# Learning Fuzzy Cognitive Maps from Data by Ant Colony Optimization

Ye Chen
School of Electronic and Computing Systems
University of Cincinnati
2600 Clifton Ave., Cincinnati, OH 45221
chenye84@gmail.com

Lawrence J. Mazlack
Applied Computational Intelligence Laboratory
University of Cincinnati
2600 Clifton Ave., Cincinnati, OH 45221
mazlack@ucmail.uc.edu

Long J. Lu
Division of Biomedical Informatics
Cincinnati Children's Hospital Medical Center
3333 Burnet Ave., Cincinnati, OH 45229
Long.Lu@cchmc.org

## ABSTRACT
Fuzzy Cognitive Maps (FCMs) are a flexible modeling technique with the goal of modeling causal relationships. Traditionally FCMs are developed by experts. We need to learn FCMs directly from data when expert knowledge is not available. The FCM learning problem can be described as the minimization of the difference between the desired response of the system and the estimated response of the learned FCM model. Learning FCMs from data can be a difficult task because of the large number of candidate FCMs. A FCM learning algorithm based on Ant Colony Optimization (ACO) is presented in order to learn FCM models from multiple observed response sequences. Experiments on simulated data suggest that the proposed ACO based FCM learning algorithm is capable of learning FCM with at least 40 nodes. The performance of the algorithm was tested on both single response sequence and multiple response sequences. The test results are compared to several algorithms, such as genetic algorithms and nonlinear Hebbian learning rule based algorithms. The performance of the ACO algorithm is better than these algorithms in several different experiment scenarios in terms of model errors, sensitivities and specificities. The effect of number of response sequences and number of nodes is discussed.

## Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *heuristic methods.*

## General Terms
Algorithms

## Keywords
Fuzzy Cognitive Maps, Ant Colony Optimization, numerical optimization, data-driven learning algorithm

## 1. INTRODUCTION
Fuzzy Cognitive Maps (FCM) [1] are a modeling technique that is used in a variety of applications. A FCM is a graph consists of weighted edges and nodes. The nodes represent concepts. The edges are associated with weights to represent the strength of causal relations between two concepts. FCMs have several advantages, such as (a) the nodes in FCM represent real-world concepts and should be easier to be understood by human than other modeling techniques such as neural networks; (b) FCMs can represent cycles and recursive cycles. The traditional FCM learning approach is that experts build an initial network and then learning algorithms, such as Hebbian learning rule [2], can be used to improve the accuracy of the FCM. However, expert knowledge may not be available for all application areas for which FCMs might be desirable. For example, there is limited expert knowledge available if FCMs are used to describe the gene regulatory relation contains less studied genes. It follows that it may be difficult to build FCM models for gene regulatory networks as well as many other areas with only limited available expert knowledge. Consequently, it is useful to learn FCMs from data without prior knowledge.

Several computational intelligence algorithms have been proposed to learn FCMs directly from data. These computational intelligence algorithms include: genetic algorithm [3], particle swarm optimization [4], simulated annealing [5] as well as others. Stach [6] was able to solve problems with tens of nodes. The number of weights to be optimized increases quadratically as the number of nodes increases. The possible combination of different weights increases exponentially as the number of weights increases. Performance measured as out-of-sample data error [7] drops as the number of nodes increases. For practical problems, such as building gene regulatory network from gene expression data, the number of nodes ranges from several dozens to thousands; for these larger problems, previously used methods are computationally intractable. A more powerful learning algorithm is necessary to learn these networks.

Ant Colony Optimization (ACO) is a prominent algorithm for both combinatory and numerical optimization problems [8, 9]. It has been shown to outperform genetic algorithms, particle swarm optimizations and simulated annealing algorithms in many application areas [8, 10, 11]. It seems reasonable to consider applying ACO to the FCM learning problem.

An initial study has shown that ACO is capable of building an FCM model for a process control problem [12]. However, this prior research has several limitations:

(a) The learning algorithm was tested on a very small problem with only 5 nodes in the FCM model;

(b) The structure of the FCM model is pre-defined by domain experts and the number of un-determined weights is reduced;

(c) The ranges of several weights are pre-defined to reduce the size of solution space.

Because of the above three limitations, the dimension of the problem (i.e. the number of weights to be determined) is only 8. This is a relatively small dimension. Many recent optimization algorithms are able to solve problems with more than one hundred variables [10, 13].


*GECCO'12*, July 7–11, 2012, Philadelphia, Pennsylvania, USA.

In this paper, the ACO algorithm is used to build FCMs from data. The proposed algorithm is different from the previous ACO approach [12] in two aspects:

- First, the objective functions are different. The aim of Ding's approach is to maintain the values of the nodes in a reasonable interval and the objective function is to minimize the violation to the constraints on the range of several manually designated nodes [12]; while in our approach, the aim is to discover causal relations among the nodes from the observed data and the objective function is the difference between the observed value and estimated value of every node.
- Second, the scales of the problem are different. Ding's approach was applied to a problem with only 8 weights to be optimized; while in this paper, the proposed algorithm is applied to optimize 25 to 1600 weights.

The remainder of paper is organized as follows. Section 2 introduces FCMs and the problem of FCM learning. Section 3 introduces the proposed ACO based learning algorithm. Numerical experiments and results analysis are introduced in Section 4. Finally a conclusion is drawn in Section 5.

# 2. FUZZY COGNITIVE MAPS LEARNING

This section describes the basic concepts of FCM; briefly reviews FCM learning algorithms; and presents the mathematic formulation of the FCM learning problem used in this paper.

## 2.1 Fuzzy Cognitive Maps

Figure 1 shows an example of a FCM. A fuzzy cognitive map consists of $N_n$ concept nodes $C_i$ ($i$=1,2, …, $N_n$) and the weighted edges between pairs of nodes $w_{ij} \in [-1,1]$. Every node represents a concept. The values of the concept nodes represent the activation degree of the concept. The activation degree is a fuzzy variable ranges from 0 to 1. The weighted edges represent the causal relations between pairs of nodes. The weight is a positive (negative) number if one concept has a positive (negative) causal effect on another concept. If two concepts have no causal relation, the corresponding weight should be 0.

