Applied Intelligence (2019) 49:3652–3667
https://doi.org/10.1007/s10489-019-01476-7

# A novel quantum inspired algorithm for sparse fuzzy cognitive maps learning

Mojtaba Kolahdoozi[1] · Abdollah Amirkhani[2] · Mohammad H. Shojaeefard[3] · Ajith Abraham[4]

Published online: 1 May 2019


**Abstract**
Fuzzy cognitive maps (FCMs) represent a graphical modeling technique based on the decision-making and reasoning rules and algorithms similar to those used by humans. The graph-like structure and the execution model of FCMs respectively allow static and dynamic analyses to be carried out. The learning algorithms of FCMs that are based on expert opinion are weak in dynamic analysis, and fully automatic algorithms are weak in static analysis. In this paper, for providing the facility for simultaneous static and dynamic analyses, a new training algorithm called the quantum FCM (QFCM) is presented. In our proposed algorithm, the quantum inspired evolutionary algorithm (QEA) and the particle swarm optimization algorithm are employed for generating static and dynamic analyses properties respectively. In the QFCM, instead of coding the presence and absence of links between concepts with 1 and 0, respectively, the probability of their existence or inexistence is modeled with a Q-bit (the smallest information unit in the QEA) and, depending on the outcome of dynamic analysis, the quantum state of this Q-bit is updated. Using a probabilistic representation instead of 0 and 1, in addition to creating diversity in the solution space, can lead to escapes from many local optima; which is an issue of concern in the optimization of FCM structure. Experiments on synthetic, real-life, and gene regulatory network reconstruction problems demonstrated that not only does QFCM find potentially good structures, providing static analysis, but also it brings about low data error, showing good dynamic property. Furthermore, QFCM successfully outshined most of the state-of-the-art FCM's learning algorithms, without any need to human knowledge, illustrating its power in this regard.

**Keywords** Quantum theory · Evolutionary algorithm · Fuzzy cognitive maps · Sparse learning · Dynamic analysis

## 1 Introduction

A fuzzy cognitive map (FCM) is a fuzzy directed graph consisting of nodes (concepts) and the interconnections between nodes (edges) [1]. Each node indicates a system state or variable. A connection between nodes is a causal relationship and shows the degree of influence of one node on another [2]. FCM has several advantages, which is not present in neural networks as well as expert systems: transparency, adaptability, and flexibility [3]; hence they have been used in an assortment of applications such as time series prediction [4], control and robotics [5, 6], to name but a few. Based on the survey which was conducted by Stach et al [7], the FCM training algorithms are divided into three types: manual, semi-automatic, and fully-automatic. This division is based on the type of knowledge which is adopted in the process of learning.

In the manual algorithms, the opinions of specialists in the field for which an FCM is designed are used to get the link weights. For this purpose, the association between two nodes is described by an IF-THEN statement [8]. Since the expert-based models rely on limited human knowledge, they don't have a strong dynamic analysis capability; and also the designing of FCMs with a large number of nodes becomes a difficult task for specialists [9, 10].

In fully-automatic algorithms, an evolutionary algorithm such as particle swarm optimization (PSO) [11] or genetic algorithm [12] is generally used along with an objective

✉ Abdollah Amirkhani
amirkhani@iust.ac.ir

[1] Department of Electrical Engineering, Iran University of Science and Technology, Tehran 16846-13114, Iran

[2] School of Automotive Engineering, Iran University of Science and Technology, Tehran 16846-13114, Iran

[3] Department of Mechanical Engineering, Iran University of Science and Technology, Tehran 16846-13114, Iran

[4] Machine Intelligence Research Labs (MIR Labs), Auburn, WA 98071, USA

function in order to find the weight matrix of FCM. No specialist is involved in the process of FCM design by fully-automatic algorithms [13]. The FCMs designed by fully-automatic algorithms are much denser than those designed with the help of specialists. Since the sparseness of an FCM is directly related to its transparency and static analysis ability, the FCMs that are trained by fully-automatic algorithms are weak in static analysis [14].

Semiautomatic methods make an effort to use the strong points of expert-based techniques as well as fully-automatic approaches. Most of these semiautomatic techniques first try to establish an initial FCM structure based on the knowledge of specialists and then to improve the numerical values of weights by using automatic methods. Semiautomatic algorithms are not as good as expert-based algorithms in static analysis, and not as efficient as fully-automatic algorithms in dynamic analysis [14].

To overcome these flaws and also to provide the ability of simultaneous static and dynamic analysis, two algorithms of SRCGA [15] and MOEA-FCM [16] have been proposed recently.

In [15], the authors intended to train FCMs by using the 'crossover' and 'mutation' operators along with the objective function, which considers the error of dynamic analysis. Their main novelty is to introduce a parameter called 'density estimate,' whose value indicates the density achieved by the FCM (the density of FCM is equal to the ratio of non-zero links to the total number of links). The major drawback of this method is that estimating the density of FCMs is difficult in real-life problems. Although the authors of the SRCGA paper have proposed a default density estimate of 37% for cases in which an FCM density cannot be estimated, they have got this 37% figure by averaging the densities of FCMs recently developed by specialists; and therefore by using several other models of these FCMs, a number other than 37% may be obtained.

In [16], the authors have modeled the FCM training problem as a multi-objective optimization problem with two objective functions of error and density, and have tried to simultaneously minimize these two objectives by the well-known NSGA-II algorithm [17]. Since the obtained Pareto front contains various densities and since, for comparison with the results of previous works, ultimately one density should be selected, the authors in [16], pursuant to the authors in [15], have chosen a density of 37% in their accuracy report.

Despite the MOEA-FCM [16] results, its flaw lies in the fact that a user cannot determine which of the obtained densities in the Pareto front provides a better static analysis. To shed more light on the matter, let's look at

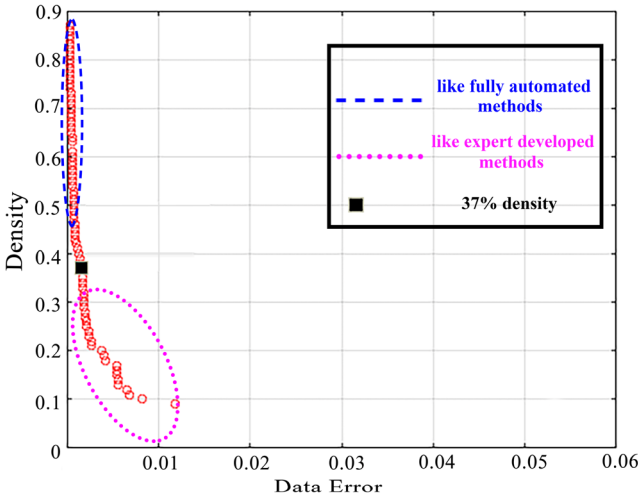


**Fig. 1** The Pareto front obtained from the simulation of MOEA-FCM algorithm

Fig. 1. This figure is the result of our simulation for an FCM with 10 nodes and 40% density by means of MOEA-FCM algorithm. As this figure shows, the top section of the Pareto front is denser and contains less error; so it resembles the fully-automatic FCM training methods. The bottom section of the front has a higher error and is more like the FCMs developed by specialists. However, the midsection of the Pareto front, which can hopefully provide simultaneous static and dynamic analyses (The 37% density is also in this region.), includes a vast number of different densities; thus making it difficult to choose the density that can achieve the best, or the closest best, dynamic and static analyses simultaneously.

To deal with the abovementioned problems, a new algorithm called the quantum fuzzy cognitive map (QFCM) is presented in this paper for FCM training. The proposed QFCM uses quantum evolutionary algorithm (QEA) [18], and thus probabilistic representation, for encoding the FCM structure in terms of Q-bits (static analysis) along with PSO algorithm [19] for obtaining the ability of dynamic analysis. Unlike the both MOEA-FCM [16] and SRCGA [15], QFCM is able to fully automatically converge to the best structure and weights, or at least the closest best, removing the difficulty of choosing a best density inside a pool of different densities ranging from 1% to 100%. To verify the efficacy of the QFCM, we have applied it on 3 different datasets – that is – synthetic, real-life, and DREAM3 as well as DREAM4, two popular benchmark datasets for gene regulatory network (GRN) reconstruction task which are provided by dialogue for reverse engineering assessments and methods (DREAM) challenge [20–22]. In most of the cases, it outperformed

other newly devised algorithms, showing the promising results of this algorithm.

The rest of this paper has been organized as follows: the QEA, PSO algorithm, and fuzzy cognitive maps are introduced and described in Section 2. The QFCM algorithm is thoroughly explained in Section 3. The empirical results are presented in Section 4. And finally, the conclusion of this paper is presented in Section 5.

## 2 Background

### 2.1 Fuzzy cognitive maps

In vector $C = [C_1 \ldots C_{N_n}]$, which is used to define the nodes (concepts), $C_i$denotes the $i^{th}$ node. The causal relationships between the $i^{th}$ and $j^{th}$ nodes are defined by weight matrix $w = \{w_{ij}, 1 \le i \le N_n \ 1 \le j \le N_n\}$, where $w_{ij} \in [-1, +1]$, and it indicates the weight of the link from the $i^{th}$ node to the $j^{th}$ nodes. If $w_{ij} > 0$, then an increase (decrease) in the $i^{th}$ node will create an increase (decrease) of intensity $|w_{ij}|$ in the $j^{th}$ node. And if $w_{ij} < 0$, then an increase (decrease) in $C_i$ will produce a decrease (increase) in $C_j$. Also, $w_{ij} = 0$ means that there is no causal relationship between concepts $C_i$ and $C_j$ [23].

