results. Lastly, Section 5 points out the main conclusions.

2. FCM modeling in control processes

In FCMs, concepts (nodes) are utilized to represent different aspects and behavior of the system. The system dynamics are simulated by the interaction of concepts. The concept C_i, $i = 1, 2, \ldots, N$ is characterized by a value $A_i \in [0, 1]$.

Concepts are interconnected concerning the underlying causal relationships amongst factors, characteristics, and components that constitute the system. Each interconnection between two concepts, C_i and C_j, has a weight, $W_{i,j}$, which is numerically represented by the strength of the causal relationships between C_i and C_j. The sign of $W_{i,j}$ indicates whether the concept C_i causes the concept C_j or vice versa. Hence, if:

$$\begin{cases} W_{i,j} > 0, \text{ positive causality} \\ W_{i,j} < 0, \text{ negative causality} \\ W_{i,j} = 0, \quad \text{no relation} \end{cases}$$

The number of concepts and the initial weights of the FCM are determined by human knowledge and experience. The numerical values, A_i, of each concept is a transformation of the fuzzy values assigned by the experts. The FCM converges to a steady state (limit cycle) according to the scheme proposed in [3]:

$$A_i(k+1) = f\left(A_i(k) + \sum_{\substack{j=1 \\ j \neq i}}^{N} W_{ji} A_j(k)\right), \tag{1}$$

where k is the interaction index and $f(\cdot)$ is the sigmoid function

$$f(x) = \frac{1}{1 + e^{-\lambda x}}, \tag{2}$$

that guarantees the values $A_i \in [0, 1]$. $\lambda > 0$ is a parameter representing the learning memory. In this work, $\lambda = 1$ has been adopted.

http://dx.doi.org/10.5772/52918

2.1. First control system (PROC1)

A simple chemical process frequently considered in literature [8, 13, 14], is initially selected for illustrating the need of a FCM learning technique. Figure 1 represents the process (PROC1) consisting of one tank and three valves that control the liquid level in the tank. Valves V_1 and V_2 fill the tank with different liquids. A chemical reaction takes place into the tank producing a new liquid that leaves the recipient by valve V_3. A sensor (gauger) measures the specific gravity of the resultant mixture.

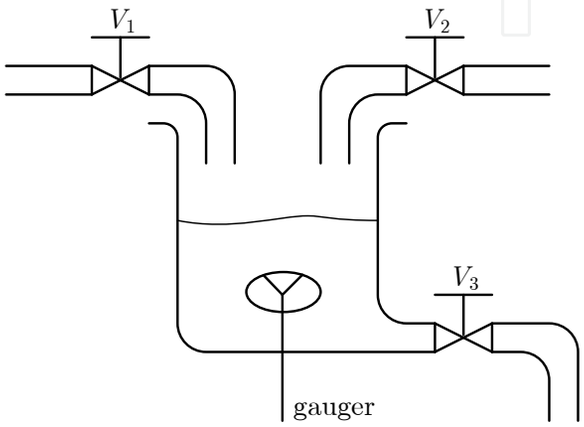


Figure 1. PROC1: a chemical process control problem described in [8].

When the value of the specific gravity, G, is in the range $[G_{\min}, G_{\max}]$, the desired liquid has been produced. The height of the liquid inside, H, must lie in the range $[H_{\min}, H_{\max}]$. The controller has to keep G and H within their bounds, i.e.,

$$H_{\min} \leq H \leq H_{\max}; \tag{3}$$

$$G_{\min} \leq G \leq G_{\max}. \tag{4}$$

The group of experts defined a list of five concepts, C_i, $i = 1, 2, \ldots, 5$, related to the main physical quantities of the process [8]:

- Concept C_1: volume of liquid inside the tank (depends on V_1, V_2, and, V_3);
- Concept C_2: state of V_1 (closed, open or partially open);
- Concept C_3: state of V_2 (closed, open or partially open);
- Concept C_4: state of V_3 (closed, open or partially open);
- Concept C_5: specific gravity of the produced mixture.

For this process, the fuzzy cognitive map in Figure 2 can be abstracted [8].

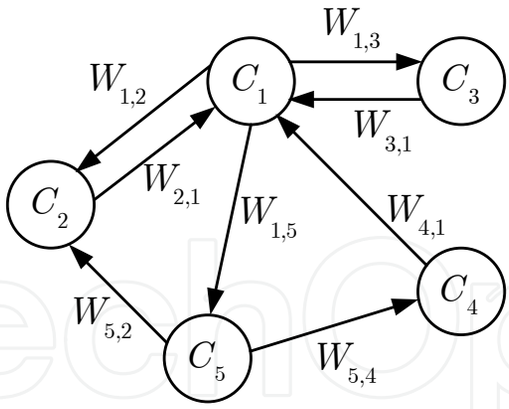


Figure 2. Fuzzy Cognitive Map proposed in [8] for the chemical process control problem.

The experts also had a consensus regarding the range of the weights between concepts, as presented in Equations (5a) to (5h).

$$-0.50 \leq W_{1,2} \leq -0.30; \tag{5a}$$
$$-0.40 \leq W_{1,3} \leq -0.20; \tag{5b}$$
$$0.20 \leq W_{1,5} \leq 0.40; \tag{5c}$$
$$0.30 \leq W_{2,1} \leq 0.40; \tag{5d}$$
$$0.40 \leq W_{3,1} \leq 0.50; \tag{5e}$$
$$-1.0 \leq W_{4,1} \leq -0.80; \tag{5f}$$
$$0.50 \leq W_{5,2} \leq 0.70; \tag{5g}$$
$$0.20 \leq W_{5,4} \leq 0.40. \tag{5h}$$

For this problem the following weight matrix is obtained:

$$\mathbf{W} = \begin{bmatrix} 0 & W_{1,2} & W_{1,3} & 0 & W_{1,5} \\ W_{2,1} & 0 & 0 & 0 & 0 \\ W_{3,1} & 0 & 0 & 0 & 0 \\ W_{4,1} & 0 & 0 & 0 & 0 \\ 0 & W_{5,2} & 0 & W_{5,4} & 0 \end{bmatrix}. \tag{6}$$

According to [8], all the experts agreed on the range of values for $W_{2,1}$, $W_{3,1}$, and $W_{4,1}$, and most of them agreed on the same range for $W_{1,2}$ and $W_{1,3}$. However, regarding the weights $W_{1,5}$, $W_{5,2}$, and $W_{5,4}$, their opinions varied significantly.

Finally, the group of experts determined that the values output concepts, C_1 and C_5, which are crucial for the system operation, must lie, respectively, in the following regions:

$$0.68 \leq A_1 \leq 0.70; \tag{7a}$$
$$0.78 \leq A_5 \leq 0.85. \tag{7b}$$

http://dx.doi.org/10.5772/52918

2.2. Second control system (PROC2)

In [9] it is considered a system consisting of two identical tanks with one input and one output valve each one, with the output valve of the first tank being the input valve of the second (PROC2), as illustrated in Figure 3.