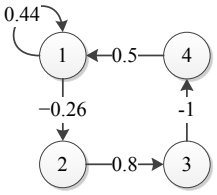


**Figure 1. An example of a FCM**

The dynamics of FCMs is determined by the following equation:

$$C_i^{(t+1)} = f\left(\sum_{i=1}^{N_n} w_{ij} C_i^{(t)}\right) \quad (1)$$

where $C_i^{(t)}$ is the activation degree of node $i$ in iteration $t$, $f(\cdot)$ is a transfer function that is used to bound the activation degree in the range of [0,1]. There are many different transfer functions. Usually the transfer function should be chosen according to the nature of the problem. However comparison study carried out by Tsadiras [14] suggests that sigmoid transfer functions are better than the other transfer functions in general. In this paper, sigmoid transfer functions will be used and it is defined as follows:

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

where $\lambda$ is a parameter used to characterize the steepness of the function around zero. $\lambda$ should be chosen according to the nature of the problem. Usually it is chosen by performing experiments on different values for $\lambda$ and the best value is chosen based on the performance of the overall model. In this paper, $\lambda$ is set to 5 because this is a widely used value in many FCM learning research [7]. Another reason is that the value of this parameter does not change the nature of the FCM learning problem. The performance of the algorithms test on different $\lambda$ value should be consistent. In practice, the value of $\lambda$ should be determined according to the property of the system to be modeled. Usually a small $\lambda$ is suitable for highly nonlinear system.

## 2.2 Data-driven Learning Algorithms

The learning of Fuzzy Cognitive Maps (FCM) from data is an important issue that has largely limited the application of FCM to many areas. For example, biologists need to study gene regulatory relations of genes related to diseases. In some cases, the mechanism of the diseases hasn't been well studied by biologists yet. They need to discover the gene regulatory relations from data. As there is only limited expert knowledge available, we need to learning regulatory networks from data.

Hebbian rule based learning algorithms [15] are designed to improve the accuracy of FCM generated by human experts. A good initial FCM model is required so that Hebbian rule based learning algorithms can be used to improve the initial FCM model. For some applications, domain knowledge is limited. It is hard to build a good initial FCM model by human experts for Hebbian rule based learning algorithms to improve.

Optimization algorithms based on computational intelligence were used to learn FCMs from data without domain knowledge.

Koulouriotis [16] proposed learning FCMs from data by using Evolution Strategy (ES). ES is similar to genetic algorithms except solutions in ES are represented by real numbers while solutions are represented in binary strings in genetic algorithms. ES algorithms are suitable for continuous domain optimization problems because the solutions are directly represented as real number. The problem of learning FCMs was stated as finding a suitable FCM so that for a given input the output should be the same as the observed data. Several input-output pairs are used. Evolution strategy algorithm is used to minimize the error between the output of the candidate FCM in the algorithm and the output that is observed from the to-be-modeled system.

Stach [17] proposed learning FCM from data by using the Real-Coded Genetic Algorithm (RCGA), an improved genetic algorithm for continuous domain problems. They assumed a sequence of values of every node (concept) has been observed from the to-be-modeled system. The sequence is broken down into several pairs. For example, assume the sequence is {$V(1)$, $V(2)$, $V(3)$}, where $V(t)$ is the values of all the nodes at time point $t$. This sequence implies that if the previous values of the nodes are $V(1)$, the values should be $V(2)$ in the next iteration. It is similar for the pair of $V(2)$ and $V(3)$. Therefore there will be two input-output pairs. The estimated values can be calculated with the following equation:

$$\hat{V}(n+1) = FCM(V(n)) \quad (3)$$

where $FCM(X)$ means the output of the FCM network given initial node value vector $X$. RCGA was used to minimize the errors between the estimated and the observed values. Stach [17] showed the advantage of RCGA over ES [16] through experiments.

A scalable divide and conquer based approach was proposed by Stach to learn large FCMs [6]. The division is performed on the response sequence that is being learnt from. It is divided into several small sequences. Each genetic algorithm focuses on minimizing the output error of one sequence. There will be

several FCMs after all the genetic algorithms have finished their calculation. Every FCM is assigned a weight as 1−objective function value. Final result FCM is calculated as the weighted average of the individual FCMs.

Parsopoulos [18] introduced an approach of learning FCM by particle swarm optimization (PSO). PSO is an algorithm based on moving a set of solution in the solution space towards the presently found best solution. It has been shown by experiments that PSO can obtain an accurate FCM model.

Alizadeh [19] proposed learning FCM by Tabu search. Tabu search is a search algorithm that can remember some of the solutions that has been evaluated. This is proposed to avoid repeatedly evaluation of the same solution, which often happens in genetic algorithms and other computational intelligence algorithms. It is applied to the same optimization model as [17].

Alizadeh [20] proposed a Simulated Annealing (SA) based FCM learning algorithm. The problem formulation is the same as [19], except the optimization algorithm is changed to SA.

Although many computational intelligence algorithms have been applied to FCM learning problems, relatively few studies have been done on the application of ACO to FCM learning problems. Furthermore, many of the FCM learning algorithms are applied to learn small scale FCM learning problems with less than 100 weights to be determined. As discussed in Section 1, an initial study on the application of ACO to FCM learning problems shows the applicability of ACO based learning algorithm on small scale problems. This paper focuses on relatively large problems. A FCM learning algorithm based on ACO is proposed and it is used to learn FCMs with 40 nodes (i.e. 1600 weights to be determined).

## 2.3 Problem Formulation

FCMs can be used to model different problems. Different mathematical formulations of FCM learning are used in different modeling problems. For example, FCM models were used to design a proper fuzzy controller in some of the researches [4, 12, 18]. In these works, the concept nodes are divided into internal state nodes and output nodes. The FCM learning problem is formulated as minimizing the difference between the desired output values and the steady state values of the output nodes. Another example is that FCM models are used to study the causal relations among the concept nodes [6, 17, 19]. In these works, all the concept nodes are considered as output nodes, i.e. the objective of the optimization algorithm is to minimize the difference between the desired value of every concept node and the estimated activation degree based on the learnt FCM model. Ideally, the result FCM model should be able to generate the same sequence data as the system under study. The problem addressed in this paper is similar to the second example. It is more challenging than the first one because there are more weights to be optimized in the second example.