The value of each node in the $(t+1)^{th}$iteration is influenced by the weight matrix and the state values of other nodes connected to it in the $t^{th}$iteration. The value of the $i^{th}$node in the $(t+1)^{th}$iteration is obtained from:

$$C_i(t+1) = \varphi\left(\sum_{j=1}^{N_n} w_{ij} C_j(t)\right) \tag{1}$$

in which, $C_i(t)$ is the value of the $i^{th}$ node in the $t^{th}$ iteration and $\varphi$ is a transfer function, whose task is to confine the node values to the interval [0, 1]. In order for an FCM to make an inference, vector $C$ is repeatedly passed through the weight matrix $w$. Different transfer functions can be used in Eq. 1. According to [24], the sigmoid transfer function is superior to other transfer functions; thus, a sigmoid function in the form of Eq. 2 has been used in this paper.

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

where, $\lambda$ is a parameter that determines the slope of transfer function about the point zero; and a small $\lambda$ is usually used for severely nonlinear systems [25]. In the majority of former works, [15, 16, 26], $\lambda$ has been considered as 5; and so for the sake of comparison with previous works, the value of $\lambda$ was also considered as 5 in this paper. By

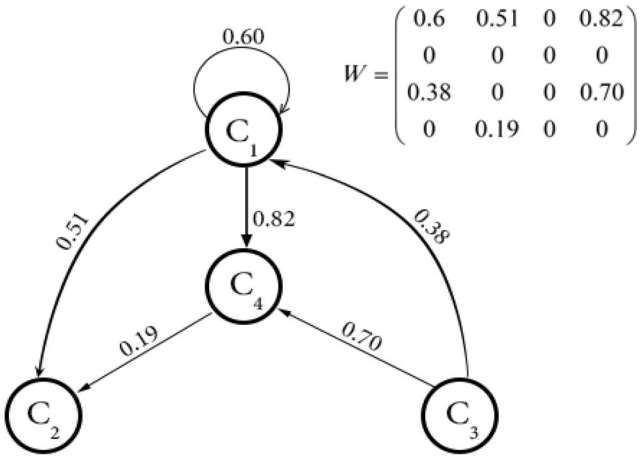


**Fig. 2** A sample FCM and its associated weight matrix

using continuous transfer functions, the FCM simulation can achieve a fixed state vector value known as the 'fixed point attractor', or it can fluctuate between several fixed state vector values, which is known as the 'limit cycle'. Reaching a chaotic attractor is also possible, in which case, the FCM produces different state vector values in successive iterations. Figure 2 shows an FCM with 4 nodes along with the corresponding weight matrix, and Fig. 3 illustrates the simulation of the fixed point attractor for this FCM.

### 2.2 The particle swarm optimization (PSO) algorithm

Inspired by the social behavior of birds and fish, Kennedy and Eberhart introduced the optimization algorithm of PSO [19] in 1995 which has gained a lot of popularity among researchers [27–29]. As an optimization algorithm, PSO provides a population-based search mechanism in which every single individual (i.e. particle) changes its position with time. In the PSO algorithm, the particles

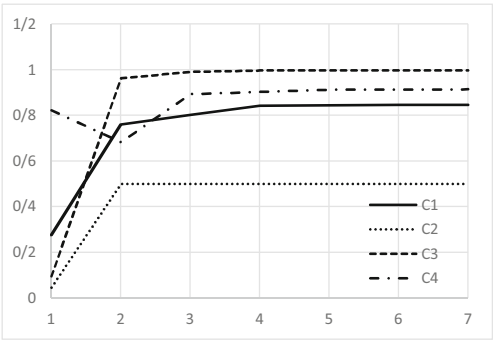


**Fig. 3** Simulating the fixed point attractor for the FCM of Fig. 2 by using a random

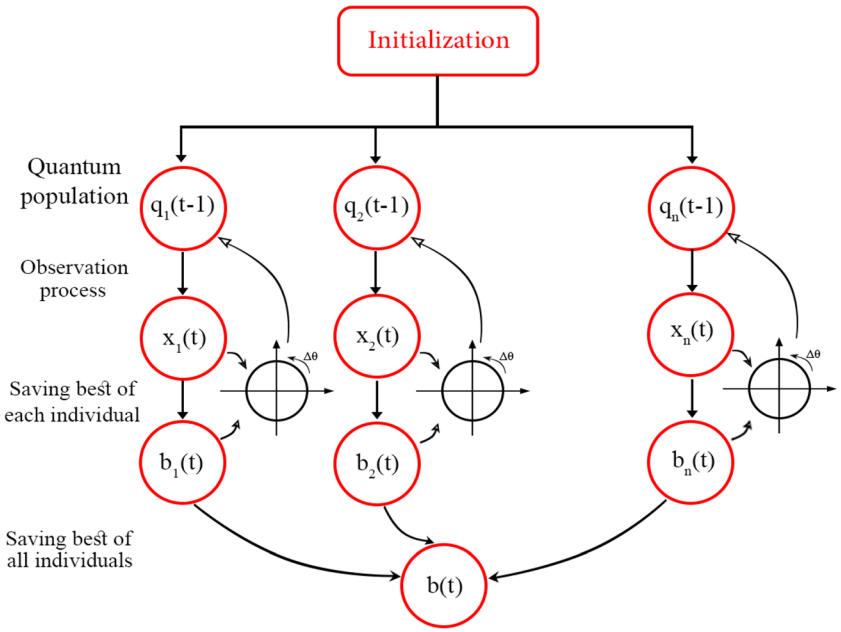


**Fig. 4** The general schematic of QEA algorithm

fly in a multidimensional search space. During flight, each particle varies its position with regard to the best position it has already experienced and the best global position of the whole particles.

The position and the velocity of the $i^{th}$ particle in the search space are determined by vectors $x_i = (x_{i1} \ldots x_{id})^t$ and $v_i = (v_{i1} \ldots v_{id})^t$, respectively; where $d$represents the number of dimensions of the search space. The best experienced position of the $i^{th}$ particle is saved and indicated by $pbest_i = (pbest_{i1} \ldots pbest_{id})^t$, and the best global position of the whole particles is specified by $gbest = (gbest_1 \ldots gbest_d)^t$. The modified velocity and position of each particle can be expressed as:

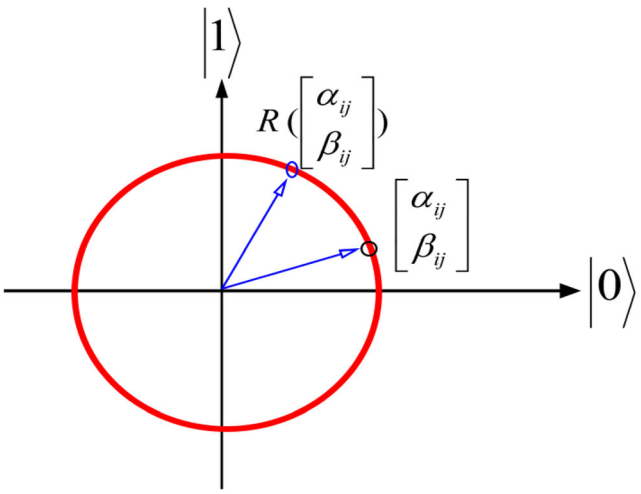


**Fig. 5** The function of the quantum rotation gate

$$v_i(t+1) = \omega v_i(t) + c_1 r_1 (pbest_i - x_i(t)) + c_2 r_2 (gbest - x_i(t)) \quad (3)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (4)$$

where, $\omega$ is the inertia, $r_1$ and $r_2$ are two random numbers of uniform distribution in the interval [0, 1], and $c_1$ and $c_2$ are acceleration constants. Since it is difficult to tune parameters$\omega$, $c_1$ and $c_2$ in real problems, Sierra et al. [30] have suggested:

$$v_i(t+1) = W v_i(t) + C_1 r_1 (pbest_i - x_i(t)) + C_2 r_2 (gbest - x_i(t)) \quad (5)$$

in which, $\omega$, $c_1$ and $c_2$ are 3 random numbers of uniform distribution in intervals [0.1, 0.5], [1.5, 2] and [1.5, 2], respectively.

### 2.3 The quantum inspired evolutionary algorithm

The various aspects of quantum computations and physics have been a source of inspiration for the design and development of numerous mathematical tools and algorithms such as power system stability [31], modeling the Prisoner's dilemma [32], neural networks [33] and particle swarm optimization [34]. One of the most successful experiences of using the quantum idea in evolutionary computations is the presentation of the QEA [18] by Han and Kim. This algorithm is based on the use of Q-bit. The governing state of a Q-bit can be indicated by:

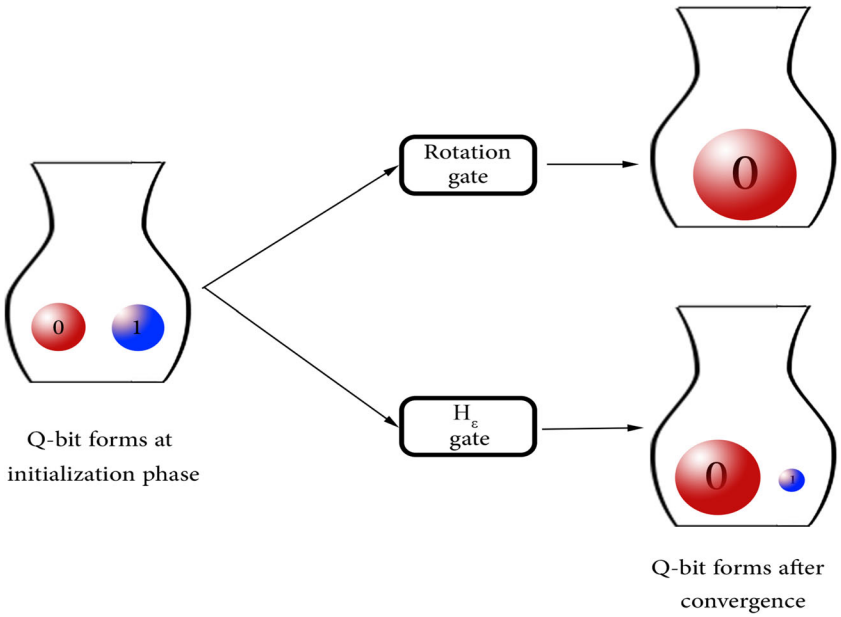


**Fig. 6** The functions of the rotation gate and $H_\varepsilon$ gate

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{6}$$

where, $|\alpha|^2$ and $|\beta|^2$ indicate the probability of a Q-bit as being '0' and '1', respectively. Since the sample space only contains 0 s and 1 s, then $|\alpha|^2 + |\beta|^2 = 1$. Every candidate solution in the problem space is obtained by putting together m Q-bits.

$$Q_i = \begin{bmatrix} \alpha_{i1} & |\ldots| & \alpha_{im} \\ \beta_{i1} & & \beta_{im} \end{bmatrix} \tag{7}$$

in which, $Q_i$ denotes the $i^{th}$candidate solution and it can simultaneously indicate $2^m$ states. For example, the probability of $Q_i$ being in state '0' is $|\alpha_{i1}|^2 \times |\alpha_{i2}|^2 \times \ldots \times |\alpha_{im}|^2$. Using the superposition of states and probabilistic representation along with quantum gates for the manipulation of Q-bits, provides an enormous solution diversity for QEA; because with only m Q-bits, $2^m$ states in the solution space can be manipulated simultaneously; and this makes the QEA very powerful in searching the problem space.