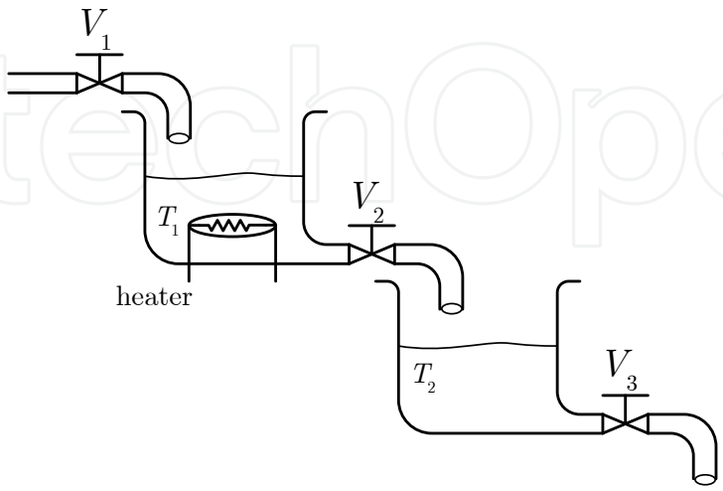


Figure 3. PROC2: a chemical process control problem described in [9].

The objective is to control the volume of liquid within the limits determined by the height $H_{\min}$ and $H_{\max}$ and the temperature of the liquid in both tanks within the limits $T_{\min}$ and $T_{\max}$, such that

$$T^1_{\min} \leq T^1 \leq T^1_{\max}; \tag{8a}$$
$$T^2_{\min} \leq T^2 \leq T^2_{\max}; \tag{8b}$$
$$H^1_{\min} \leq H^1 \leq H^1_{\max}; \tag{8c}$$
$$H^2_{\min} \leq H^2 \leq H^2_{\max}. \tag{8d}$$

The temperature of the liquid in tank 1 is increased by a heater. A temperature sensor continuously monitors the temperature in tank 1, turning the heater on or off. There is also a temperature sensor in tank 2. When T_2 decreases, the valve V_2 is open and hot liquid comes into tank 2.

Based on this process, a FCM is constructed with eight concepts:

- Concept C_1: volume of liquid inside the tank 1 (depends on V_1 and V_2);
- Concept C_2: volume of liquid inside the tank 2 (depends on V_1 and V_2);
- Concept C_3: state of V_1 (closed, open or partially open);
- Concept C_4: state of V_2 (closed, open or partially open);
- Concept C_5: state of V_3 (closed, open or partially open);
- Concept C_6: Temperature of the liquid in tank 1;

- Concept C_7: Temperature of the liquid in tank 2;
- Concept C_8: Operation of the heater.

According to [9], the fuzzy cognitive map in Figure 4 can be constructed.

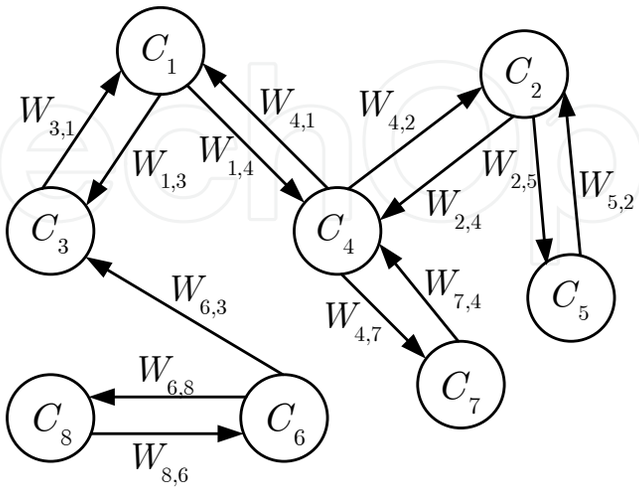


Figure 4. Fuzzy Cognitive Map proposed in [9] for the chemical process control problem.

It is assumed for PROC2 in this Chapter only causal constraints in the weights between concepts, where concepts $W_{4,1}$ and $W_{5,2}$ are $\in (-1, 0]$ and the others have positive causality. The weight matrix for PROC2 is given by

$$\mathbf{W} = \begin{bmatrix} 0 & 0 & W_{1,3} & W_{1,4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & W_{2,4} & W_{2,5} & 0 & 0 & 0 \\ W_{3,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ W_{4,1} & W_{4,2} & 0 & 0 & 0 & 0 & W_{4,7} & 0 \\ 0 & W_{5,2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & W_{6,3} & 0 & 0 & 0 & 0 & W_{6,8} \\ 0 & 0 & 0 & W_{7,4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & W_{8,6} & 0 & 0 \end{bmatrix}. \tag{9}$$

Finally, the values output concepts, C_1, C_2, C_6 and C_7, which are crucial for the system operation, must lie, respectively, in the following regions:

$$0.64 \leq A_1 \leq 0.69; \tag{10a}$$
$$0.48 \leq A_2 \leq 0.52; \tag{10b}$$
$$0.63 \leq A_6 \leq 0.67; \tag{10c}$$
$$0.63 \leq A_7 \leq 0.67. \tag{10d}$$

Two significant weaknesses of FCMs are its critical dependence on the experts opinions and its potential convergence to undesired states. In order to handle these impairments, learning procedures can be incorporated, increasing the efficiency of FCMs. In this sense, heuristic optimization approach has been deployed as an effective learning method in FCMs [15].

http://dx.doi.org/10.5772/52918

3. Heuristic FCM learning

A FCM construction can be done in the following manner:

- Identification of concepts and its interconnections determining the nature (positive, negative or null) of the causal relationships between concepts.
- Initial data acquisition by the expert opinions and/or by an equation analysis when the mathematical system model is known.
- Submitting the data from the expert opinions to a fuzzy system which output represents the weights of the FCM.
- Weight adaptation and optimization of the initially proposed FCM, adjusting its response to the desired output.
- Validation of the adjusted FCM.

This section focuses on the weight adaptation (FCM learning). In [16] a very interesting survey on FCM learning is provided. The FCM weights optimization (FCM learning) can be classified into three different methods.

1. Hebbian learning based algorithm;
2. Heuristic optimization techniques, including genetic algorithm, particle swarm optimization, differential evolution, simulated annealing, etc;
3. Hybrid approaches.

In the Hebbian based methodologies, the FCM weights are iteratively adapted based on a law which depends on the concepts behavior [10], [17]. These algorithms require the experts' knowledge for initial weight values. The differential Hebbian learning (DHL) algorithm proposed by Dickerson and Kosko is a classic example [10]. On the other hand, heuristic (metaheuristic) techniques tries to find a proper $\mathbf{W}$ matrix by minimizing a cost function based on the error among the desired values of the output concepts and the current output concepts' values (13). The experts' knowledge is not totally necessary, except for the causality constraints, due to the physical restrictions[1]. These techniques are optimization tools and generally are computationally complex. Examples of hybrid approaching considering Hebbian learning and Heuristic optimization techniques can be found in [18], [19].