In this paper, a FCM is isomorphically represented by a weight matrix $W$=[$w_{ij}$]. The FCM itself is a graph. It is represented in an isomorphic matrix form for the convenience of writing equations. To learn a FCM, ACO algorithm is used to minimize the following error function:

$$Data\ Error = \frac{1}{(N_t - 1)N_n N_s} \sum_{\substack{1 \le t \le N_t - 1 \\ 1 \le n \le N_n \\ 1 \le s \le N_s}} \left( C_n^{(t)}(s) - \hat{C}_n^{(t)}(s) \right)^2 \quad (4)$$

where $N_t$ is the length of the response sequence data, $N_s$ is the number of response sequences, $N_n$ is the number of nodes in the FCM network, $C_n^{(t)}(s)$ is the $t$-th value of node $n$ in the $s$-th observed data series. $\hat{C}_n^{(t)}(s)$ is the $t$-th value of node $n$ in the $s$-th data series generated by the FCM network.

# 3. LEARNING ALGORITHM

ACO is used to minimize the objective function, i.e. equation (4). ACO is first introduced to solve combinatory optimization problems [9, 21]. It is inspired from ants' behavior. Ants leave pheromone on their paths as they are moving between a food source and their nest. For the ants moving on short paths, they travel more times between the food source and the nest. Therefore more pheromone is left on the short paths. Ants tend to explore the space around the paths with high pheromone intensities. Over time, the ants will find a heuristically short path between the food source and the nest (the discovered path may or may not be optimal). In ACO algorithms, solutions to the problems are encoded as ant paths. Pheromone is used to guide the search process of the artificial ants. Because of the way the pheromone is stored, ACO algorithms are usually used in combinatory optimization problems.

Several extensions to ACO have been proposed to address continuous numerical optimization problems. One approach is to represent pheromone as Probability Density Functions (PDF) [10]. Another approach is to discretize the continuous space [11]. Both approaches have their advantages. The PDF approach represents pheromone intensities more accurate, while the discretization approach runs faster [11]. Discretization approach is used here.

## 3.1 Representation of the Candidate Solutions

The candidate solutions to the FCM learning problem are weight matrices. Since the problem is formulated as a numerical optimization problem, the candidate solutions are represented in vector form instead of the matrix form so that the notations can be fit into numerical optimization literature more easily. The vector form is

$$W_v = [w_{11}, w_{12}, ..., w_{1,N_n}, ..., w_{N_n,1}, w_{N_n,2}, ... w_{N_n,N_n}] \quad (5)$$

where $w_{ij}$ is the weight on the edge from node $i$ to $j$. The values in $W_v$ are continuous values. They are represented as several decimal numbers according to the required precision:

$$w_{ij} = [d_{ij1}, d_{ij2}, ..., d_{ijk}, ..., d_{ijN_d}] \quad (6)$$

where $d_{ijk}$ is the $k$-th digit for the weight $w_{ij}$, $d_{ijk}$ ranges from 0 to 9, is the number of digits used to represent one weight. The digits for all the $w_{ij}$'s consist a candidate solution, i.e. a candidate FCM model. The digits are converted to the weights according to the following equation:

$$w_{ij} = g\left( \sum_{k=1}^{N_d} d_{ijk} \times 10^{-k} \right) \quad (7)$$

where function $g(x)$ is used to map the values in interval [0,1) to the interval of the weights [−1,1). It is defined as follows:

$$g(x) = 2 \times (x - 0.5) \quad (8)$$

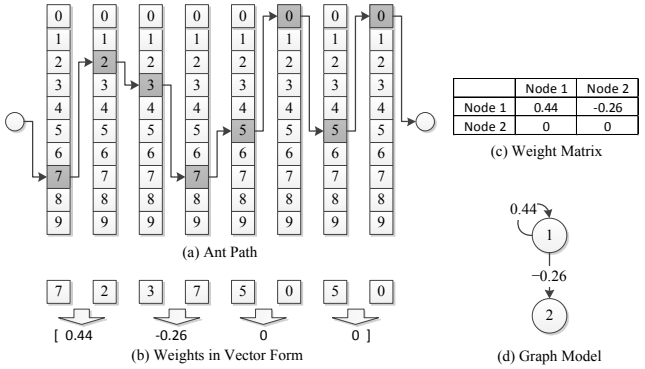


**Figure 2. Different representation forms of FCMs**

Examples of the different representation forms are shown in Figure 2. Ant paths are generated by ACO algorithm described in the following sub-sections. The ant paths are decoded according to equation (7) to obtain the vector form $W_v$. The weights in vector form can be rearranged according to the definition of $W_v$ to obtain the weight matrix. The vector form of the weight matrix is used in the ACO algorithm. The weight matrix is the internal form used in the calculation of activation levels of the FCMs. Both vector form and matrix form are equivalent. We distinguish them because in the literature of optimization, variables to be optimized are usually represented in vector form, while the matrix form is usually used in the FCM simulations. The graph model is usually used as a user friendly representation form of FCMs.

## 3.2 Storage of Pheromone Intensities

As shown in Figure 2(a), the ants begin their search from a virtual starting node (the left-most circle). In every decision step, the ants will choose one node (the squares marked with numbers) from the next column of nodes. The pheromone intensities are stored on the edges between any two nodes in adjacent columns. The pheromone intensities are stored in a three-dimension matrix $\boldsymbol{\tau} = [\tau_{cn_1n_2}]$, where $\tau_{cn_1n_2}$ is the pheromone intensity stores on the edge from node $n_1$ in the $c$-th column to node $n_2$ in the ($c$+1)-th column. Both $n_1$ and $n_2$ range from 0 to 9. The initial values for $\boldsymbol{\tau}$ is set to the parameter $\tau_0$. This parameter is usually chosen by experience. In this paper, experiments are performed on different values for $\tau_0$ and then use the parameter that generates the minimum data error.

