Fuzzy Optimization and Decision Making

Fuzzy cognitive maps for autonomous agents in dynamic environments

--Manuscript Draft--

Manuscript Number:	FODM-D-18-00369
Full Title:	Fuzzy cognitive maps for autonomous agents in dynamic environments
Article Type:	Original Research
Keywords:	fuzzy cognitive maps; autonomous agents; decision-making; multi-agent model; artificial intelligence
Corresponding Author:	Tomas Nachazel Univerzita Hradec Kralove Hradec Kralove, CZECH REPUBLIC
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Univerzita Hradec Kralove
Corresponding Author's Secondary Institution:	
First Author:	Tomas Nachazel
First Author Secondary Information:	
Order of Authors:	Tomas Nachazel
Order of Authors Secondary Information:	
Funding Information:	Faculty of Informatics and Management, University of Hradec Králové — Mr. Tomas Nachazel
Abstract:	This paper describes a new modification of fuzzy cognitive maps (FCMs) for the modeling of autonomous entities that make decisions in a dynamic environment. The paper offers a general design of an FCM for autonomous agents (AA-FCM). Unlike an original FCM, the AA-FCM modification offers a more complex structure and processing with an emphasis on the demands of autonomous systems. The AA-FCM design distributes concepts into three different classes according to their purpose in the map: Needs, Activities, and States. This classification enables decision-making, easy processing of inputs from sensors, faster reactions of the system, more realistic behavior in simulations, the modeling of inner needs, the granularity of a simulation, and self-evaluation of the AA-FCM that supports unsupervised evolutionary learning. To demonstrate its strengths, the proposed extension was implemented into the agent-based artificial life model, where it takes advantage of the all above mentioned features in competition for resources, natural selection and evolution.

Fuzzy cognitive maps for autonomous agents in dynamic environments

Tomas Nachazel
Faculty of Informatics and Management, University of Hradec Králové, Rokitanskeho 62,500 03, Hradec Králové, Czech Republic
tomas.nachazel@uhk.cz, orcid ID: 0000-0001-6484-3793

Abstract. This paper describes a new modification of fuzzy cognitive maps (FCMs) for the modeling of autonomous entities that make decisions in a dynamic environment. The paper offers a general design of an FCM for autonomous agents (AA-FCM). Unlike an original FCM, the AA-FCM modification offers a more complex structure and processing with an emphasis on the demands of autonomous systems. The AA-FCM design distributes concepts into three different classes according to their purpose in the map: Needs, Activities, and States. This classification enables decision-making, easy processing of inputs from sensors, faster reactions of the system, more realistic behavior in simulations, the modeling of inner needs, the granularity of a simulation, and self-evaluation of the AA-FCM that supports unsupervised evolutionary learning. To demonstrate its strengths, the proposed extension was implemented into the agent-based artificial life model, where it takes advantage of the all above mentioned features in competition for resources, natural selection and evolution.

Keywords: fuzzy cognitive maps, autonomous agents, decision-making, multi-agent model, artificial intelligence

1. Introduction

Fuzzy cognitive maps (FCMs) are powerful tools for the simulation of dynamic phenomena. They are generally used to predict or simulate systems that consist of many dependent variables in a complex dynamic structure (Kosko 1986). However, FCM has proved to be a strong artificial intelligence method even for autonomous agents. An agent is an intelligent computer system that is capable of evaluating a situation, making decisions and performing actions (Mendonça et al. 2014). It is usually an entity with a location and an ability to move within the environment, but the proposed approach is also applicable to static ambient intelligence, in which the observed environment is inside the entity.

The motivation of this paper is to propose a new modification of the FCM for autonomous agents (AA-FCM). The original design of FCMs does not allow for differences between concepts; each concept node has the same range of values, the same behavior, and the same interpretation. This approach is sufficient in a dynamic system with equal elements of the same type. However, if an FCM has different categories of concept nodes, some changes are unavoidable. The management of different kinds of nodes after computation often becomes very confusing and context-dependent, hindering the modularity and scalability of the FCM. This paper offers a new design of FCMs that allows for various types of nodes, and processes them according to an implemented classification during computation, meaning that special treatment after computation is unnecessary.

This method has several advantages that are impossible for a standard approach to achieve without heavy editing of the algorithm. It enables decision-making, easy processing of inputs from sensors, faster reactions of the system, more realistic behavior in simulations (disabling parallel activities if necessary), a simulation of inner needs, adjustments to the granularity of the simulation, and self-evaluation of the agent (fitness), which supports learning.