The initial version of QEA was presented in 2002 [18], and the general schematic of this algorithm can be observed in Fig. 4.

In the initialization phase, n candidate solutions such as Eq. 7, with $|\alpha_{ij}| = |\beta_{ij}| = \frac{1}{\sqrt{2}}$, form the initial quantum population. The reason for considering a value of $\frac{1}{\sqrt{2}}$ for $\alpha_{ij}$ and $\beta_{ij}$s ($1 \le j \le m \quad 1 \le i \le n$) is that at the start of the algorithm, each Q-bit has the same probability ($\frac{1}{2}$) of being in state 0 or 1. Then, during the observation process, a state of 0 or 1 is assigned to each Q-bit. For this purpose, a random number r with uniform distribution is generated in the interval [0, 1]. If $r < |\alpha_{ij}|^2$, state 0 is assigned to the $j^{th}$ Q-bit; otherwise, state 1 is assigned to it; and in this way, a binary sequence $x_j$is obtained from the candidate solution $Q_i$.

The best experience of each candidate solution and the best solution of the entire population are saved as $b_j(t)$ and $b(t)$, respectively. The variation operator in the QEA is the quantum rotation gate. The rotation gate is defined as

$$R\left(\begin{bmatrix} \alpha_{ij} \\ \beta_{ij} \end{bmatrix}\right) = \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) \\ \sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \begin{bmatrix} \alpha_{ij} \\ \beta_{ij} \end{bmatrix} \tag{8}$$

where, $\Delta\theta$ determines the amount of rotation, and its normal value is in the interval $[0.001\pi, 0.05\pi]$. Figure 5 shows the graphical schematic of a Q-bit, before and after applying the quantum gate of rotation. Also, the sign of $\Delta\theta$ is determined with respect to the type of problem examined.

In QEA, two mechanisms have been predicted for escaping from local optima. One of these is the local migration. In local migration, one of the $b_j(t)$s which provides a better solution than the rest of $b_j(t)$s is copied in them (Fig. 4). The second mechanism is the global migration. In this case, the best solution ($b(t)$) is copied in all the $b_j(t)$s.

The second version of QEA [35] has three major differences from the first version:

1. Offering a new stopping condition instead of the maximum iteration termination condition which is based on the convergence of Q-bits. Hence, contrary to most evolutionary algorithms, the termination condition of QEA is transparent and interpretable.
2. Offering a new quantum gate known as the $H_\varepsilon$ gate for preventing the entrapment of Q-bits in states 0 and 1. The problem with the rotation gate was that if, for example, a Q-bit converged to $\alpha_{ij} = 1$, in the observation process, it was always observed as having a state of 0; and therefore, its escape from local optima was only possible by means of local and global migration schemes. In order to solve this problem, the $H_\varepsilon$ quantum gate was presented according to

$$H_\varepsilon\left(\begin{bmatrix}\alpha_{ij}\\ \beta_{ij}\end{bmatrix}\right) = \begin{cases}\begin{bmatrix}\sqrt{\varepsilon}\\ \sqrt{1-\varepsilon}\end{bmatrix} & |\alpha_{ij}|^2 \le \varepsilon \\ R\left(\begin{bmatrix}\alpha_{ij}\\ \beta_{ij}\end{bmatrix}\right) & \varepsilon < |\alpha_{ij}|^2 < 1-\varepsilon \\ \begin{bmatrix}\sqrt{1-\varepsilon}\\ \sqrt{\varepsilon}\end{bmatrix} & |\alpha_{ij}|^2 \ge 1-\varepsilon\end{cases} \quad (9)$$

in which, $\varepsilon$ is a design parameter with a typical value of 0.01 [35]. According to this equation, the Q-bits converge to $\alpha_{ij} = \sqrt{\varepsilon}$ or $\alpha_{ij} = \sqrt{1-\varepsilon}$ instead of converging to $\alpha_{ij} = 0$ or $\alpha_{ij} = 1$ and, consequently, they can escape from local optima during the observation process with probability $\varepsilon$. In fact, if we assume the $H_\varepsilon$ and the rotation gates as two vases and the Q-bit as two balls of 0 and 1 (one of which is to be taken out of a vase during the observation process), after convergence, only one ball will remain inside the rotation gate vase, but two balls will be left inside the $H_\varepsilon$ gate vase; the ratio of the volume of one ball (the probability of its selection) to that of the second ball is equal to $\frac{1-\varepsilon}{\varepsilon}$ (Fig. 6).

3. Studying the effect of the initial values of Q-bits at the start of the algorithm and presenting a two-phase scheme QEA based on the optimally-tuned initial values of Q-bits. It has been demonstrated that the initial values of Q-bits influence the convergence speed and the quality of final solutions; hence a quantum two-phase algorithm has been presented. In the first phase, the search space is divided into smaller sections, and the initial values of each Q-bit are placed in the middle of these sections. In the second phase, the Q-bits are initialized with the best initial values obtained from the first phase, and then the QEA is executed with these initial values.

**Table 1** The values of $\Delta\theta$

| *Qobserved* | *Qbestobserved* | $f(Qobserved) \ge f(Qbestobserved)$ | 1st and 3rd quadrants | 2nd and 4th quadrants |
|---|---|---|---|---|
| 0 | 0 | True | 0 | 0 |
| | | False | 0 | 0 |
| 0 | 1 | True | $+0.005\pi$ | $-0.005\pi$ |
| | | False | 0 | 0 |
| 1 | 0 | True | $-0.005\pi$ | $+0.005\pi$ |
| | | False | 0 | 0 |
| 1 | 1 | True | 0 | 0 |
| | | False | 0 | 0 |

By comparing the two algorithms of PSO and QEA, the inherent similarities between these two algorithms can be observed. For example, for guiding the genes (in PSO and QEA terminologies, a gene is called 'particle' and 'Q-bit', respectively) toward better solutions, both algorithms use the best solution of the entire population and also the best individual experience of each member of the population. In QEA, the best solution of the entire population is used through the global migration mechanism.

## 3 The QFCM algorithm

In this section, the QFCM algorithm for training fuzzy cognitive maps, with the ability of simultaneous static and dynamic analyses, will be presented and explained. The QFCM models the likelihood of the presence/absence of a link in the FCM in the form of a Q-bit, the state of which is modified with regard to the error obtained from dynamic analysis. We have used the PSO algorithm for obtaining the numerical values of weights under the structure determined by the QEA; therefore, network structure is determined by the Q-bits, and the numerical values of weights are obtained by means of the PSO algorithm.

After training an FCM by the QFCM algorithm, the density of the trained FCM will converge to the real density (best static analysis) or, at least, to a density close to the real density. Also, by presenting a mathematical theorem, the probability of escaping from the local optima of the densest FCM (fully-connected FCM) by the QFCM will be discussed. Since the denser models produce smaller errors than the sparser models, the presented theorem displays the power and ability of the QFCM in escaping from these dense local optima.

The pseudocode of the QFCM algorithm and the explanations of its various steps have been given below.

ALGORITHM I
THE QFCM ALGORITHM

1. Initialize $Q(i)$ $(1 \le i \le nPop)$.
2. **For** every $i = 1...nPop$, do the following:
   2.1. Initialize $nSubPop$ numbers of particles for the $i^{th}$ subpopulation ($p(i,j), 1 \le i \le nPop$ and $1 \le j \le nSubPop$). Also, set the $pbest(i,j)$ of each particle equal to $p(i,j)$.
   2.2. Set the initial velocity of each particle equal to zero ($v(i,j) = 0$).
   2.3. Observe $Q(i)$ in order to obtain $Qobserved(i)$ and $Qbestobserved(i)$.
   2.4. **For** $j = 1...nSubPop$, do the following:
      2.4.1. Evaluate the $j^{th}$ particle under the $Qobserved(i)$ structure and put the cost obtained from the objective function in $p\cos t(i,j)$ and $pbest\cos t(i,j)$.
   **End for**
   2.5. Put $\min_{1 \le j \le nSubPop}(p\cos t(i,j))$ in $Q\cos t(i)$, $Qbest\cos t(i)$ and $gbest\cos t(i)$.
   2.6. Put $\arg\min_{1 \le j \le nSubPop}(p\cos t(i,j))$ in $gbest(i)$.
**End for**
3. Put $\arg\min_{1 \le i \le nPop}(Q\cos t(i))$ in $b$ and put $\min_{1 \le i \le nPop}(Q\cos t(i))$ in $b\cos t$.
4. **For** $iter = 1...MaxIter$, do the following:
   4.1. **For** $i = 1...nPop$, do the following:
      4.1.1. Observe $Q(i)$ and put it in $Qobserved(i)$.
      4.1.2. **For** $j = 1...nSubPop$, do the following:
         4.1.2.1. Update $v(i,j)$.
         4.1.2.2. Update $p(i,j)$.
         4.1.2.3. **If** the mutation conditions are satisfied for the $j^{th}$ particle, **then** perform the mutation.
         4.1.2.4. Repair the $j^{th}$ particle.
         4.1.2.5. Evaluate the $j^{th}$ particle under the $Qobserved(i)$ structure and put the cost obtained from the objective function in $p\cos t(i,j)$.
         4.1.2.6. Update $pbest(i,j)$ and $pbest\cos t(i,j)$.
      **End for**
      4.1.3. Update $gbest(i)$ and $gbest\cos t(i)$.
      4.1.4. Put $\min_{1 \le j \le nSubPop}(p\cos t(i,j))$ in $Q\cos t(i)$.
      4.1.5. Update $Q(i)$.
      4.1.6. Update $Qbestobserved(i)$ and $Qbest\cos t(i)$.
   **End for**
   4.2. Update $b$ and $b\cos t$.
   4.3. **If** the conditions permit, **then** perform global or local migration.
**End for**