## 3.3 Ant Colony Optimization Algorithm

The ACO algorithm consists of several steps include (a) building ant tours; (b) decoding the ant tours to obtain candidate FCMs (in the form of weight matrices); (c) evaluate the candidate FCMs; and (d) update the pheromone intensities. These steps will be performed many times until either a satisfactory solution is found or the maximum number of iteration is reached.

### *3.3.1 Build ant tours*

Usually there are several ants building individual tours simultaneously. Each ant starts from the virtual initial node and chooses one node from the first column of nodes. Then the ants start from their nodes in current column and choose their next nodes in the next column iteratively. The node in the next column is chosen according to the following equation:

$$T_{mc} = \begin{cases} argmax_{n_2}\left(\tau_{c-1,T_{m,c-1},n_2}\right) & ,q > p_0 \\ S_r & ,q \le p_0 \end{cases} \quad (9)$$

where $T_{mc}$ is the node the $m$-th ant chose in the $c$-th column, $\tau_{c-1,T_{m,c-1},n_2}$ is the pheromone intensity from node $T_{m,c-1}$ (i.e. the node the $m$-th ant chose in the ($c$-1)-th column) to node $n_2$ in the $c$-th column. $argmax_{n2}(\tau_{c-1,T_{m,c-1},n_2})$ is a function returns the value of $n_2$ which maximize the value of $\tau_{c-1,T_{m,c-1},n_2}$, $q \in [0,1]$ is a random number generated every time this equation is applied, $p_0$ is a pre-defined parameter, $S_r$ is the node chosen by pseudo-random proportional rule. In this rule, a node is chosen randomly according to the probabilities defined as below:

$$P_{cn_2} = \frac{\tau_{c-1,T_{m,c-1},n_2}}{\sum_{n=0}^{9} \tau_{c-1,T_{m,c-1},n}} \quad (10)$$

where $P_{cn2}$ is the probability of node $n_2$ in column $c$ being chosen. The probability is proportional to the corresponding pheromone intensity.

### *3.3.2 Local pheromone update*

Local pheromone update performs as the ants move. In this step, the pheromone intensities on the ants' path will be decreased in order to encourage the ants exploring more paths. After the $m$-th ant chose a new node $T_{mc}$, the pheromone intensity on the edge from the ant's previous choice $T_{m,c-1}$ to its new choice $T_{mc}$ will be updated according to the following equation:

$$\tau_{c-1,T_{m,c-1},T_{mc}} = (1-\rho) \times \tau_{c-1,T_{m,c-1},T_{mc}} + \rho\tau_0 \quad (11)$$

where $\rho \in [0,1]$ is a parameter represents the decay rate of pheromone.

### *3.3.3 Evaluate ant tours*

After all the ants have chosen one node in every column, the choices of the ants are decoded to obtain FCM models as described in Section 3.1. The values of the objective function for the FCM models are calculated according to Section 2.3. The ant with minimum objective function value is called iteration-best ant. If the iteration-best ant in current iteration has a smaller objective function value than the iteration-best ant in any previous iteration, it is called global-best ant and its tour is denoted as

$$T^* = \left[T_1^*, T_2^*, ..., T_{N_c}^*\right] \quad (12)$$

where $N_c$ is the number of columns.

### *3.3.4 Global pheromone update*

Global pheromone update is performed after all the ants have been evaluated. The pheromone intensities on the best paths are rewarded. The update equation is shown as below:

$$\tau_{c,T_c^*,T_{c+1}^*} = (1-\alpha) \times \tau_{c,T_c^*,T_{c+1}^*} + \alpha / f_{fit}^* \quad (13)$$

where $\alpha$ is a parameter reflects the strength of the global update rule, $f_{fit}^*$ is the fitness of the global-best ant. In this paper, objective function value is used as the fitness except when objective function equals zero. In order to avoid the error of being divided by 0, the objective function value is set to a very small number such as $10^{-6}$ when the objective function is zero.

### *3.3.5 Pseudo-code*

The pseudo-code of the algorithm is listed as follows:

**Algorithm 1:** FCM learning algorithm based on ACO

**Input:** desired response sequence data $C_n^{(t)}(s)$.
**Output:** the best FCM model found.

```
Initialize the parameters, pheromone intensities;
while Maximum number of iteration is not reached do
    if this is the first iteration
        Assign random initial tours to the ants;
    else
        for m = 1 to N_a do
            Build tour for ant m according to Section 3.3.1;
            Update pheromone intensity of ant m's tour
              according to Section 3.3.2;
        end for
    end if
    Decode the ant tours into FCM models according to
      Section3.1;
    Calculate objective function values for the FCM models
      according to the problem formulation in Section 2.3;
    Update the pheromone intensity on the global best ant
      tour according to Section 3.3.4;
end while
Report the results.
```

In this pseudo-code, the ACO algorithm stops while the number of iterations reached a pre-defined maximum allowed number. This stop criterion is used because it is simply to compare algorithms if the number of objective function calculation (which equals number of iterations multiplied by number of ants) is the same. In practical problems, algorithm can be stopped if the solution did not improve over a number of iterations.

# 4. EXPERIMENTS AND RESULTS

## 4.1 Test Data Generation

Simulated data is used to test our algorithm. An FCM model is first generated randomly and then the data sequences are generated according to the FCM model.

The FCM model is generated by assigning random values to the weight matrix in the range of −1 to 1. In order to control the density of the random FCM model, a fraction of the edges are chosen according to the desired density. For example, if a density of 40% is expected for a 5-node FCM, 10 random edges (the total number of edges is 5×5=25, 40% of 25 edges is 10) will be selected and random values will be assigned to these edges. After the target edges are selected, random values are assigned to these edges. The value will be set to 0 if the absolute value of the generated random number is less than 0.05. This is because causal relation with strength less than 0.05 usually has no significance in practical problems [17].

The data sequences are generated by simulating the randomly generated FCM model with random initial concept activation degrees. The detail process is described in the following:

(a) Generate random initial values for all the nodes;

(b) Use equation (1) to calculate the values of the nodes in the next time points;

(c) Repeat step (b) until the length of the sequence reaches the desired value $N_t$;

(d) Repeat step (a) to (c) until the number of sequences reaches the desired value $N_s$.