There are several works in the literature dealing with heuristic optimization learning. Most of them are population-based algorithms. For instance, in [8] the PSO algorithm with constriction factor is adopted; in [20] it is presented a FCM learning based on a Tabu Search (TS) and GA combination; in [21] a variation of GA named RCGA (real codec-G.A.) is proposed; in [22] a comparison between GA and Simulated Annealing (SA) is done; in [13] the authors presented a GA based algorithm named Extended Great Deluge Algorithm.

The purpose of the learning is to determine the values of the FCM weights that will produce a desired behavior of the system, which are characterized by M output concept values that

[1] For instance, a valve cannot be negatively open.

lie within desired bounds determined by the experts. Hence, the main goal is to obtain a connection (or weight) matrix

$$\mathbf{W} = \left[W_{i,j}\right], \quad i,j = 1, 2, \ldots, N, \tag{11}$$

that leads the FCM to a steady state with output concept values within the specified region. Note that, with this notation, and defining $\mathbf{A} = [A_1 \ \cdots \ A_N]^\top$, and $\underline{\mathbf{W}} = \mathbf{W}^\top + \mathbf{I}$, with $\{\cdot\}^\top$ meaning transposition and $\mathbf{I}$ identity matrix, Equation (1) can be compactly written as

$$\mathbf{A}(k+1) = f\left(\underline{\mathbf{W}} \cdot \mathbf{A}(k)\right). \tag{12}$$

After the updating procedure in (12), the following cost function is considered for obtaining the optimum weight matrix $\mathbf{W}$ [8]:

$$\begin{aligned} F(\mathbf{W}) = &\sum_{i=1}^{M} H\left(\min\left(A_{\text{out}}^i\right) - A_{\text{out}}^i\right)\left|\min\left(A_{\text{out}}^i\right) - A_{\text{out}}^i\right| \\ &+ \sum_{i=1}^{M} H\left(A_{\text{out}}^i - \max\left(A_{\text{out}}^i\right)\right)\left|\max\left(A_{\text{out}}^i\right) - A_{\text{out}}^i\right|, \end{aligned} \tag{13}$$

where $H(\cdot)$ is the Heaviside function, and A_{out}^i, $i = 1, \ldots, M$, represents the value of the ith output concept.

3.1. Particle Swarm Optimization

The PSO is a meta-heuristic based on the movement of a population (swarm) of individuals (particles) randomly distributed in the search space, each one with its own position and velocity. The position of a particle is modified by the application of velocity in order to reach a better performance [23, 24]. In PSO, each particle is treated as a point in a $\mathcal{W}$-dimensional space[2] and represents a candidate vector. The ith particle position at instant t is represented as

$$\mathbf{x}_i(t) = \left[\, x_{i,1}(t) \ x_{i,2}(t) \ \cdots \ x_{i,\mathcal{W}}(t) \,\right]. \tag{14}$$

In this Chapter, each $x_{i,1}(t)$ represents one of the $W_{i,j}$ in the tth iteration. Each particle retains a memory of the best position it ever encountered. The best position among all particles until the tth iteration (best global position) is represented by $\mathbf{x}_g^{\text{best}}$, while the best position of the ith

[2] $\mathcal{W}$ is the number of FCM connections (relationships).

http://dx.doi.org/10.5772/52918

particle is represented as $\mathbf{x}_i^{\text{best}}$. As proposed in [25], the particles are manipulated according to the following velocity and position equations:

$$\mathbf{v}_i(t+1) = \omega \cdot \mathbf{v}_i(t+1) + \phi_1 \cdot \mathbf{U}_{1i}\left(\mathbf{x}_g^{\text{best}}(t) - \mathbf{x}_i(t)\right) + \phi_2 \cdot \mathbf{U}_{2i}\left(\mathbf{x}_i^{\text{best}}(t) - \mathbf{x}_i(t)\right) \quad (15)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1), \quad (16)$$

where ϕ_1 and ϕ_2 are two positive constants representing the individual and global acceleration coefficients, respectively, $\mathbf{U}_{1i}$ and $\mathbf{U}_{2i}$ are diagonal matrices whose elements are random variables uniformly distributed (u.d.) in the interval $[0, 1]$, and ω is the inertial weight that plays the role of balancing the global search (higher ω) and the local search (smaller ω).

A typical value for ϕ_1 and ϕ_2 is $\phi_1 = \phi_2 = 2$ [24]. Regarding the inertia weight, experimental results suggest that it is preferable to initialize ω to a large value, and gradually decrease it.

The population size $\mathcal{P}$ is kept constant in all iterations. In order to obtain further diversification for the search universe, a factor $V_{\max}$ is added to the PSO model, which is responsible for limiting the velocity in the range $[\pm V_{\max}]$, allowing the algorithm to escape from a possible local solution.

Regarding the FCM, the ith candidate vector $\mathbf{x}_i$ is represented by a vector formed by $\mathcal{W}$ FCM weights. It is important to point out that after each particle update, restrictions must be imposed on $W_{i,j}$ according to the experts opinion, before the cost function evaluation.

3.2. Genetic Algorithm

Genetic Algorithm is an optimization and search technique based on selection mechanism and natural evolution, following Darwin's theory of species' evolution, which explains the history of life through the action of physical processes and genetic operators in populations or species. GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the "fitness" (maximizes or minimizes a cost function). Such an algorithm became popular through the work of John Holland in the early 1970s, and particularly his book Adaptation in Natural and Artificial Systems (1975). The algorithm can be implemented in a binary form or in a continuous (real-valued) form. This Chapter considers the latter case.

Initially, a set of $\mathcal{P}$ chromosomes (individuals) is randomly (uniformly distributed) defined, where each chromosome, $\mathbf{x}_i$, $i = 1, 2, \cdots, \mathcal{P}$ consists of a vector of variables to be optimized, which, in this case, is formed by FCM weights, respecting the constraints. Each variable is represented by a continuous floating-point number. The $\mathcal{P}$ chromosomes are evaluated through a cost function.

T strongest chromosomes are selected for mating, generating the mating pool, using the roulette wheel method, where the probability of choosing a given chromosome is proportional to its fitness value. In this work, each pairing generates two offspring with crossover. The weakest T chromosomes are changed by the T offspring from $T/2$ pairing.