Steps 1, 2 and 3 of QEA are the initialization steps, and Step 4 constitutes the main loop of the algorithm. In Step 1, *nPop* quantum candidate solution is generated (e.g. Eq. 7), with $|\alpha_{ij}| = |\beta_{ij}| = \frac{1}{\sqrt{2}}$ $(1 \le i \le nPop, 1 \le j \le N_n^2)$. For every quantum candidate solution, *nSubPop* particles are generated, as the $i^{th}$ subpopulation (e.g. Eq. 10); and each of these particles is also considered as the best personal experience.

$$w_{ij} = \left[ w_{ij_1} \ldots w_{ij_{N_n^2}} \right] \quad w_{ij_k} \in [-1, -0.05) \cup (+0.05, +1] \qquad (10)$$

If $|w_{ij_k}| < 0.05$, that weight cannot represent a causal relationship between two nodes in the FCM [12]. The velocity of each particle in Step 2.2 is initialized as zero. In Step 2.3, via the observation process, each quantum candidate solution is converted to a binary sequence of $N_n^2$ elements, and this binary sequence is saved as *Qobserved* and as the best personal experience of that solution (*Qbestobserved*). In this binary sequence, the existence or the lack of a link is indicated by 1 or 0, respectively. Up to this point in the algorithm, the presence and the absence of links and also their initial values have been determined and, therefore, each of the particles can be evaluated. The learned FCM weights are obtained by multiplying each *Qobserved* by its corresponding $w_{ij}$.

After obtaining the cost of each particle, these costs are saved in $p\cos t$and $pbest\cos t$. In Step 2.5, the lowest cost of the $i^{th}$ subpopulation is saved as the cost of the $i^{th}$ quantum candidate solution, cost of the best personal experience of the $i^{th}$ quantum candidate solution, and also the cost of the best experience of the entire particles of the $i^{th}$ subpopulation. In Step 2.6, the particle with the lowest cost among the entire particles of the $i^{th}$ subpopulation is saved as the *gbest*. When the initialization loop ends, the candidate quantum solution with the lowest cost is saved as *b*.

In the main loop of the algorithm, the position and velocity of each particle are updated by Eq. 4 and Eq. 5 respectively. Then in Step 4.1.2.3, mutation is performed on the position of particles. For this purpose, uniform mutation is performed on half of each subpopulation (indices $\left[\frac{nSubPop}{2}\right]$ to *nSubPop*) with a probability of $\mu$, and no mutation is carried out on the other half. Mutation is followed by repair. In the repair step, the following four cases could be considered for the position of each particle:

1. $p(i,j) \in [-0.05, +0.05]$; in this case we have $p(i,j) = 0$.
2. $p(i,j) \in [-1, -0.05) \cup (+0.05, 1]$; in this case no repair is carried out on $p(i,j)$.
3. $p(i,j) \in (+1, +\infty)$; in this case, the position of a particle is set to +1 and its velocity is multiplied by −1.
4. $p(i,j) \in (-\infty, -1)$; in this case, the position of a particle is set to −1 and its velocity is also multiplied by −1.

Following the repair step, the particles are evaluated under the structure *Qbestobserved*(*i*), and the costs of particles are saved in $p\cos t(i,j)$. If $p\cos t(iter) < pbest(iter-1)$, then *pbest* is set equal to $p\cos t$. In Step 4.1.3, *gbest*(*i*) is updated to $\min(p\cos t(i,j))_{1 \le j \le nSubPop}$, and this value is also saved as the cost of the $i^{th}$ quantum candidate solution. Step 4.1.5 includes the updating of quantum candidate solutions (*Q*(*i*)s). In this step, each Q-bit within a *Q*(*i*) is updated with regard to its observed bits in *Qobserved* and *Qbestobserved*, and by using the $H_\varepsilon$ quantum gate (Eq. 9). The sign of $\Delta\theta$ in the $H_\varepsilon$ gate should be set so as to lead us to better solutions. In this paper, we have adjusted the values and sign of $\Delta\theta$ according to Table 1, which are also proposed in the [18] as standard values of $\Delta\theta$. In cases in which no inference can be made regarding the sign of $\Delta\theta$ (for example, when both *Qobserved* and *Qbestobserved* are zero), the value of $\Delta\theta$ is set to zero.

After updating *Q*(*i*), if $Q\cos t(iter) < Qbest\cos t(iter-1)$, the value of $Qbest\cos t$is updated to $Q\cos t$. The best observed solution in Step 4.2 (*b*) is updated to the observed value of quantum solution with the lowest cost. Global and local migrations are performed in the last step of the algorithm. Global migration and local migration are performed every GMperiod and LMperiod iterations, respectively. Parameters of GMperiod and LMperiod are selected with regard to problem complexity.

As it was previously mentioned, the cost of denser FCM models is lower, and therefore, potentially robust structures may get thrown away during training. In this respect, Theorem 1 expresses the fact that even in one of the worst case scenarios (the densest FCM), the QFCM algorithm, contrary to conventional algorithms such as GA, is able to escape from local optima.

**Theorem 1** For an FCM that has been trained by the QFCM algorithm and converged to a fully-connected structure, the probability of escaping from a local optimum, during the observation through the $H_\varepsilon$ gate, is $1-(1-\varepsilon)^{N_n^2}$.

**Proof** According to [36], after infinite iterations, each Q-bit will converge to $\begin{bmatrix} \sqrt{\varepsilon} \\ \sqrt{1-\varepsilon} \end{bmatrix}$ or $\begin{bmatrix} \sqrt{1-\varepsilon} \\ \sqrt{\varepsilon} \end{bmatrix}$. However, since the FCM has converged to the densest structure, all the Q-bits have converged to $\begin{bmatrix} \sqrt{\varepsilon} \\ \sqrt{1-\varepsilon} \end{bmatrix}$:

**Table 2** The values of free QFCM parameters

| Parameter | nPop | nSubPop | $\varepsilon$ | $\mu$ | MaxIter | GMperiod | LMperiod |
|---|---|---|---|---|---|---|---|
| Value | 100 | 80 | 0.01 | 0.01 | 3000 | 20 | 10 |