## 4.2 Parameters

There are two categories of parameters: (a) problem related parameters, which are determined by the problem. For example, number of nodes $N_n$ is determined by the number of concepts need to be modeled; number of digits $N_d$ is determined according to the required precision of the weights; number of columns is determined by both $N_n$ and $N_d$: $N_c=N_n^2\times N_d$. (b) ACO algorithm related parameters. These parameters affect the performance of the algorithm and the relation between these parameters and the performance is complicated. In this paper, some of the parameters are chosen by grid search. As there are 6 parameters need to be determined, it is not time-efficient to search all the parameters. Even if only 10 different numbers are considered for every parameter, there would be 1,000,000 combinations. It will require too much computation. Some of the parameters are chosen based on experience and some of them by grid search. The initial pheromone intensity is chosen to be

$$\tau_0 = 1/f'_{fit} \tag{14}$$

where $f'_{fit}$ is an estimated objective function value of a local best solution. In the ACO algorithm for Travelling Salesman Problem (TSP), $f'_{fit}$ is calculated by nearest neighbor search. In the ACO algorithm for FCM learning, $f'_{fit}$ is chosen by a random search. 100 random FCM models are generated and evaluated. Finally $f'_{fit}$ is set to be a value slightly smaller than the local best objective function value.

As the parameters $\alpha$, $\rho$ and $p_0$ are related to the balance of exploration and exploitation, these three parameters are chosen by grid search. The other parameters are chosen based on experience. The other parameters are changed slightly around the chosen values to make sure it is at least a locally best parameter choice.

Number of iterations affects the number of objective function calculations. In order to compare with the other algorithms, the number of objective function calculations is kept the same. Stach compared several algorithms with 3,000,000 times of objective function calculations [7]. The maximum number of iterations is set to the number of objective function calculations divided by the number of ants.

The parameter settings are list in Table 1.

**Table 1. Parameter settings**

| Parameter | Description | Value |
|---|---|---|
| $N_d$ | Number of digits | 2 |
| $N_n$ | Number of nodes | 20* |
| $N_c$ | Number of columns | 800* |
| $N_{iter}$ | Maximum number of iterations | 60,000 |
| $N_a$ | Number of ants | 50 |
| $\alpha$ | Strength of global update | 0.4 |
| $\rho$ | Pheromone decay rate | 0.4 |
| $p_0$ | The probability of choosing the digit numbers deterministically | 0.9 |
| $\tau_0$ | Initial pheromone intensity | 0.01 |

* These two parameter values are chosen for the problem of modeling the relation among 20 concepts. The values should be re-calculated if the number of concepts was changed.

## 4.3 Performance Measures

The performance is measured by model error, specificity, sensitivity, the mean value of specificity and sensitivity, and receiver operating characteristics.

### *4.3.1 Model error*

Model error is used to measure the difference between the weight matrix of the learned FCM model and the weight matrix of the target FCM model. It is defined as follows.

$$Model\ Error = \frac{1}{N_n^2}\sum_{i=1}^{N_n}\sum_{j=1}^{N_n}\left|w_{ij}-\hat{w}_{ij}\right| \tag{15}$$

where $\hat{w}_{ij}$ is the estimated weight from node $i$ to node $j$. Standard deviation of model error across different experiments is used to measure the stability of the proposed algorithm.

### *4.3.2 Specificity and sensitivity*

For many application areas such as building gene regulatory networks, the goal of building network models is to identify plausible causal relation among genes. The learning algorithm is required to predict the existence of an edge between two nodes. In order to evaluate the performance of the proposed FCM learning algorithm for edge predicting problem, the FCM learning problem is extended and transformed to a binary classification problem. First, the target FCM model and the learned FCM model are transformed to binary networks (i.e. the weights are either zero or one). If the absolute weight between two nodes in the FCM model is larger than a pre-defined threshold $\varepsilon_w$, the weight between the corresponding nodes in the binary network will be set to one; otherwise the weight will be set to zero. In order to compare with the other algorithms, the same threshold value and the same definition of positive and negative results as defined by Stach [7] are used. The threshold is set to 0.05. The edges with absolute weights less than 0.05 are identified as positive edges; otherwise, they are identified as negative edges. These definitions are

opposite to the definitions in bioinformatics literature. The sensitivity is defined as

$$Sensitivity = \frac{N_{TP}}{N_{TP} + N_{FN}} \quad (16)$$

where $N_{TP}$ is the number of true positive edges (correctly identified positive edges), $N_{FN}$ is the number of false negative edges (the positive edges that are incorrectly identified as negative edges).

The specificity is defined as

$$Specificity = \frac{N_{TN}}{N_{TN} + N_{FP}} \quad (17)$$

where $N_{TN}$ is the number of true negative edges (correctly identified negative edges), $N_{FP}$ is the number of false positive edges (the negative edges that are incorrectly identified as positive edges).

### *4.3.3 Mean value of specificity and sensitivity*

SS mean is the mean value of specificity and sensitivity. It is defined as

$$SS\ mean = \frac{2 \times Specificity \times Sensitivity}{Specificity + Sensitivity} \quad (18)$$

### *4.3.4 Receiver operating characteristic*

The specificity and sensitivity described above are determined under one threshold, i.e. an edge is classified as zero if the strength of the edge is larger than a predefined threshold; it is classified as one otherwise. They depend on the threshold. The Receiver Operating Characteristic (ROC) curve shows how the specificity and sensitivity changes as the threshold changes. It is a more accurate measure of how well the positive and negative edges can be identified. It is independent of the threshold.

## 4.4 Results

### *4.4.1 Comparison of different algorithms*

For the purpose of comparison, the test data is generated in the same way as Stach used to compare 3 different algorithms [7]. Response sequences are generated randomly from FCM models with different number of nodes and different densities. For every given number of nodes and density, 10 random FCM models are generated as testing models. Several random response sequences are generated from the testing models with different random initial concept activation degrees. In order to compare the results with the other algorithms, appropriate number of sequences and length of the sequences is chosen so that the total volume of data is the same. Stach tested the algorithms on a sequence with 20 data points for each node in the FCM [7]. A sequence of the same size are used in the first set of experiments (denote as ACO-1,20 in Table 2) and 5 response sequences with 4 data points in each sequence are used to perform the second set of experiments (denote as ACO-5,4). We perform the third set of experiments with 40 sequences with 10 data points in each sequence (denote as ACO-40,10) in order to obtain results better than the first two set of experiments. The experiment results are compared with Real-Coded Genetic Algorithm (RCGA) [17], Data-Driven Nonlinear Hebbian Learning (DD-NHL) [2], and Nonlinear Hebbian Learning (NHL) [22]. The results are reported in Table 2.