The crossover procedure is similar to the one presented in [26]. It begins by randomly selecting a variable in the first pair of parents to be the crossover point

$$\alpha = \lceil u \cdot \mathcal{W} \rceil , \tag{17}$$

where u is an u.d. random variable (r.v.) in the interval $[0, 1]$, and $\lceil \cdot \rceil$ is the upper integer operator. The jth pair of parents, $j = 1, 2, \cdots, T/2$ is defined as

$$\begin{aligned} \text{dad}_j &= \left[x^d_{j,1} \; x^d_{j,2} \; \cdots \; x^d_{j,\alpha} \; \cdots \; x^d_{j,\mathcal{W}} \right] \\ \text{mom}_j &= \left[x^m_{j,1} \; x^m_{j,2} \; \cdots \; x^m_{j,\alpha} \; \cdots \; x^m_{j,\mathcal{W}} \right] \end{aligned} . \tag{18}$$

Then the selected variables are combined to form new variables that will appear in the offspring

$$\begin{aligned} x^{\text{o}}_{j,1} &= x^m_{j,\alpha} - \beta \left[x^m_{j,\alpha} - x^d_{j,\alpha} \right] ; \\ x^{\text{o}}_{j,2} &= x^d_{j,\alpha} + \beta \left[x^m_{j,\alpha} - x^d_{j,\alpha} \right] . \end{aligned} \tag{19}$$

where β is also a r.v. u.d. in the interval $[0, 1]$. Finally,

$$\begin{aligned} \text{offspring}_1 &= \left[x^d_{j,1} \; x^d_{j,2} \; x^d_{j,\alpha-1} \; \cdots \; x^{\text{o}}_{j,1} \; x^m_{j,\alpha+1} \cdots \; x^m_{j,\mathcal{W}} \right] \\ \text{offspring}_2 &= \left[x^m_{j,1} \; x^m_{j,2} \; x^m_{j,\alpha-1} \; \cdots \; x^{\text{o}}_{j,2} \; x^d_{j,\alpha+1} \cdots \; x^d_{j,\mathcal{W}} \right]. \end{aligned} \tag{20}$$

In order to allow escaping from possible local minima, a mutation operation is introduced in the resultant population, except for the strongest one (elitism). It is assumed in this work a Gaussian mutation. If the rate of mutations is given by P_m, there will be $N_m = \lceil P_m \cdot (\mathcal{P} - 1) \cdot \mathcal{W} \rceil$ mutations uniformly chosen among $(\mathcal{P} - 1) \cdot \mathcal{W}$ variables. If $x_{i,\bar{w}}$ is chosen, with $w = 1, 2, \cdots, \mathcal{W}$, than, after Gaussian mutation, it is substituted by

$$x'_{i,\bar{w}} = x_{i,\bar{w}} + \mathcal{N}\left(0, \sigma^2_m\right) , \tag{21}$$

where $\mathcal{N}\left(0, \sigma^2_m\right)$ represents a normal r.v. with zero mean and variance σ^2_m.

After mutation, restrictions must be imposed on $W_{i,j}$ according to the experts opinion, before the cost function evaluation.

3.3. Differential Evolution

The Differential Evolution (DE) search has been introduced by Ken Price and Rainer Storn [27, 28]. DE is a parallel direct search method. As in GA, a population with $\mathcal{P}$ elements is randomly defined, where each $\mathcal{W}$-dimension element consists of a $\mathbf{x}$ vector of variables to be optimized (FCM weights in this case) respecting the constraints.

http://dx.doi.org/10.5772/52918

In the classical DE, a perturbation is created by using a difference vector based mutation,

$$\mathbf{y}_i = \mathbf{x}_{r0} + F_e \cdot (\mathbf{x}_{r1} - \mathbf{x}_{r2}), \qquad i = 1, 2, \cdots, \mathcal{P}, \tag{22}$$

where the real and constant factor F_e (typically $\in [0.5, 1.0]$) controls the gain of the differential variation. The indexes $r0$, $r1$ and $r2$ are randomly chosen and mutually exclusive. In this work, an alternative perturbation procedure named DE/current-to-best/1/bin is considered [28, 29], such that

$$\mathbf{y}_i = \mathbf{x}_i + F_e \cdot (\mathbf{x}_{\text{best}} - \mathbf{x}_i) + F_e \cdot (\mathbf{x}_{r1} - \mathbf{x}_{r2}), \qquad i = 1, 2, \cdots, \mathcal{P}, \tag{23}$$

There are also other variants of the perturbation procedure [28, 29]. A uniform crossover operation is applied in order to have diversity enhancement, which mixes parameters of the mutation vector $\mathbf{y}_i$ and $\mathbf{x}_i$, for generating the trial vector $\mathbf{u}_i$:

$$\mathbf{u}_i = \begin{cases} \mathbf{y}_i & \text{if } (r \leq \chi) \\ \mathbf{x}_i & \text{otherwise} \end{cases} \qquad i = 1, 2, \cdots, \mathcal{P}, \tag{24}$$

where χ is the crossover constant typically $\in [0.8, 1.0]$ and r is a random variable u.d. in the interval $[0, 1)$. In order to prevent the case $\mathbf{u}_i = \mathbf{x}_i$, at least one component is taken from the mutation vector $\mathbf{y}_i$.

For selecting, the algorithm uses a simple method where the trial vector $\mathbf{u}_i$ competes against the target vector $\mathbf{x}_i$, such that,

$$\mathbf{x}_i(t+1) = \begin{cases} \mathbf{u}_i(t) & \text{if } f(\mathbf{u}_i(t)) \leq f(\mathbf{x}_i(t)) \\ \mathbf{x}_i(t) & \text{otherwise} \end{cases} \tag{25}$$

4. Simulation results

All the simulations were taken considering 10^4 trials for PROC1 and PROC2. It is worth noting that PSO, GA and DE input parameters were chosen previously after exhaustive simulation tests. As a result, optimal or quasi-optimal input parameters for the PSO, GA and DE heuristic algorithms have been obtained.

4.1. Process PROC1

The adopted values for the input parameters of PSO, GA and DE are summarized on Table 1. DE has less parameter inputs than the other two methods, which is a relative advantage. Four different scenarios for the process PROC1 were analyzed. The main performance results for each scenario is described in the next subsections.

4.1.1. Scenario 1

This scenario considers all the constraints on the FCM weights shown in Equations (5a) to (5h). As mentioned in [8] and also verified here, there is no solution in this case.

PSO	
Population	$\mathcal{P} = 10, 20, 30$
Acceleration Coefficients	$\phi_1 = 2$, $\phi_2 = 0.2$
Inertial Weight	$\omega = 1.2$
Initial Velocity	$\mathbf{v}_i(0) = 0.1$
Maximum Velocity	$V_{\max} = 2$
GA	
Population	$\mathcal{P} = 10$
Mating Pool	$T = 2 \cdot \text{round}(\mathcal{P}/4)$
Rate of Mutation	$P_m = 0.2$
Mutation Std. Deviation	$\sigma_m = 0.3$
DE	
Population	$\mathcal{P} = 10, 20, 30$
Crossover Constant	$\chi = 0.9$
DE Gain Variation Factor	$F_e = 0.9$

Table 1. PSO, GA and DE input parameters values for PROC1.

4.1.2. Scenario 2

In this scenario, the constraints on the FCM weights $W_{1,5}$, $W_{5,2}$, and $W_{5,4}$ have been relaxed, since the experts' opinions have varied significantly. In this case the values of these weights were allowed to assume any value in the interval $[0, 1]$ in order to keep the causality of relationships. Tables 2, 3 and 4 present the obtained simulation results for PSO, GA and DE, respectively.

As can be observed, in the Scenario 2, GA has a better performance than DE and PSO, achieving convergence without failure with $\mathcal{P} = 10$. PSO has the second best performance, presenting no convergence errors in 10^4 independent experiments when $\mathcal{P} = 20$. DE does not achieve 100% of performance success, even for $\mathcal{P} = 30$, when 63 errors occurred in 10^4 independent experiments. Figures 5 and 6 present the mean FCM concepts convergence in the Scenario 2 under 10^4 trials. Considering the best case simulated for each algorithm (GA with $\mathcal{P} = 10$, PSO with $\mathcal{P} = 20$ and DE with $\mathcal{P} = 30$), DE and PSO presented similar average convergence, being faster than GA.