**Table 3** The empirical results (averages and standard deviations) for synthetic data

| Algorithm | Number of nodes | Density | Data error | Out of sample error | Matrix error | SSmean | Converged density |
|---|---|---|---|---|---|---|---|
| QFCM | 5 | 20% | **0.00(0.00)** | **0.00(0.00)** | **0.00(0.00)** | **1** | 20% |
| | 5 | 40% | **0.00(0.00)** | **0.00(0.00)** | **0.00(0.00)** | **1** | 40% |
| | 10 | 20% | **0.003(0.002)** | **0.027(0.032)** | **0.101(0.143)** | 0.70 | 22% |
| | 10 | 40% | **0.001(0.002)** | 0.048(0.054) | **0.149(0.192)** | 0.72 | 41% |
| | 20 | 20% | **0.006(0.005)** | **0.081(0.103)** | 0.132(0.194) | **0.63** | 25% |
| | 20 | 40% | 0.007(0.006) | 0.059(0.073) | 0.203(0.229) | **0.68** | 44% |
| | 40 | 20% | **0.012(0.02)** | 0.115(0.140) | 0.138(0.239) | **0.55** | 23% |
| | 40 | 40% | **0.011(0.023)** | **0.126(0.151)** | 0.239(0.237) | 0.53 | 44% |
| dMAGA [37] | 5 | 20% | 0.00(0.00) | – | 0.015(0.008) | 0.87 | – |
| | 5 | 40% | 0.002(0.001) | – | 0.146(0.098) | 0.75 | – |
| | 10 | 20% | 0.004(0.002) | – | 0.133(0.100) | **0.81** | – |
| | 10 | 40% | 0.002(0.001) | – | 0.192(0.157) | 0.53 | – |
| | 20 | 20% | 0.010(0.005) | – | 0.143(0.148) | 0.53 | – |
| | 20 | 40% | 0.025(0.005) | – | **0.174(0.163)** | 0.43 | – |
| | 40 | 20% | 0.056(0.009) | – | 0.152(0.132) | 0.38 | – |
| | 40 | 40% | 0.109(0.015) | – | **0.190(0.128)** | 0.39 | – |
| MOEA-FCM [16] | 5 | 20% | 0.002(0.001) | 0.013(0.010) | 0.002(0.003) | 0.96 | – |
| | 5 | 40% | 0.002(0.002) | 0.024(0.015) | 0.004(0.002) | 0.94 | – |
| | 10 | 20% | 0.004(0.002) | 0.031(0.038) | 0.103(0.152) | 0.72 | – |
| | 10 | 40% | 0.004(0.003) | **0.047(0.053)** | 0.152(0.193) | 0.68 | – |
| | 20 | 20% | 0.007(0.005) | 0.085(0.102) | 0.139(0.186) | 0.62 | – |
| | 20 | 40% | 0.008(0.010) | **0.058(0.073)** | 0.201(0.225) | 0.68 | – |
| | 40 | 20% | 0.014(0.018) | **0.113(0.138)** | 0.139(0.238) | 0.55 | – |
| | 40 | 40% | 0.015(0.025) | 0.130(0.154) | 0.238(0.240) | 0.52 | – |
| SRCGA [15] | 5 | 20% | 0.004(0.004) | 0.010(0.012) | 0.002(0.002) | 0.89 | – |
| | 5 | 40% | 0.004(0.005) | 0.008(0.011) | 0.003(0.003) | 0.89 | – |
| | 10 | 20% | 0.005(0.005) | 0.123(0.141) | 0.105(0.198) | 0.71 | – |
| | 10 | 40% | 0.005(0.005) | 0.124(0.151) | 0.168(0.200) | **0.76** | – |
| | 20 | 20% | 0.007(0.006) | 0.142(0.143) | **0.122(0.219)** | 0.62 | – |
| | 20 | 40% | **0.006(0.008)** | 0.146(0.148) | 0.203(0.295) | 0.68 | – |
| | 40 | 20% | 0.017(0.020) | 0.166(0.183) | **0.135(0.235)** | 0.50 | – |
| | 40 | 40% | 0.019(0.022) | 0.164(0.196) | 0.245(0.288) | **0.58** | – |
| RCGA [12] | 5 | 20% | 0.005(0.006) | 0.017(0.017) | 0.321(0.381) | 0.20 | – |
| | 5 | 40% | 0.005(0.005) | 0.012(0.018) | 0.361(0.372) | 0.16 | – |
| | 10 | 20% | 0.006(0.007) | 0.135(0.137) | 0.398(0.322) | 0.19 | – |
| | 10 | 40% | 0.006(0.007) | 0.131(0.140) | 0.385(0.316) | 0.20 | – |
| | 20 | 20% | 0.008(0.009) | 0.151(0.149) | 0.426(0.346) | 0.16 | – |
| | 20 | 40% | 0.008(0.008) | 0.152(0.149) | 0.413(0.376) | 0.14 | – |
| | 40 | 20% | 0.019(0.022) | 0.171(0.187) | 0.453(0.385) | 0.15 | – |
| | 40 | 40% | 0.020(0.022) | 0.167(0.189) | 0.436(0.368) | 0.15 | – |
| DD-NHL [38] | 5 | 20% | 0.197(0.186) | 0.199(0.209) | 0.317(0.345) | 0.11 | – |
| | 5 | 40% | 0.189(0.197) | 0.197(0.203) | 0.381(0.333) | 0.12 | – |
| | 10 | 20% | 0.191(0.188) | 0.201(0.181) | 0.412(0.356) | 0.13 | – |
| | 10 | 40% | 0.211(0.184) | 0.192(0.189) | 0.423(0.348) | 0.10 | – |
| | 20 | 20% | 0.203(0.245) | 0.201(0.222) | 0.464(0.316) | 0.14 | – |
| | 20 | 40% | 0.203(0.168) | 0.203(0.224) | 0.436(0.348) | 0.13 | – |
| | 40 | 20% | 0.194(0.168) | 0.198(0.199) | 0.468(0.388) | 0.12 | – |
| | 40 | 40% | 0.187(0.160) | 0.206(0.209) | 0.465(0.384) | 0.12 | – |

$$converged\,FCM\,structure = \left[\frac{\sqrt{\varepsilon}}{\sqrt{1-\varepsilon}} \mid \ldots \mid \frac{\sqrt{\varepsilon}}{\sqrt{1-\varepsilon}}\right]$$

Because the sum of the probabilities of all the members of sample space is equal to 1, we have:

$$\sum_{i=1}^{2^{N_n^2}} P_i = 1 \rightarrow \sum_{i=1}^{2^{N_n^2}-1} P_i + P_{2^{N_n^2}} = 1 \rightarrow \sum_{i=1}^{2^{N_n^2}-1} P_i = 1 - P_{2^{N_n^2}} \rightarrow \sum_{i=1}^{2^{N_n^2}-1} P_i = 1-(1-\varepsilon)^{N_n^2}$$

Therefore, the probability of escaping from local optima in this case is $1-(1-\varepsilon)^{N_n^2}$. It should be mentioned that the probability of escaping toward the structures that have a shorter Hamming distance from sequence (111…1) is greater than the probability of escaping toward other structures. For example, in an FCM with 2 nodes, the probability of escaping from local optima toward the (1110), (1101), (1011) and (0111) structures is higher than the probability of escaping toward other structures, and equal to $(1-\varepsilon)^3\varepsilon$.

**Table 4** Results of applying paired T-test at significance level of 5%. Positive (+) sign shows that QFCM is better and significantly different than specified method

| Criteria | Number of nodes | Density | dMAGA [37] | MOEA-FCM [16] | SRCGA [15] | RCGA [12] | DD-NHL [38] |
|---|---|---|---|---|---|---|---|
| Data_Error | 5 | 20% | + | + | + | + | + |
| | | 40% | + | + | + | + | + |
| | 10 | 20% | + | + | + | + | + |
| | | 40% | + | + | + | + | + |
| | 20 | 20% | + | + | + | + | + |
| | | 40% | + | + | * | + | + |
| | 40 | 20% | + | + | + | + | + |
| | | 40% | + | + | + | + | + |
| out of sample error | 5 | 20% | N/A | + | + | + | + |
| | | 40% | N/A | + | + | + | + |
| | 10 | 20% | N/A | + | + | + | + |
| | | 40% | N/A | – | + | + | + |
| | 20 | 20% | N/A | + | + | + | + |
| | | 40% | N/A | * | + | + | + |
| | 40 | 20% | N/A | * | + | + | + |
| | | 40% | N/A | + | + | + | + |

## 4 Experimental results

In this section, synthetic, real-life, and DREAM3 as well as DREAM4 benchmark datasets [20–22] are used to assess the performance of the QFCM algorithm and to compare it with that of other newly devised algorithms. These experiments evaluate the ability of QFCM algorithm in providing dynamic and static analyses. Using these tasks to compare the efficacy of algorithms is so common in the literature [16].

In all the performed tests, the free parameters of the QFCM algorithm were tuned according to Table 2 in order to demonstrate the robustness of the QFCM model against the settings of free parameters.

### 4.1 QFCM performance on the synthetic dataset

To obtain the synthetic data, first, a reference FCM with a random weight matrix and a specific density is created. Weights of the reference FCM should be greater or less than +0.05 and − 0.05 respectively. This FCM is simulated with an initial random state vector, and then the response of the network is calculated by Eq. 3. This response sequence forms the synthetic data.

The objective function of optimization, which is also used in [15, 16], can be expressed as Eq. 11.

$$f(w) = \frac{-1}{1 + \eta Data\_Error} \tag{11}$$

where, $Data\_Error$ is defined according to Eq. 12, and $\eta$ is a constant whose value has been considered as 10,000, in order to have a fair comparison with other works. The reason for using the auxiliary function $f$ is to have a nonlinear function for rewarding the better solutions [15].

$$Data\_Error = \frac{1}{(k-1)N_n N_s} \sum_{q=1}^{N_s} \sum_{n=1}^{N_n} \sum_{t=1}^{k-1} \left| C_n^q(t) - \overline{C}_n^q(t) \right|^2 \tag{12}$$

where, $k$ is the number of data points, $N_n$ is the number of nodes, $N_s$ is the number of existing data sequences, $C_n^q(t)$ is the real value of data in the $q^{th}$ sequence for the $n^{th}$ node and $t^{th}$ state value, and $\overline{C}_n^q(t)$ is the value obtained from one-iteration simulation of the learned FCM using the initial state of $C_n^q(t-1)$ for the $q^{th}$ sequence, $n^{th}$ node, and $t^{th}$ state value.

In addition to the Data_Error, the following evaluation criteria are employed for measuring the efficacy of algorithms:

**Out-of-sample error** This criterion shows the ability of generalization of an algorithm. In order to calculate this measure, 10 random initial state vectors, which have not been used in the training phase, are produced and used to simulate the learned FCM. After simulation, the out-of-sample error criterion can be obtained according to:

$$out \quad of \quad sample \quad error = \frac{1}{(k-1)N_n N_s} \times \sum_{q=1}^{N_s} \sum_{n=1}^{N_n} \sum_{t=1}^{k-1} |\bar{C}_n^q(t) - \hat{\bar{C}}_n^q(t)| \tag{13}$$

where, $\bar{C}_n^q(t)$ indicates the simulated value for the $n^{th}$ node, the $t^{th}$ state value in the reference FCM which has started from the