It may be noticed that the SS mean of ACO-1,20 is better than the three non-ACO algorithms in the experiments with 5 nodes. However, it is worse than RCGA in the experiments with more

**Table 2. Experiment results of different algorithms**

| Algorithm | Data Error | Model Error | Spec | Sens | SS mean | Data Error | Model Error | Spec | Sens | SS mean |
|---|---|---|---|---|---|---|---|---|---|---|
| | Number of Nodes = 5, Density = 20% | | | | | Number of Nodes = 5, Density = 40% | | | | |
| **ACO-1,20** | 0.000±0.000 | 0.362±0.132 | 0.92±0.16 | 0.17±0.11 | 0.27±0.17 | 0.000±0.000 | 0.324±0.108 | 0.93±0.08 | 0.11±0.10 | 0.19±0.15 |
| **ACO-5,4** | 0.000±0.000 | 0.064±0.058 | 0.95±0.09 | 0.68±0.24 | 0.76±0.21 | 0.000±0.000 | 0.075±0.066 | 0.97±0.06 | 0.58±0.32 | 0.67±0.30 |
| **ACO-40,10** | 0.000±0.000 | **0.012**±0.014 | **1.00**±0.00 | **0.93**±0.07 | **0.96**±0.04 | 0.000±0.000 | **0.020**±0.012 | **0.99**±0.03 | **0.88**±0.09 | **0.93**±0.05 |
| RCGA | | 0.321 | 0.93 | 0.12 | 0.20 | | 0.361 | 0.96 | 0.09 | 0.16 |
| DD-NHL | | 0.317 | 0.96 | 0.06 | 0.11 | | 0.381 | 0.96 | 0.07 | 0.12 |
| NHL | | 0.345 | 0.94 | 0.07 | 0.13 | | 0.346 | 0.93 | 0.05 | 0.10 |
| | Number of Nodes = 10, Density = 20% | | | | | Number of Nodes = 10, Density = 40% | | | | |
| **ACO-1,20** | 0.001±0.001 | 0.449±0.041 | 0.92±0.08 | 0.09±0.03 | 0.16±0.05 | 0.001±0.000 | 0.485±0.034 | 0.93±0.04 | 0.09±0.04 | 0.17±0.06 |
| **ACO-5,4** | 0.004±0.001 | 0.319±0.057 | 0.93±0.08 | 0.13±0.06 | 0.23±0.09 | 0.002±0.001 | 0.302±0.040 | 0.92±0.05 | 0.13±0.05 | 0.22±0.08 |
| **ACO-40,10** | 0.003±0.001 | **0.187**±0.040 | 0.93±0.04 | **0.26**±0.06 | **0.40**±0.07 | 0.002±0.001 | **0.170**±0.049 | 0.93±0.05 | **0.26**±0.07 | **0.41**±0.08 |
| RCGA | | 0.398 | **0.96** | 0.11 | 0.19 | | 0.385 | **0.94** | 0.11 | 0.20 |
| DD-NHL | | 0.412 | **0.96** | 0.07 | 0.13 | | 0.423 | 0.93 | 0.05 | 0.10 |
| NHL | | 0.420 | 0.93 | 0.07 | 0.12 | | 0.435 | 0.93 | 0.06 | 0.11 |
| | Number of Nodes = 20, Density = 20% | | | | | Number of Nodes = 20, Density = 40% | | | | |
| **ACO-1,20** | 0.002±0.001 | 0.513±0.026 | 0.93±0.04 | 0.07±0.01 | 0.13±0.02 | 0.001±0.001 | 0.517±0.022 | **0.94**±0.02 | 0.07±0.02 | 0.14±0.03 |
| **ACO-5,4** | 0.011±0.005 | 0.461±0.019 | 0.93±0.03 | 0.09±0.02 | 0.16±0.03 | 0.006±0.002 | 0.473±0.020 | 0.92±0.02 | 0.08±0.02 | 0.14±0.03 |
| **ACO-40,10** | 0.009±0.002 | **0.348**±0.017 | 0.92±0.03 | **0.12**±0.02 | **0.21**±0.04 | 0.006±0.002 | **0.328**±0.031 | 0.92±0.02 | **0.14**±0.03 | **0.24**±0.04 |
| RCGA | | 0.426 | **0.94** | 0.09 | 0.16 | | 0.413 | **0.94** | 0.08 | 0.14 |
| DD-NHL | | 0.464 | 0.93 | 0.08 | 0.14 | | 0.436 | **0.94** | 0.07 | 0.13 |
| NHL | | 0.461 | 0.93 | 0.07 | 0.13 | | 0.468 | 0.93 | 0.05 | 0.10 |
| | Number of Nodes = 40, Density = 20% | | | | | Number of Nodes = 40, Density = 40% | | | | |
| **ACO-1,20** | 0.006±0.006 | 0.525±0.011 | 0.94±0.01 | 0.07±0.01 | 0.13±0.01 | 0.004±0.002 | 0.562±0.010 | 0.94±0.01 | 0.06±0.01 | 0.12±0.01 |
| **ACO-5,4** | 0.020±0.005 | 0.508±0.010 | 0.94±0.01 | 0.07±0.00 | 0.12±0.01 | 0.014±0.005 | 0.538±0.007 | 0.94±0.01 | 0.07±0.01 | 0.13±0.01 |
| **ACO-40,10** | 0.019±0.006 | **0.448**±0.021 | 0.94±0.02 | **0.08**±0.01 | **0.15**±0.01 | 0.019±0.004 | 0.437±0.012 | 0.93±0.01 | **0.09**±0.01 | **0.16**±0.01 |
| RCGA | | 0.453 | 0.94 | **0.08** | **0.15** | | **0.436** | **0.96** | 0.08 | 0.15 |
| DD-NHL | | 0.468 | **0.96** | 0.07 | 0.12 | | 0.465 | 0.93 | 0.06 | 0.12 |
| NHL | | 0.489 | 0.95 | **0.08** | **0.15** | | 0.498 | 0.94 | 0.06 | 0.12 |

Note: Spec means specificity. Sens means sensitivity. Data errors for RCGA, DD-NHL and NHL are not available.

than 5 nodes and it is worse than DD-NHL and NHL in the experiments with more than 10 nodes. Although the model errors and SS means are not satisfactory, the data error (i.e. the objective function value) is low (approximately $10^{-5}$). It is possible that the low performance is because that only one response sequence does not provide enough information. So another two sets of experiments are performed (i.e. ACO-5,4, and ACO-40,10).