PSO with $\mathcal{P} = 10$					
$A_{\max}$	0.6882	0.8058	0.6203	0.8389	0.8186
$A_{\min}$	0.6477	0.7297	0.5749	0.6590	0.7800
A_{average}	0.6840	0.7911	0.6178	0.6713	0.8148
Number of Failures: 49					
Probability of Success: 0.9951					
PSO with $\mathcal{P} = 20$					
$A_{\max}$	0.6882	0.8051	0.6183	0.6967	0.8186
$A_{\min}$	0.6800	0.7297	0.5750	0.6590	0.7801
A_{average}	0.6838	0.7926	0.6178	0.6730	0.8139
Number of Failures: 0					
Probability of Success: 1.0					

Table 2. PSO simulation results for Scenario 2, PROC1.

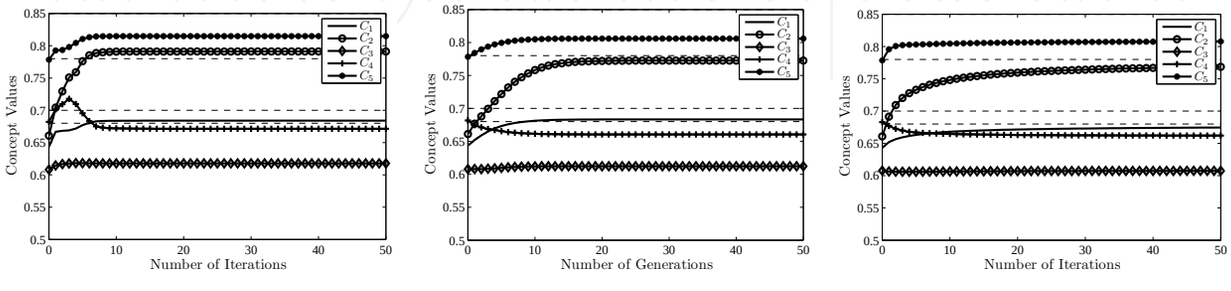
http://dx.doi.org/10.5772/52918

GA with $\mathcal{P} = 10$					
A_{max}	0.6882	0.8051	0.6183	0.6964	0.8186
A_{min}	0.6800	0.7297	0.5749	0.6590	0.7800
A_{average}	0.6833	0.7726	0.6123	0.6605	0.8059
Number of Failures: 0					
Probability of Success: 1.0					

Table 3. GA simulation results for Scenario 2, PROC1.

DE with $\mathcal{P} = 10$					
A_{max}	0.6882	0.8062	0.6217	0.8389	0.8186
A_{min}	0.6243	0.5563	0.5749	0.6590	0.6590
A_{average}	0.6746	0.7685	0.6074	0.6620	0.8078
Number of Failures: 4372					
Probability of Success: 0.5628					
DE with $\mathcal{P} = 20$					
A_{max}	0.6882	0.8058	0.6203	0.8389	0.8186
A_{min}	0.6477	0.5563	0.5749	0.6590	0.7776
A_{average}	0.6810	0.7845	0.6070	0.6616	0.8089
Number of Failures: 537					
Probability of Success: 0.9463					
DE with $\mathcal{P} = 30$					
A_{max}	0.6882	0.8054	0.6192	0.6965	0.8186
A_{min}	0.6653	0.5984	0.5749	0.6590	0.7800
A_{average}	0.6819	0.7861	0.6072	0.6614	0.8089
Number of Failures: 63					
Probability of Success: 0.9937					

Table 4. DE simulation results for Scenario 2, PROC1.



(a) (b) (c)

Figure 5. Mean convergence for (a) PSO, (b) GA and (c) DE in PROC1, Scenario 2 with $\mathcal{P} = 10$.

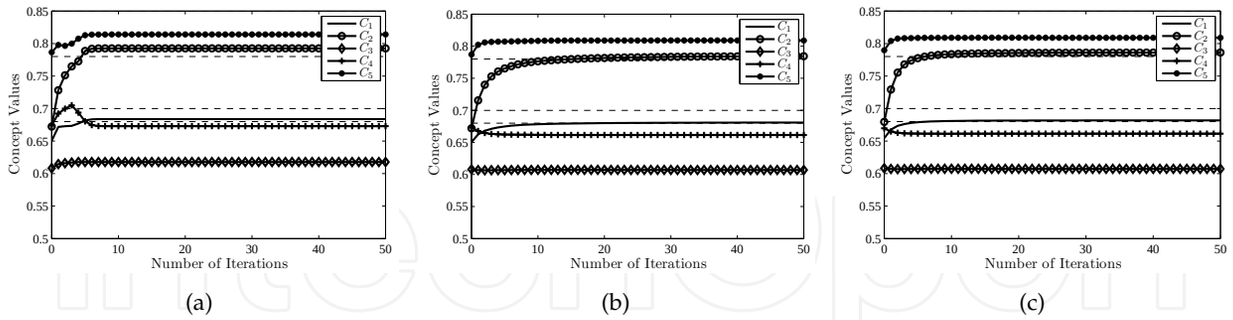


Figure 6. Mean convergence in PROC1, Scenario 2 for (a) PSO with $\mathcal{P} = 20$, (b) DE with $\mathcal{P} = 20$ and (c) DE with $\mathcal{P} = 30$.

4.1.3. Scenario 3

In this Scenario, all the weights constrains were relaxed, but the causalities were kept, i.e., the value of the weights were fixed in the interval $[0, 1]$ or in the interval $[-1, 0)$, according to the causality determined by the experts. Table 5 presents the obtained results. As can be seen, $\mathcal{P} = 10$ was enough for achieving 100% of convergence in GA and PSO. With $\mathcal{P} = 10$, DE was not able to find the proper solution in 36 trials, resulting in a probability of success equal to 0.9964, but when $\mathcal{P} = 20$, DE obtained 100% of success. Figure 7 presents the mean convergence of the concepts in 10^4 independent experiments. In this scenario, DE presented the fastest average convergence.