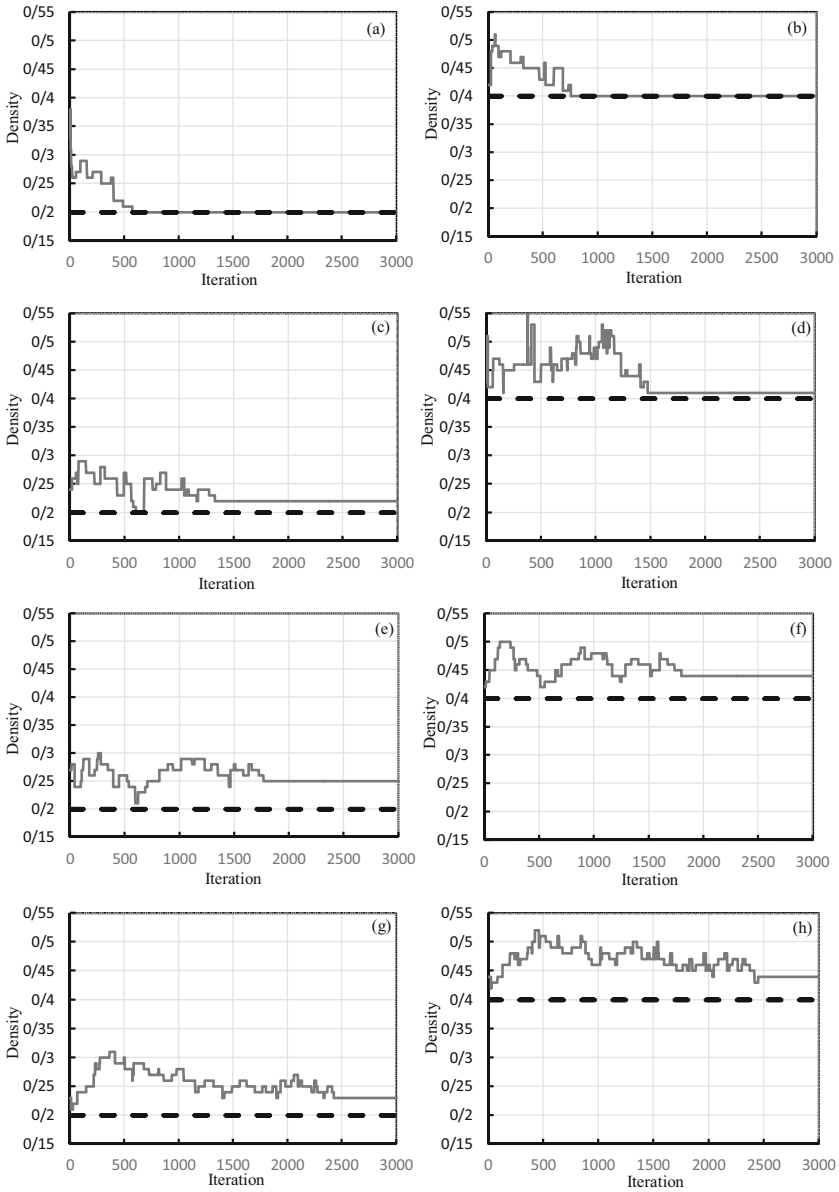


**Fig. 7** The variations of density with the number of iterations in FCMs with (a) 5 nodes and 20% density, (b) 5 nodes and 40% density, (c) 10 nodes and 20% density, (d) 10 nodes and 40% density, (e) 20 nodes and 20% density, (f) 20 nodes and 40% density, (g) 40 nodes and 20% density and (h) 40 nodes and 40% density

$q^{th}$ initial state vector; and $\widehat{\widehat{C}}_n^q(t)$ shows the simulated value for the $n^{th}$ node, the $t^{th}$ state value in the learned FCM which has started from the $q^{th}$ initial state vector.

**Matrix error** This error measures the difference between the weight matrix of the reference FCM and that of the learned FCM.

$$Matrix \quad Error = \frac{1}{N_n^2} \sum_{j=1}^{N_n} \sum_{i=1}^{N_n} |w_{ij} - \hat{w}_{ij}| \tag{14}$$

in which, $w_{ij}$ denotes the weight of the link from the $i^{th}$ node to the $j^{th}$ node in the reference FCM, and $\hat{w}_{ij}$ indicates the weight of the link from the $i^{th}$ node to the $j^{th}$ node in the learned FCM.

**SSmean** If the problem of FCM training is regarded as a classification problem (the class of zero weights versus the class of non-zero weights), the notions of 'true positive', 'false positive', 'true negative' and 'false negative' can be defined for this problem:

TP: The number of weights that are zero in the reference FCM and also in the learned FCM.
FP: The number of weights that are non-zero in the reference FCM, but zero in the learned FCM.
TN: The number of weights that are non-zero in the reference FCM and also in the learned FCM.
FN: The number of weights that are zero in the reference FCM, but non-zero in the learned FCM.

**Table 5** The results of the real-life datasets (average and standard deviations)

| Model | Learning algorithm | Data_Error | Out of sample error | Matrix error | SSmean |
|---|---|---|---|---|---|
| Plant supervisory process system [39] | QFCM | **0.001(0.004)** | **0.005(0.013)** | 0.114(0.128) | **0.75** |
| | MOEA-FCM | 0.003(0.004) | 0.008(0.120) | 0.117(0.126) | 0.73 |
| | SRCGA | 0.005(0.005) | 0.127(0.120) | **0.112(0.149)** | 0.71 |
| | D&C SRCGA | 0.006(0.005) | 0.129(0.122) | 0.119(0.118) | 0.69 |
| | RCGA | 0.005(0.005) | 0.131(0.141) | 0.395(0.348) | 0.16 |
| | D&C RCGA | 0.007(0.005) | 0.136(0.123) | 0.358(0.325) | 0.16 |
| | DD-NHL | 0.187(0.184) | 0.189(0.197) | 0.438(0.316) | 0.13 |
| Deforestation in Brazilian Amazon [40] | QFCM | 0.0051(0.003) | **0.0124(0.0137)** | **0.184(0.142)** | **0.69** |
| | MOEA-FCM | 0.0074(0.0040) | 0.0129(0.0134) | 0.2032 | 0.68 |
| | RCGA | **0.0048(0.0033)** | 0.1370(0.1529) | – | – |
| | D&C RCGA | 0.1138(0.0972) | 0.1471(0.1408) | – | – |
| | DD-NHL | 0.2257(0.2406) | 0.2213(0.1803) | – | – |
| Educational software adoption [41] | QFCM | **0.0063(0.0013)** | **0.0109(0.0111)** | 0.1073 | 0.21 |
| | MOEA-FCM | 0.0071(0.0010) | 0.0113(0.0118) | **0.1070** | **0.22** |
| | RCGA | 0.0085(0.0064) | 0.1539(0.1608) | – | – |
| | D&C RCGA | 0.1315(0.1369) | 0.1825(0.1586) | – | – |
| | DD-NHL | 0.2125(0.2138) | 0.2081(0.2096) | – | – |

So, SSmean can be expressed as

$$SSmean = \frac{2 \times sensitivity \times specificity}{sensitivity + specificity} \tag{15}$$

where 'sensitivity' and 'specificity' can be computed as

$$sensitivity = \frac{TP}{TP + FN} \tag{16}$$

$$specificity = \frac{TN}{TN + FP} \tag{17}$$

The 'sensitivity' and the 'specificity' represent the precision of the training algorithm in determining the weights with zero and non-zero values, respectively. Also, SSmean is a number in the interval [0, 1]; with larger values indicating better results.

The synthetic data for FCMs with various sizes and densities are generated and by using these data, we attempted to train the FCMs by applying the QFCM algorithm. This was repeated 5 times, and the average and standard deviation of each evaluation measure were reported in Table 3. Also, for comparison, the results of 7 recent and well-known FCM learning algorithms, called dMAGA [37], MOEA-FCM [16], SRCGA [15], RCGA [12], and DD-NHL [38] have been reported in Table 3 (the learning algorithms have been arranged from top to bottom, based on their year of presentation).

As it is observed in Table 3, in all the scenarios, the QFCM algorithm has been able to converge to the real density or to a density close to the real density; while the MOEA-FCM [16] or SRCGA [15] do not show such a capability and their authors, regardless of the real density of the network, have used 37% as the reported density. dMAGA [37], like QFCM, is able to converge to a real density, but since the authors didn't report their achieved densities, we didn't report them likewise.

In terms of Data_Error, showing the ability of algorithms in providing dynamic analysis, QFCM has shown lower error as compared to all other methods. The only exception is the 20-node FCM with 40% density, in which SRCGA [15] has shown better results. This can be explained by the usage of continuous genetic algorithm in the SRCGA [15], since it has shown shining results in the complex system designs [17]. Regarding the out of sample error, which illustrate the generalization ability of the algorithms, the QFCM outdo the other algorithms in 5 out of 8 cases. MOEA-FCM [16] has shown slightly better results in the remaining 3 cases. Nevertheless, in the 40-node FCM with 40% density, which has the hugest search space among all other FCMs and thus prone to lower generalization ability, the QFCM achieved better result. As for Matrix error, the QFCM has shown better results in the 5-node as well as 10-node FCMs (4 cases). dMAGA [37] achieved better results in FCMs with 20% density (2 cases) while and SRCGA [15] obtained more desirable results in FCMs with 40% density (2 cases). Concerning SSmean criterion, which directly states the superiority of algorithms in providing dynamic analysis, the QFCM algorithm has been able to achieve higher SSmean values than all the other methods in most of the cases. There are only three cases in which the QFCM has achieved a lower SSmean relative to the SRCGA [15] and dMAGA [37] method: the 10-node FCM with 20% and 40% density, and the 40-node FCM with 40% density.