In ACO-5,4, the size of the input data sequence is the same as ACO-1,20, but the data contains more response sequences. Each of the response sequence provides an independent observation of the FCM model. The results suggest that ACO-5,4 is significantly improved over ACO-1,20 with the confidence level of 95%. ACO-5,4 is better than the three non-ACO algorithms in terms of both model error and SS mean for the problems with 5 nodes and 10 nodes. However ACO-5,4 is worse than RCGA for problems with 20 and 40 nodes.

The number of response sequences is further increased in order to obtain better results. The ACO-40,10 experiments use 40 response sequences with 10 data points in each sequence. The model errors, sensitivities and SS means are all better than the other three algorithms, except the case with 40 nodes and 40% of density (the model error is 0.001 higher than RCGA). Although specificity is slightly lower, SS mean should be considered as a more accurate measure of the performance because the balance between specificity and sensitivity can be adjusted as shown in Figure 3.

In many of the experiments, the sensitivity value is between 0.1 and 0.2. The low sensitivity value is partly due to the way of classifying the edges. $\varepsilon_w$=0.05 is used as the threshold for the weights to determine the class label of the edges. In practical problems, a larger threshold should be chosen. The ROC curve is usually used to show how the true positive rate (specificity) and false positive rate (1−sensitivity) changes as the threshold changes. The ROC curve for three randomly chosen experiments from Table 2 is shown in Figure 3. The curves are all above the reference line. The Area Under ROC Curve (AUC) is 0.98, 0.85, 0.72 and 0.65. These values are relatively high and it is possible to find a good balance between sensitivity and specificity.

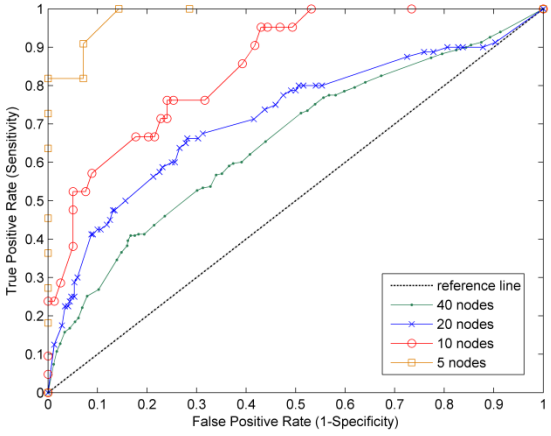


**Figure 3. ROC curves of three randomly chosen experiments**

From the experiment results we may conclude that ACO can learn FCM models more accurate than the other three non-ACO algorithms in some of the problems. However the ACO's performance drops as the number of nodes increases. The accuracy of the learned FCM models can be improved by providing more data to the ACO algorithm. As both the number of response sequences and the number of nodes affect the accuracy of the learned FCM models, more experiments are performed to show the effect of these two factors.

### *4.4.2 Effect of number of response sequences*

In practical problems, multiple response sequences may be observed. The results in Table 2 suggest that the model error and SS mean improves as number of response sequences increases. More response sequences should be provided to the algorithm if they are available. Figure 4 and Figure 5 shows the box-plots of model errors and SS means for different number of response sequences respectively. The results are based on the experiments on FCMs with 20 nodes and a density of 40%. Every response sequence contains 10 data points. 10 experiments are performed for every number of response sequences. As shown in Figure 4, the maximum, minimum and average model errors decreases as the number of response sequences increases, with only three exceptions: the average model error for $N_s$=25 is slightly higher (than the average model error for $N_s$=20); the average model error for $N_s$=40 is slightly higher; and the minimum model error for $N_s$=40 is higher. These exceptions should be caused by random effect. The SS mean values shown in Figure 5 have a similar trend. The maximum SS mean increases as the number of response sequences increases, except for the case of $N_s$=40. However the trend for average values and minimum values of SS mean is not as clear as the trend for model errors. It should be clearer if more experiments are performed.

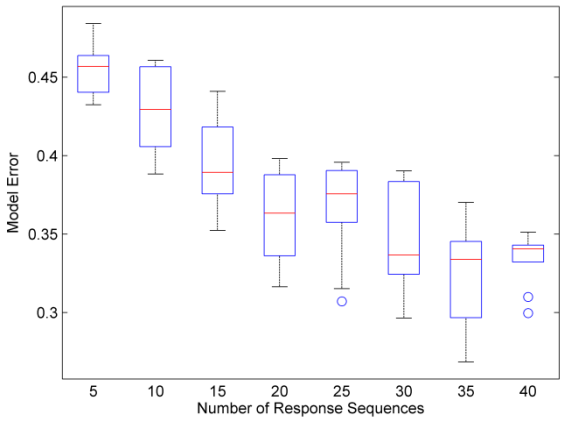


**Figure 4. Effect of number of sequences on model errors**

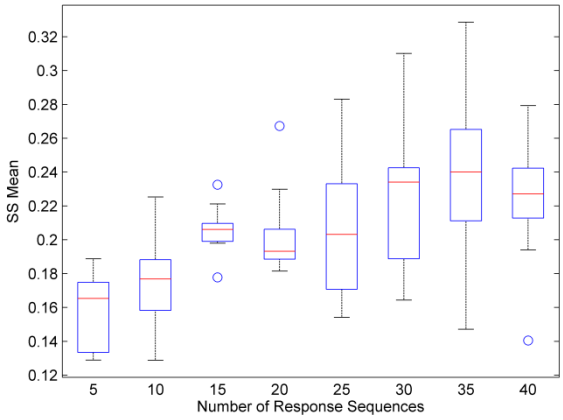


**Figure 5. Effect of number of response sequences on SS means**

### *4.4.3 Effect of number of FCM nodes*

As the number of FCM nodes increases, the number of weights increases quadratically and the possible number of solutions increases exponentially. The performance of the algorithm drops because there are more solutions to be explored and the algorithm may be easier to be trapped in the increasing number of local optimums. Figure 6 shows how model error and

SS mean changes as the number of FCM nodes changes. The average, minimum and maximum values of both performance measures drops as the number of nodes increases.