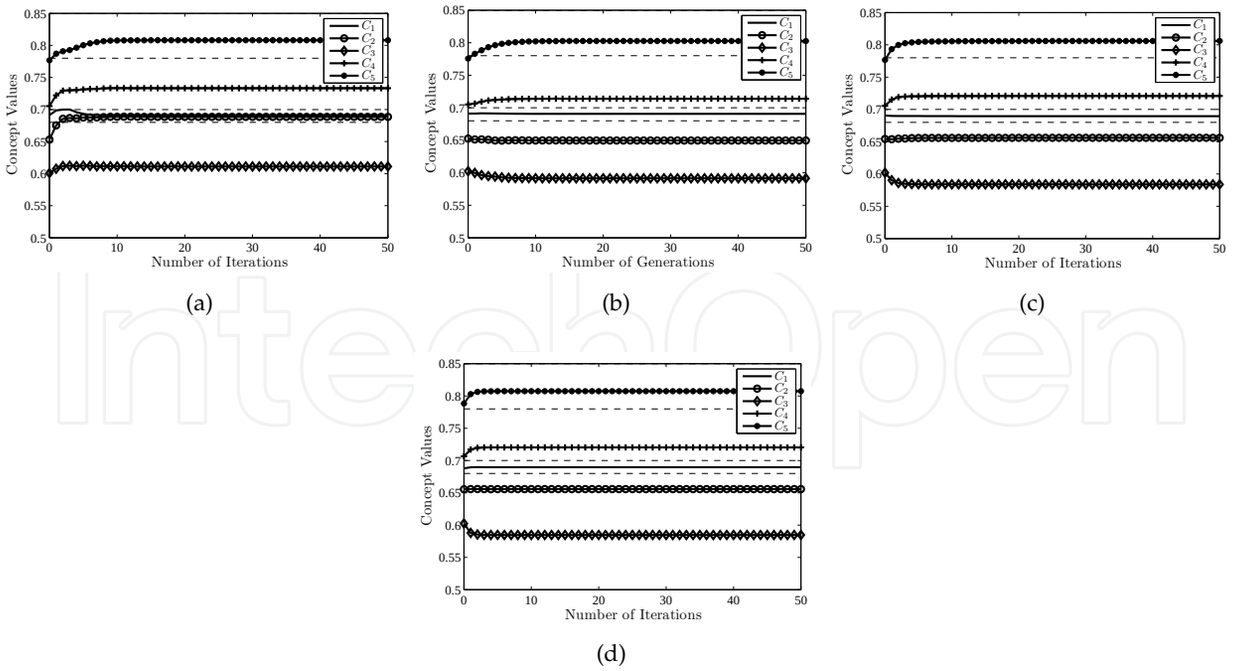


Figure 7. Mean convergence for Scenario 3 in PROC1 considering (a) PSO with $\mathcal{P} = 10$, (b) GA with $\mathcal{P} = 10$, (c) DE with $\mathcal{P} = 10$, and (d) DE with $\mathcal{P} = 20$.

http://dx.doi.org/10.5772/52918

PSO with $\mathcal{P} = 10$					
A_{max}	0.7000	0.8404	0.6590	0.8404	0.8206
A_{min}	0.6800	0.4339	0.4339	0.6590	0.7800
A_{average}	0.6907	0.6886	0.6112	0.7333	0.8080
Number of Failures: 0					
Probability of Success: 1.0					
GA with $\mathcal{P} = 10$					
A_{max}	0.7000	0.8404	0.6590	0.8404	0.8206
A_{min}	0.6800	0.4339	0.4339	0.6590	0.7800
A_{average}	0.6906	0.6496	0.5914	0.7139	0.8027
Number of Failures: 0					
Probability of Success: 1.0					
DE with $\mathcal{P} = 10$					
A_{max}	0.7189	0.8404	0.6590	0.8404	0.8206
A_{min}	0.6590	0.4294	0.4277	0.6590	0.6590
A_{average}	0.6896	0.6560	0.5839	0.7209	0.8060
Number of Failures: 36					
Probability of Success: 0.9964					
DE with $\mathcal{P} = 20$					
A_{max}	0.7000	0.8404	0.6590	0.8404	0.8206
A_{min}	0.6800	0.4339	0.4339	0.6590	0.7800
A_{average}	0.6897	0.6557	0.5848	0.7204	0.8074
Number of Failures: 0					
Probability of Success: 1.0					

Table 5. GA, PSO and DE simulation results for Scenario 3, PROC1.

4.1.4. Scenario 4

In this situation, the causality and the strength of the causality were totally relaxed. PSO and GA algorithms were able to determine proper weights for the connection matrix with $\mathcal{P} = 10$, and DE did not have success in only 4 experiments, resulting in a probability of success equal to 0.9996. When $\mathcal{P} = 20$, DE did not presented convergence errors. Table 6 presents the obtained results, while Figure 8 presents the mean convergence of the concepts in 10^4 independent experiments. As in Scenario 3, DE presented the fastest average convergence, while GA and PSO presented similar average convergence speed.

4.2. Process PROC2

For this process the value of the weights were fixed in the interval $[0, 1]$ or in the interval $[-1, 0)$, according to the causality determined by the experts. The adopted input simulation parameters for PSO, GA and DE algorithms were the same considered in PROC1, except for the population size in PSO that was assumed $\mathcal{P} = 10$ and $\mathcal{P} = 20$.

Table 7 summarizes the simulation results, while Figures 9 and 10 present the mean convergence of the concepts value considering 10^4 independent experiments. As can be seen, $\mathcal{P} = 10$ was enough for achieving convergence rate of 100% in GA. With $\mathcal{P} = 10$, PSO presented 30 failures and probability of success equal to 0.9970, while DE presented 897 failures and probability of success equal to 0.9103. However, with $\mathcal{P} = 20$, both PSO and DE were able to achieve 100% of success. For this process, DE has a faster average convergence than PSO and GA, being PSO the second fastest.

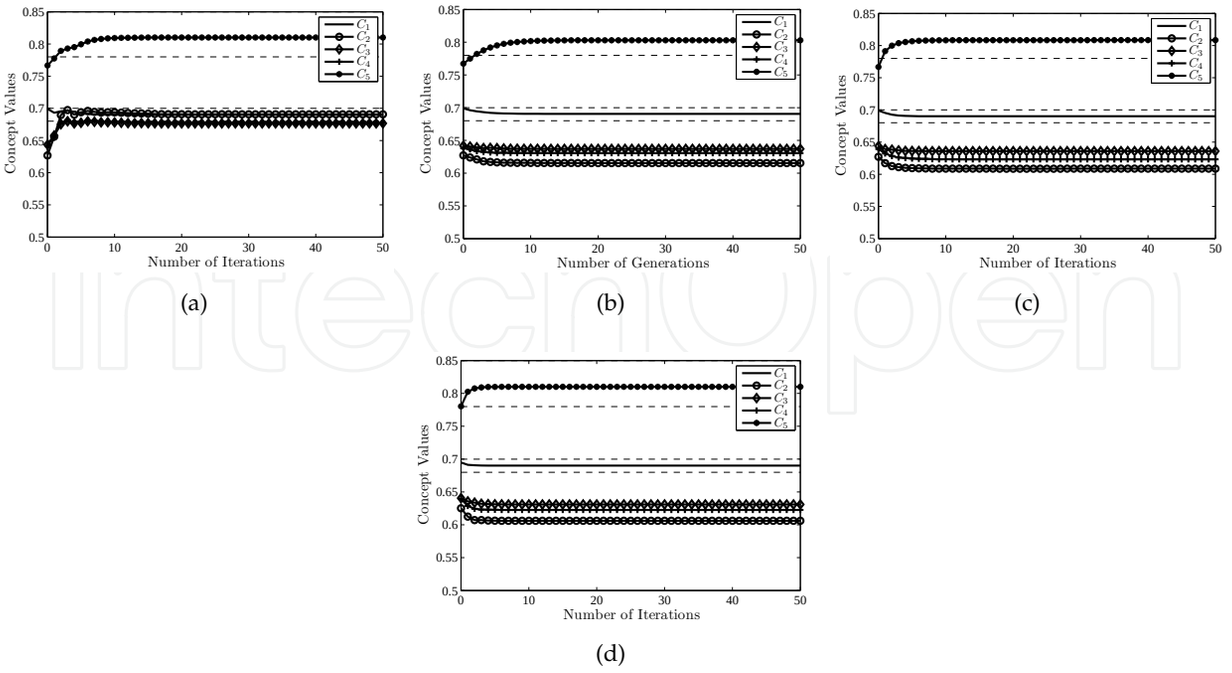


Figure 8. Mean convergence for Scenario 4 in PROC1 considering (a) PSO with $\mathcal{P} = 10$, (b) GA with $\mathcal{P} = 10$, (c) DE with $\mathcal{P} = 10$, and (d) DE with $\mathcal{P} = 20$.