**Table 6** Results of different algorithms on the DREAM3 and DREAM4 benchmark datasets for GRN reconstruction task

| Dataset-number of genes-topology number | QFCM | dMAGA [36] | RCGA [11] | D&C RCGA [42] |
|---|---|---|---|---|
| DREAM3–10-1 | **0.009** | 0.012 | 0.112 | 0.068 |
| DREAM3–10-2 | **0.009** | 0.011 | 0.100 | 0.079 |
| DREAM3–10-3 | **0.011** | 0.014 | 0.056 | 0.038 |
| DREAM3–10-4 | **0.013** | 0.014 | 0.107 | 0.036 |
| DREAM3–10-5 | **0.013** | **0.013** | 0.066 | 0.089 |
| DREAM3–50-1 | **0.039** | 0.044 | 0.138 | 0.111 |
| DREAM3–50-2 | 0.027 | **0.023** | 0.136 | 0.145 |
| DREAM3–50-3 | 0.35 | **0.030** | 0.109 | 0.117 |
| DREAM3–50-4 | **0.027** | 0.032 | 0.147 | 0.139 |
| DREAM3–50-5 | **0.021** | 0.029 | 0.158 | 0.136 |
| DREAM3–100-1 | 0.162 | **0.156** | 0.175 | 0.229 |
| DREAM3–100-2 | **0.097** | 0.103 | 0.207 | 0.241 |
| DREAM3–100-3 | 0.149 | **0.140** | 0.249 | 0.235 |
| DREAM3–100-4 | **0.129** | 0.138 | 0.195 | 0.237 |
| DREAM3–100-5 | 0.152 | **0.141** | 0.233 | 0.260 |
| DREAM4–10-1 | **0.001** | 0.006 | 0.079 | 0.090 |
| DREAM4–10-2 | **0.004** | 0.008 | 0.104 | 0.074 |
| DREAM4–10-3 | **0.009** | 0.013 | 0.124 | 0.079 |
| DREAM4–10-4 | **0.005** | 0.008 | 0.103 | 0.082 |
| DREAM4–10-5 | **0.016** | 0.020 | 0.066 | 0.099 |
| DREAM4–100-1 | 0.110 | **0.081** | 0.181 | 0.143 |
| DREAM4–100-2 | **0.098** | 0.101 | 0.171 | 0.157 |
| DREAM4–100-3 | 0.101 | **0.095** | 0.158 | 0.177 |
| DREAM4–100-4 | 0.095 | **0.086** | 0.161 | 0.150 |
| DREAM4–100-5 | **0.082** | 0.089 | 0.162 | 0.153 |

These two cases can be justified by the fact that in [15], the density estimate parameter has been considered equal to the density of the synthetic data produced and, therefore, there is an information leakage from the generated synthetic data to the training algorithm. While in the QFCM, such an information leakage does not exist.

As it is mentioned before, lower Data_Error and higher SSmean mean better dynamic and static analyses properties respectively; hence, it is feasible that any given algorithm in the Table 3 which simultaneously has higher SSmean and lower Data_Error has the ability of better simultaneous static and dynamic analyses. According to the Table 3, the QFCM simultaneously has lower Data_Error and higher SSmean in 4 cases. While, none of the other algorithms have shown such capability. This matter confirms our first claim about the QFCM's power in providing simultaneous static and dynamic analyses.

We have applied paired t-test, a statistical hypothesis test, for investigation of significant difference existence between QFCM algorithm's results and those of Table 3. The results of this test have been presented in Table 4. This table reveals that in terms of Data_Error, QFCM has achieved significantly better results than all other methods. The only exception is the SRCGA [15] which obtained significantly better results for the FCM with 20 nodes and 40% density. In terms of out of sample error, QFCM results are significantly better at most of the cases. Even though MOEA-FCM [16] achieved a lower out of sample error for the FCM with 10 nodes and 40% density, there is not any significant difference between it and QFCM.

Negative (−) sign shows insignificant difference. Star (*) sign shows that specified method is better and significantly different than QFCM.

**Table 7** Baseline values for SSmean and matrix error measures

| Number of nodes | Density (%) | SSmean | Matrix error |
|---|---|---|---|
| 5 | 20 | 0.286 | 0.155 |
| | 40 | 0.49 | 0.33 |
| 10 | 20 | 0.36 | 0.20 |
| | 40 | 0.46 | 0.34 |
| 20 | 20 | 0.29 | 0.18 |
| | 40 | 0.45 | 0.35 |
| 40 | 20 | 0.29 | 0.18 |
| | 40 | 0.45 | 0.35 |

The diagrams of the changes of learned FCMs' densities, obtained by the QFCM algorithm, with the number of iterations can also illustrate some aspects of the FCM training. These graphs have been plotted in Fig. 7 for all the 8 existing FCMs in Table 3 (dashed line indicates the real density and solid line indicates the learned FCM density in every iteration). This figure illustrates 2 aspects of the QFCM:

1. It is clear in all the cases convergence has occurred. As the number of nodes increases, so is the search space, we have seen delayed convergence. This delayed convergence is normal in that the larger the search space is, the more difficult it becomes for the QFCM to locate the real links of the network.
2. During the training phase, in none of the cases, the density of the learned FCM has got lower than the real density (the solid lines are always above the dashed lines). This confirms that the denser FCMs have a smaller error than the sparser FCMs. In fact, the QFCM algorithm, by modifying the quantum states during training, automatically prevents the selection of points in the search space that are sparser than the real density. Moreover, in none of the examined scenarios, the difference from the real density, during training, has become larger than 37.5%.

### 4.2 QFCM performance on the real-life datasets

Three real-life datasets—namely 'plant supervisory process system' (with 9 nodes) [39], the 'model of deforestation in Brazilian Amazon' (with 12 nodes) [40], and the 'educational software adoption model' (with 24 nodes) [41]—which are proposed in [15] as benchmark dataset for FCM's learning algorithms have been used to evaluate the QFCM's performance. The process of training was repeated 10 times for everyone of the models, and the obtained averages and standard deviations were listed in Table 5. The outcomes of the DD-NHL [38], RCGA [12], SRCGA [15] and MOEA-FCM [16] methods along with the results of the divide and conquer strategy [42] have also been reported in this table, the authors of the dMAGA [37] didn't report their results regarding the aforementioned datasets.

As it is observed in Table 5, regarding the SSmean value, the QFCM algorithm has been able to achieve better results in all the models except for educational software adoption model [41], in which MOEA-FCM [16] has achieved slightly better result. Regarding the matrix error, the QFCM algorithm has been able to obtain better results for the deforestation model [40]; but for the other two models, it has slightly larger errors relative to the MOEA-FCM [16] and SRCGA [15] methods. As for the out-of-sample error, the QFCM algorithm has been able to get better results than all the other considered methods in all scenarios; and this shows the generalization capacity of the FCMs that have been trained by the QFCM algorithm. The Data_Error of QFCM is lower than that of the other approaches. The only exception is the deforestation model [40], in which the RCGA [12] algorithm has been able to display better results. In the 'educational software adoption' model [41], the QFCM has achieved an SSmean of 21%. Despite the fact that this result is near to that of the MOEA-FCM, the number 21%, by itself, does not indicate a favorable accuracy in static analysis. The reason for a lower SSmean in this model is that in [41], a density of only 8% has been considered for this FCM; and due to the number of nodes in this model (24 nodes), the correct allocation of weights is a difficult task for the QFCM. Thus, it can be concluded that the quality of static analysis (and not dynamic analysis) diminishes in severely sparse FCMs.

All in all, according to the Table 5, the QFCM has shown more favorable results in 8 out of 12 evaluation scenarios.

### 4.3 QFCM performance in the gene regulatory network reconstruction problems

FCMs have shown promising results in the GRN reconstruction problems [37]. DREAM3 and DREAM4 benchmark datasets [20–22] are widely used for the evaluation of the algorithms attempting to reconstruct GRNs. The time series which are provided by them are recorded from real and known GRNs.

DREAM3 has given the time series of networks with 10, 50, and 100 genes, each has 5 different topologies. It recorded under different perturbations and overall, each topology provides with a 21-point time series. DREAM4 dataset has provided the time series of 2 networks with 10 and 100 genes. Similar to the DREAM3, DREAM4 has set 5 different topologies for each of the networks.

Here, we intended to reconstruct GRNs by applying the QFCM on DREAM3 as well as DREAM4 datasets. Also, we compare our results with dMAGA [37], RCGA [12], and D&C RCGA [42] algorithms. Since, previous works reported their achieved results in terms of the Data_Error, we did likewise. Table 6 shows the obtained results.

As it is plain in Table 6, QFCM achieved lower errors in most of the cases. Specifically, it has shown more desirable results in 17 out of 25 cases. Regarding 10-node networks, QFCM outdo all other algorithms in all scenarios. When the network has 50 nodes, QFCM obtained better results except for DREAM3–50-2 and DREAM3–50-3 datasets, in which dMAGA [37] achieved more favorable results. In 100-node networks, QFCM achieved better results in 2 cases, that is, DREAM4–100-2 and DREAM4–100-5.

### 4.4 Comparison with baseline

To better evaluate the power of static analysis in the FCMs trained by the QFCM algorithm, the baseline values of SSmean and matrix error for FCMs with 5, 10, 20 and 40 nodes and 20 as well as 40% densities were obtained and reported in Table 7. In order to obtain the baseline, FCMs with random weights and 20 and 40% densities (density limit) were produced for 10 times. Then by using Eq. 14 and Eq. 15, the matrix errors and SSmean values of these 10 FCMs were computed. Finally, the baseline was obtained by averaging these values. The values in Table 7 are not related to the random method; because in obtaining these values, there is an information leakage from train set to the learned FCMs.

As shown in Table 7, the QFCM algorithm has been able to improve the SSmean in the 40-node FCM by 47% (20% density) and 15% (40% density) and the matrix error of the 40-node FCM by 23% (20% density) and 32% (40% density); and this improvement has increased with the reduction of the number of nodes. For example, in the 20-node FCM with 20 and 40% densities, the SSmean improvements have been 54 and 33% and the improvements of the matrix error have reached 27 and 42%, respectively.

## 5 Conclusion

The structural simplicity, the adaptability and the deduction power of FCM have made it a powerful and versatile tool for researchers. Despite their widespread popularity, the FCMs created based on the opinions of experts and the FCMs based on evolutionary algorithms still limited in some ways such as the inability of simultaneous static and dynamic analyses properties. In this paper, a new training algorithm called the QFCM, which operates totally automatically, has been presented. The proposed QFCM uses the quantum inspired evolutionary algorithm coupled with PSO algorithm for providing static and dynamic analyses capabilities. Another achievement of this paper is that the FCMs trained by the QFCM converge to the real density of the network or to a density close to the real density. Using the probabilistic representation in encoding the FCM links leads to the survival of good structures, albeit with low dynamic analysis capability, in the course of different generations. This aspect has been examined closely in the paper through a mathematical theorem. The performance of the QFCM algorithm has been evaluated in synthetic, real-life, and gene regulatory network reconstruction datasets. The empirical results indicate that in most cases, the QFCM algorithm, without the involvement of any expert, can make more accurate FCMs than similar training algorithms.