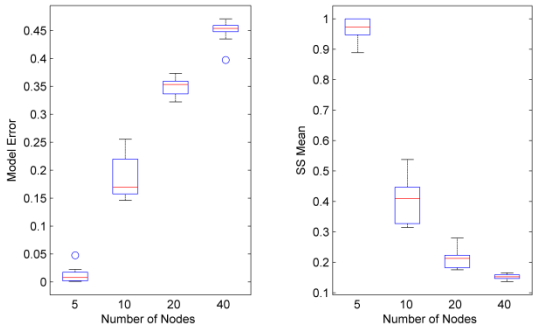


**Figure 6. The effect of number of FCM nodes**

# 5. CONCLUSION

Building causal models from data is an important problem with many applications. Fuzzy Cognitive Maps (FCMs) are a particular causal model. The aim of this paper is to present an Ant Colony Optimization (ACO) based algorithm to develop a FCM causal model from data.

A prior study proposed learning FCMs by an ACO. The algorithm was shown to be able to learn 8 weights of a FCM. A new ACO approach that is able to learn FCMs with at least 1600 weights (40 nodes) from several response sequences is proposed.

Several algorithms, such as Real-Coded Genetic Algorithm (RCGA), Nonlinear Hebbian Learning (NHL) and Data-Driven NHL (DD-NHL), were proposed to learn FCMs directly from data and applied to the same problem as this paper (i.e. recovering causal model from data). The ACO approach was compared to these algorithms through experiments. The proposed ACO algorithm outperforms RCGA, NHL and DD-NHL in terms of model error and SS mean measures when multiple response sequences are used in the learning process. However the ACO algorithm outperforms the three algorithms in only limited cases when only one response sequence is used in the learning process.

Because the basic ACO algorithm is used without any further improvements such as local search procedures, the performance of the ACO algorithm can be improved in the future studies.

# 6. REFERENCES


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

[2] Stach, W., Kurgan, L. and Pedrycz, W. Data-driven nonlinear Hebbian learning method for fuzzy cognitive maps. In *Proceedings of the 2008 IEEE International Conference on Fuzzy Systems, FUZZ 2008,* (Hong Kong, China, June 1 - June 6, 2008), 1975-1981.

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

[4] 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(2005), 95-121.

[5] 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 (2007), 56-68.

[6] Stach, W., Kurgan, L. and Pedrycz, W. A divide and conquer method for learning large fuzzy cognitive maps. *Fuzzy Sets and Systems*, 161, 19 (2010), 2515-2532.

[7] Stach, W. *Learning and Aggregation of Fuzzy Cognitive Maps - An Evolutionary Approach*. PhD Dissertation, University of Alberta, 2010.

[8] Chandra Mohan, B. and Baskaran, R. A survey: Ant Colony Optimization based recent research and implementation on several engineering domain. *Expert Systems with Applications*, 39, 4 (2012), 4618-4627.

[9] Dorigo, M., Maniezzo, V. and Colorni, A. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26, 1 (1996), 29-41.

[10] Liao, T., Oca, M. A. M. d., Aydin, D., Stützle, T. and Dorigo, M. An incremental ant colony algorithm with local search for continuous optimization. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation* (Dublin, Ireland, 2011), 125-132.

[11] Xiao-Min, H., Jun, Z., Chung, H. S. H., Yun, L. and Ou, L. SamACO: Variable Sampling Ant Colony Optimization Algorithm for Continuous Optimization. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 40, 6 (2010), 1555-1566.

[12] Ding, Z., Li, D. and Jia, J. First Study of Fuzzy Cognitive Map Learning Using Ants Colony Optimization. *Journal of Computational Information Systems*, 7, 13 (2011), 4756 - 4763.

[13] Chen, N. and Zhang, J. Index-based genetic algorithm for continuous optimization problems. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation* (Dublin, Ireland, 2011), 1029-1036.

[14] Tsadiras, A. K. Comparing the inference capabilities of binary, trivalent and sigmoid fuzzy cognitive maps. *Information Sciences*, 178, 20 (2008), 3880-3894.

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

[16] Koulouriotis, D. E., Diakoulakis, I. E. and Emiris, D. M. Learning fuzzy cognitive maps using evolution strategies: A novel schema for modeling and simulating high-level behavior. In *Proceedings of the Congress on Evolutionary Computation* (Soul, Korea, Republic of, 2001), 364-371.

[17] Stach, W., Kurgan, L., Pedrycz, W. and Reformat, M. Evolutionary development of fuzzy cognitive maps. In *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2005* (Reno, NV, United states, May 22 - May 25, 2005), 619-624.

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

[19] Alizadeh, S., Ghazanfari, M., Jafari, M. and Hooshmand, S. Learning FCM by Tabu Search. *International Journal of Computer Science*, 2, 2 (2007), 142–149.

[20] Alizadeh, S. and Ghazanfari, M. Learning FCM by chaotic simulated annealing. *Chaos, Solitons & Fractals*, 41, 3 (2009), 1182-1190.

[21] Christian, B. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2, 4 (2005), 353-373.

[22] Papageorgiou, E. I., Stylios, C. D. and Groumpos, P. P. Fuzzy cognitive map learning based on nonlinear Hebbian rule. In *Proceedings of the Australian Conference on Artificial Intelligence* (2003), 256-268.