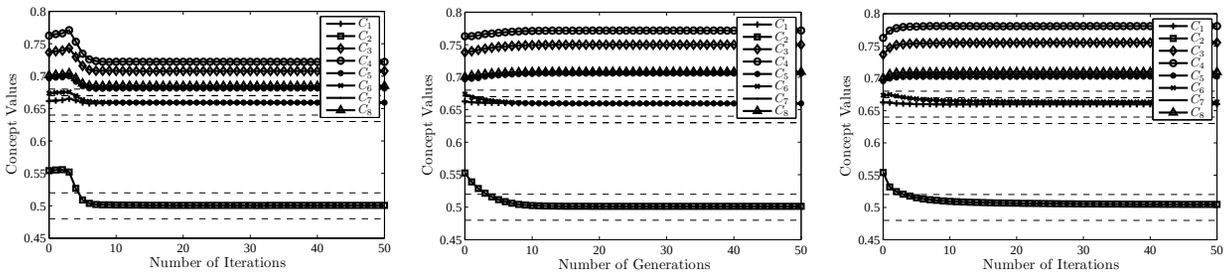


Figure 9. Mean convergence for (a) PSO, (b) GA and (c) DE in PROC2 with $\mathcal{P} = 10$.

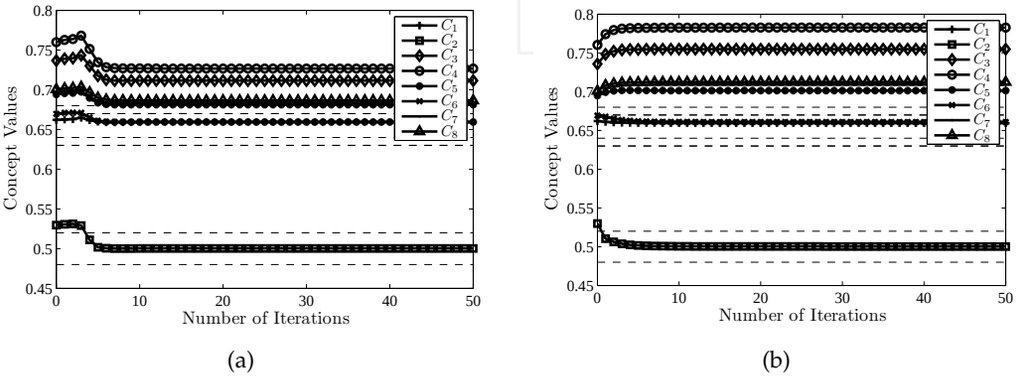


Figure 10. Mean convergence for (a) PSO and (b) DE in PROC2 with $\mathcal{P} = 20$.

http://dx.doi.org/10.5772/52918

PSO with $\mathcal{P} = 10$					
$A_{\max}$	0.7000	0.9199	0.8206	0.8404	0.8206
$A_{\min}$	0.6800	0.2129	0.4339	0.3952	0.7800
A_{average}	0.6901	0.6908	0.6767	0.6777	0.8101
Number of Failures: 0					
Probability of Success: 1.0					
GA with $\mathcal{P} = 10$					
$A_{\max}$	0.7000	0.9192	0.8206	0.8404	0.8206
$A_{\min}$	0.6800	0.2130	0.4339	0.3952	0.7800
A_{average}	0.6905	0.6154	0.6371	0.6306	0.8030
Number of Failures: 0					
Probability of Success: 1.0					
DE with $\mathcal{P} = 10$					
$A_{\max}$	0.7166	0.9199	0.8206	0.8404	0.8229
$A_{\min}$	0.6800	0.2112	0.4285	0.3948	0.7628
A_{average}	0.6901	0.6087	0.6358	0.6232	0.8084
Number of Failures: 4					
Probability of Success: 0.9996					
DE with $\mathcal{P} = 20$					
$A_{\max}$	0.7000	0.9199	0.8206	0.8404	0.8206
$A_{\min}$	0.6800	0.2129	0.4338	0.3952	0.7800
A_{average}	0.6902	0.6061	0.6311	0.6227	0.8102
Number of Failures: 0					
Probability of Success: 1.0					

Table 6. GA, PSO and DE simulation results for Scenario 4, PROC1.

5. Conclusions

There are in the literature, several models based on fuzzy cognitive maps developed and adapted to a large range of applications. The use of learning algorithms may be necessary for obtaining proper values for the weight matrix and reducing the dependence on the experts' knowledge.

Within this context, this Chapter presented a comparison of three heuristic search approaches, PSO, GA and DE, applied to FCM weight optimization in two processes control. In all the considered cases, GA presented a performance in terms of probability of success better or equal to the other two schemes, being PSO the second best technique in terms of probability of success in the two considered processes.

Specially in scenario 2 of PROC 1, when there are several weight constraints, GA achieved 100% of success in 10^4 independent experiments with a population of 10 chromosomes. PSO needed $\mathcal{P} = 20$ particles in the population in order to reach 100% of success. DE was not able to achieve 100% of success even for $\mathcal{P} = 30$.