## References


1. Kosko B (1986) Fuzzy cognitive maps. Int J Man Mach Stud 24: 65–75
2. Amirkhani A, Mosavi MR, Mohammadi K, Papageorgiou EI (2016) A novel hybrid method based on fuzzy cognitive maps and fuzzy clustering algorithms for grading celiac disease. Neural Comput Appl. https://doi.org/10.1007/s00521-016-2765-y
3. Zdanowicz P, Petrovic D (2018) New mechanisms for reasoning and impacts accumulation for rule-based fuzzy cognitive maps. IEEE Trans Fuzzy Syst 26:543–555. https://doi.org/10.1109/TFUZZ.2017.2686363
4. Papageorgiou EI, Poczęta K (2017) A two-stage model for time series prediction based on fuzzy cognitive maps and neural networks. Neurocomputing 232:113–121. https://doi.org/10.1016/J.NEUCOM.2016.10.072
5. Amirkhani A, Shirzadeh M, Papageorgiou EI, Mosavi MR (2016) Fuzzy cognitive map for visual servoing of flying robot. 2016 IEEE international conference on fuzzy systems (FUZZ-IEEE): 1371–1376
6. Amirkhani A, Shirzadeh M, Papageorgiou EI, Mosavi MR (2016) Visual-based quadrotor control by means of fuzzy cognitive maps. ISA Trans 60:128–142. https://doi.org/10.1016/j.isatra.2015.11.007
7. Stach W, Kurgan LA, Pedrycz W (2005) A survey of fuzzy cognitive map learning methods. Issues soft Comput theory Appl 71–84
8. Shirzadeh M, Shojaeefard MH, Amirkhani A, Behroozi H (2019) Adaptive fuzzy nonlinear sliding-mode controller for a car-like robot. 2019 IEEE 5th international conference on knowledge-based engineering and innovation (KBEI), Tehran, Iran
9. Amirkhani A, Kolahdoozi M, Wang C, Kurgan L (2018) Prediction of DNA-binding residues in local segments of protein sequences with fuzzy cognitive maps. IEEE/ACM Trans Comput Biol Bioinform. https://doi.org/10.1109/TCBB.2018.2890261
10. Wu K, Liu J (2017) Learning large-scale fuzzy cognitive maps based on compressed sensing and application in reconstructing gene regulatory networks. IEEE Trans Fuzzy Syst 25:1546–1560. https://doi.org/10.1109/TFUZZ.2017.2741444
11. Kyriakarakos G, Dounis AI, Arvanitis KG, Papadakis G (2017) Design of a Fuzzy Cognitive Maps variable-load energy management system for autonomous PV-reverse osmosis desalination systems: a simulation survey. Appl Energy 187:575–584. https://doi.org/10.1016/J.APENERGY.2016.11.077
12. Stach W, Kurgan L, Pedrycz W, Reformat M (2005) Genetic learning of fuzzy cognitive maps. Fuzzy Sets Syst 153:371–401. https://doi.org/10.1016/j.fss.2005.01.009
13. Poczeta K, Kubus L, Yastrebov A (2019) Analysis of an evolutionary algorithm for complex fuzzy cognitive map learning based on graph theory metrics and output concepts. Biosystems. https://doi.org/10.1016/j.biosystems.2019.02.010
14. Stach W, Kurgan L, Pedrycz W (2010) In: Glykas M (ed) Expert-based and computational methods for developing fuzzy cognitive maps BT - fuzzy cognitive maps: advances in theory, methodologies, tools and applications. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 23–41
15. Stach W, Pedrycz W, Kurgan LA (2012) Learning of fuzzy cognitive maps using density estimate. IEEE Trans Syst Man, Cybern Part B 42:900–912. https://doi.org/10.1109/TSMCB.2011.2182646
16. Chi Y, Liu J (2016) Learning of fuzzy cognitive maps with varying densities using a multiobjective evolutionary algorithm. IEEE Trans Fuzzy Syst 24:71–81. https://doi.org/10.1109/TFUZZ.2015.2426314
17. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197. https://doi.org/10.1109/4235.996017

18. Han K-H, Kim J-H (2002) Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Trans Evol Comput 6:580–593. https://doi.org/10.1109/TEVC.2002.804320
19. Kennedy J, Eberhart R (1995) Particle swarm optimization. Neural Netw 1995 Proc IEEE Int Conf 4:1942–1948
20. Marbach D, Prill RJ, Schaffter T et al (2010) Revealing strengths and weaknesses of methods for gene network inference. Proc Natl Acad Sci U S A 107:6286–6291. https://doi.org/10.1073/pnas.0913357107
21. Marbach D, Schaffter T, Mattiussi C, Floreano D (2009) Generating realistic in silico gene networks for performance assessment of reverse engineering methods. J Comput Biol 16:229–239. https://doi.org/10.1089/cmb.2008.09TT
22. Prill RJ, Marbach D, Saez-Rodriguez J et al (2010) Towards a rigorous assessment of systems biology models: the DREAM3 challenges. PLoS One 5:e9202. https://doi.org/10.1371/journal.pone.0009202
23. Najafi A, Amirkhani A, Papageorgiou EI, Mosavi MR (2017) Medical decision making based on fuzzy cognitive map and a generalization linguistic weighted power mean for computing with words. 2017 IEEE international conference on fuzzy systems (FUZZ-IEEE): 1–6
24. Tsadiras AK (2008) Comparing the inference capabilities of binary, trivalent and sigmoid fuzzy cognitive maps. Inf Sci (Ny) 178:3880–3894. https://doi.org/10.1016/j.ins.2008.05.015
25. Bueno S, Salmeron JL (2009) Benchmarking main activation functions in fuzzy cognitive maps. Expert Syst Appl 36:5221–5229
26. Amirkhani A, Papageorgiou EI, Mohseni A, Mosavi MR (2017) A review of fuzzy cognitive maps in medicine: taxonomy, methods, and applications. Comput Methods Prog Biomed 142:129–145. https://doi.org/10.1016/j.cmpb.2017.02.021
27. Hu H, Wang H, Bai Y, Liu M (2019) Determination of endometrial carcinoma with gene expression based on optimized Elman neural network. Appl Math Comput 341:204–214. https://doi.org/10.1016/J.AMC.2018.09.005
28. Du W, Zhang M, Ying W et al (2018) The networked evolutionary algorithm: a network science perspective. Appl Math Comput 338:33–43. https://doi.org/10.1016/J.AMC.2018.06.002
29. Lahmiri S (2018) Minute-ahead stock price forecasting based on singular spectrum analysis and support vector regression. Appl Math Comput 320:444–451. https://doi.org/10.1016/J.AMC.2017.09.049
30. Sierra MR, Coello Coello CA (2005) In: Coello Coello CA, Hernández Aguirre A, Zitzler E (eds) Improving PSO-based multi-objective optimization using crowding, mutation and ∈−dominance BT - evolutionary multi-criterion optimization. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 505–519
31. Ganjefar S, Tofighi M, Karami H (2015) Fuzzy wavelet plus a quantum neural network as a design base for power system stability enhancement. Neural Netw 71:172–181. https://doi.org/10.1016/j.neunet.2015.07.010
32. Huang Z, Yang L, Jiang W (2019) Uncertainty measurement with belief entropy on the interference effect in the quantum-like Bayesian networks. Appl Math Comput 347:417–428. https://doi.org/10.1016/J.AMC.2018.11.036
33. Lu TC, Yu GR, Juang JC (2013) Quantum-based algorithm for optimizing artificial neural networks. IEEE Trans Neural Netw Learn Syst 24:1266–1278. https://doi.org/10.1109/TNNLS.2013.2249089
34. Liu J, Sun J, Xu W (2006) In: Huang D-S, Li K, Irwin GW (eds) Improving quantum-behaved particle swarm optimization by simulated annealing BT - computational intelligence and bioinformatics. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 130–136
35. Han K-H, Kim J-H (2004) Quantum-inspired evolutionary algorithms with a new termination criterion, $H_{\ε}$ gate, and two-phase scheme. IEEE Trans Evol Comput 8:156–169. https://doi.org/10.1109/TEVC.2004.823467
36. Lv F, Yang G, Yang W et al (2017) The convergence and termination criterion of quantum-inspired evolutionary neural networks. Neurocomputing 238:157–167. https://doi.org/10.1016/j.neucom.2017.01.048
37. Liu J, Chi Y, Zhu C (2016) A dynamic multiagent genetic algorithm for gene regulatory network reconstruction based on fuzzy cognitive maps. IEEE Trans Fuzzy Syst 24:419–431. https://doi.org/10.1109/TFUZZ.2015.2459756
38. Stach W, Kurgan L, Pedrycz W (2008) Data-driven nonlinear Hebbian learning method for fuzzy cognitive maps. 2008 IEEE international conference on fuzzy systems (IEEE world congress on computational intelligence): 1975–1981
39. Stylios CD, Groumpos PP (1999) Fuzzy cognitive maps: a model for intelligent supervisory control systems. Comput Ind 39:229–238. https://doi.org/10.1016/S0166-3615(98)00139-0
40. Kok K (2009) The potential of fuzzy cognitive maps for semi-quantitative scenario development, with an example from Brazil. Glob Environ Chang 19:122–133. https://doi.org/10.1016/j.gloenvcha.2008.08.003
41. Hossain S, Brooks L (2008) Fuzzy cognitive map modelling educational software adoption. Comput Educ 51:1569–1588. https://doi.org/10.1016/j.compedu.2008.03.002
42. Stach W, Kurgan L, Pedrycz W (2010) A divide and conquer method for learning large fuzzy cognitive maps. Fuzzy Sets Syst 161:2515–2532. https://doi.org/10.1016/j.fss.2010.04.008