PSO with $\mathcal{P} = 10$								
$A_{\max}$	0.7040	0.6590	0.9029	0.9407	0.8134	0.7234	0.6700	0.8153
$A_{\min}$	0.6400	0.4130	0.6590	0.6590	0.6590	0.6590	0.6590	0.6590
A_{average}	0.6592	0.5006	0.7079	0.7221	0.6809	0.6593	0.6592	0.6850
Number of Failures: 30								
Probability of Success: 0.9970								
GA with $\mathcal{P} = 10$								
$A_{\max}$	0.6800	0.5200	0.9038	0.9409	0.7870	0.6700	0.6700	0.8153
$A_{\min}$	0.6400	0.4800	0.6590	0.6590	0.6590	0.6590	0.6590	0.6590
A_{average}	0.6596	0.5014	0.7500	0.7717	0.7055	0.6596	0.6596	0.7082
Number of Failures: 0								
Probability of Success: 1.0								
DE with $\mathcal{P} = 10$								
$A_{\max}$	0.7523	0.6591	0.9120	0.9477	0.8134	0.7608	0.7530	0.8271
$A_{\min}$	0.5747	0.4472	0.6590	0.6590	0.6590	0.6590	0.6590	0.6590
A_{average}	0.6597	0.5046	0.7553	0.7808	0.7026	0.6632	0.6632	0.7097
Number of Failures: 897								
Probability of Success: 0.9103								
PSO with $\mathcal{P} = 20$								
$A_{\max}$	0.6800	0.5200	0.9040	0.9411	0.7869	0.6700	0.6700	0.8154
$A_{\min}$	0.6400	0.4800	0.6590	0.6590	0.6590	0.6590	0.6590	0.6590
A_{average}	0.6594	0.5002	0.7116	0.7267	0.6813	0.6594	0.6593	0.6865
Number of Failures: 0								
Probability of Success: 1.0								
DE with $\mathcal{P} = 20$								
$A_{\max}$	0.6800	0.5200	0.9043	0.9427	0.7870	0.6700	0.6700	0.8154
$A_{\min}$	0.6400	0.4800	0.6590	0.6590	0.6590	0.6590	0.6590	0.6590
A_{average}	0.6595	0.5003	0.7548	0.7827	0.7015	0.6608	0.6608	0.7124
Number of Failures: 0								
Probability of Success: 1.0								

Table 7. GA, PSO and DE simulation results for PROC2.

In most scenarios, DE has presented a faster convergence in comparison to PSO and GA, but the population size needed in DE was higher than the other two methods.

Author details

Bruno Augusto Angélico[1], Márcio Mendonça[1],
Lúcia Valéria R. de Arruda[2], Taufik Abrão[3]

1 Federal University of Technology - Paraná (UTFPR), Campus Cornélio Procópio, PR, Brazil
2 Federal University of Technology - Paraná (UTFPR), Campus Curitiba, PR, Brazil
3 State University of Londrina - Paraná (UEL), Londrina, PR, Brazil

http://dx.doi.org/10.5772/52918

References

[1] Kosko B. Fuzzy cognitive maps. *International Journal of Man-Machine Studies*, 24:65–75, January 1986.

[2] Kosko B. *Neural networks and fuzzy systems : a dynamical systems approach to machine intelligence*. Prentice Hall, Upper Saddle River, NJ, 1992.

[3] Kosko B. *Fuzzy Engineering*. Prentice-Hall, Inc., Upper Saddle River, NJ, 1997.

[4] Axelrod R.M. *Structure of Decision : The Cognitive Maps of Political Elites*. Princeton University Press, Princeton, NJ, 1976.

[5] Andreou A.S., Mateou N.H., and Zombanakis G.A. Soft computing for crisis management and political decision making: the use of genetically evolved fuzzy cognitive maps. *Soft Computing*, 9:194–210, March 2005.

[6] Stylios C.D., Georgopoulos V.C., Malandraki G.A., and Chouliara S. Fuzzy cognitive map architectures for medical decision support systems. *Applied Soft Computing*, 8:1243–1251, June 2008.

[7] Papageorgiou E.I., Stylios C.D., and Groumpos P.P. An integrated two-level hierarchical system for decision making in radiation therapy based on fuzzy cognitive maps. *IEEE Transactions on Biomedical Engineering*, 50(12):1326 –1339, dec. 2003.

[8] Papageorgiou E.I., Parsopoulos K.E., Stylios C.S., Groumpos P.P., and Vrahatis M.N. Fuzzy cognitive maps learning using particle swarm optimization. *Journal of Intelligent Information Systems*, 25:95–121, July 2005.

[9] Stylios C.D., Georgopoulos V.C., and Groumpos P.P. The use of fuzzy cognitive maps in modeling systems. In *Proceeding of 5th IEEE Mediterranean Conference on Control and Systems, Paphos*, 1997.

[10] Dickerson J. A. and Kosko B. Virtual worlds as fuzzy cognitive maps. In *IEEE Virtual Reality Annual International Symposium*, volume 3, pages 471 – 477, 1993.

[11] Taber R. Knowledge Processing With Fuzzy Cognitive Maps. *Expert Systems With Applications*, 1(2):83–87, 1991.

[12] Perusich K. Fuzzy cognitive maps for policy analysis. In *International Symposium on Technology and Society Technical Expertise and Public Decisions*, pages 369 –373, jun 1996.

[13] Baykasoglu A., Durmusoglu Z.D.U., and K. Vahit. Training fuzzy cognitive maps via extended great deluge algorithm with applications. *Computers in Industry*, 62(2):187 – 195, 2011.

[14] Glykas M. *Fuzzy Cognitive Maps: Advances in Theory, Methodologies, Tools and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[15] Parsopoulos K.E. and Groumpos P.P. A first study of fuzzy cognitive maps learning using particle swarm optimization. In *IEEE 2003 Congress on Evolutionary Computation*, pages 1440–1447, 2003.

[16] Papageorgiou E.I. Learning algorithms for fuzzy cognitive maps - a review study. *IEEE Transactions on Systems, Man, and Cybernetics - Part C*, 42(2):150 –163, march 2012.

[17] Stylios C.D. Papageorgiou E.I. and P.P. Groumpos. Active hebbian learning algorithm to train fuzzy cognitive maps. *International Journal of Approximate Reasoning*, 37(3):219–249, 2004.

[18] Papageorgiou E.I. and Groumpos P.P. A new hybrid learning algorithm for fuzzy cognitive maps learning. *Applied Soft Computing*, 5:409–431, 2005.

[19] Zhu Y. and Zhang W. An integrated framework for learning fuzzy cognitive map using rcga and nhl algorithm. In *Int. Conf. Wireless Commun., Netw. Mobile Comput., Dalian, China*, 2008.

[20] Alizadeh S., Ghazanfari M., Jafari M., and Hooshmand S. Learning fcm by tabu search. *International Journal of Computer Science*, 3:142–149, 2007.

[21] Stach W., Kurgan L.A., Pedrycz W., and Reformat M. Genetic learning of fuzzy cognitive maps. *Fuzzy Sets and Systems*, 153(3):371–401, 2005.

[22] Ghazanfari M., Alizadeh S., Fathian M., and Koulouriotis D.E. Comparing simulated annealing and genetic algorithm in learning fcm. *Applied Mathematics and Computation*, 192(1):56–68, 2007.

[23] Kennedy J. and Eberhart R. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[24] Kennedy J. and Eberhart R.C. *Swarm Intelligence*. Morgan Kaufmann, first edition, March 2001.

[25] Shi Y. and Eberhart R.C. A modified particle swarm optimizer. In *IEEE International Conference on Evolutionary Computation*, pages 69–73, 1998.

[26] Haupt R.L. and Haupt S.E. *Practical Genetic Algorithms*. John Wiley & Sons, Hoboken, NJ, USA, 2nd edition, 2004.

[27] Storn R. and Price K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11(4):341–359, December 1997.

[28] Storn R.M. Price K.V. and Lampinen J.A. *Differential Evolution A Practical Approach to Global Optimization*. Natural Computing Series. Springer-Verlag, Berlin, Germany, 2005.

[29] Das S. and Suganthan P.N. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4 –31, feb. 2011.