The AA-FCM approach is quite different from the classical FCM in terms of its structure and use. It is an artificial intelligence method that lies between the approaches of simple rule sets and complex neural networks. It is a compromise that combines advantages from both sides, allowing for complex behavior and learning similar to that of a neural network, while retaining human readability and a straightforward initial design.

To verify the proposed design and demonstrate its abilities, this paper describes the implementation of the design within an artificial life model. The main reason for this choice is its ability to test the quality of artificial intelligence, as thousands of test subjects with various attributes and behaviors compete in a single model. Artificial life modeling allows us to explore real natural phenomena, emergence and evolution within a runtime of only minutes. Artificial individuals behave according to their needs and the situation in their vicinity, and their intelligence is tested through competition for limited resources. To survive, they also need to deal with changes in a dynamic environment. Natural selection guides the main directions of evolution and the specialization of various species after several generations.

The concept proposed in this paper is implemented in ambient intelligence model to simulate human behavior for testing purposes, and has already been modified for large-scale models in which performance is a key factor. In the latter case, an analytic hierarchy process (AHP) replaced the decision-making part of AA-FCM (Nachazel 2018). The combination with AHP improved performance, but the quality of decision-making and evolutionary progress were slightly decreased. Although this modification has been already published, a full description of AA-FCM has not previously been published, and this is the objective of the current paper.

The paper is organized as follows: Section 2, Related Work, provides an overview of the various usages of FCM, and its extensions and modifications that relate to agents or systems. Section 3 introduces the essential elements of the original FCMs. Section 4 continues with a description of the new FCM extension for autonomous agents (AA-FCM). This section covers the fundamental aspects of this modification, and is followed by Section 5, Additional features, in which some more advanced features and additions are described. Section 6 presents the complete, step-by-step design of the AA-FCM for an artificial life model and discusses the testing of the created AA-FCM in the model. Finally, Section 7 gives concluding remarks.

2. Related Work

Fuzzy cognitive maps (FCMs) offer a powerful approach to modeling various systems that consist of many dependent variables within a complex structure. Usually, an FCM is designed for a single, narrowly focused task, such as a support tool for analysis (Hsieh et al. 2013; Senniappan et al. 2016), decision-making (Ahmadi et al. 2015; Kyriakarakos et al. 2014), predictions (Jayashree et al. 2015), and various tasks in social sciences (Giabbanelli and Crutzen 2014; Barón et al. 2014). These FCMs aim to generate specific values or a steady state after a few iterations. However, this paper regards FCMs as an artificial intelligence method for systems in dynamic environments. Autonomous systems need to handle a much wider range of situations, and require a different approach to FCMs. FCMs with a few enhancements have been shown to be strong artificial intelligence methods, even for autonomous agents (Mendonça et al. 2014).

In the field of intelligent systems, some projects aimed at interaction with humans use an FCM to simulate emotions. For educational purposes, a virtual pet was designed based on an FCM that ensures believable reactions to the user’s actions (Laureano-Cruces and Rodriguez-Garcia 2011). An ambient intelligence system also enriched its user interface with FCM-based emotions to provide additional comfort and to respond naturally to the presence of users (Acampora et al. 2011).

In many projects, FCMs are a core component of a system or agent in a model. For instance, this approach has been used by monitoring systems in which an FCM assesses risk in critical situations (Szwed et al. 2014), a situation awareness model for infantry platoon leaders (Jones et al. 2011), or even landing site selection for planetary exploration (Furfaro et al. 2012).

Several articles have addressed the possibilities of learning FCMs. Systems in a dynamic environment need to deal with continuous changes, and thus unsupervised training of the FCM or even its adjustment during runtime are desirable features in such projects. Genetic algorithms and Hebbian algorithms are commonly used as bases for various extensions to deal with this issue (Motlagh et al. 2015; Stach et al. 2010).

A dynamic fuzzy cognitive map (DFCM) is one possible extension focusing on the learning of FCMs. This is based on the random neural model, which was designed to react to random events by modifying causal relations. The main feature of this approach is its ability to change an FCM during runtime enabling adaptation and switching different behaviors at runtime. Although its usage is suitable for autonomous agents and systems, it does not necessarily overlap with the modification proposed in this paper, since the DFCM approach focuses on dynamic changes in relations based on additional fuzzy rules. This extension has been used as a supervision system (Jose 2010) and a navigation system for a robot (Mendonça et al. 2015; Mendonça et al. 2014).

FCMs also provide useful services in multi-agent systems. There are two areas that are suitable for this method: a supervisor monitoring and controlling an environment (Stula et al. 2011; Jose 2010), or an artificial intelligence controlling individual agents. The latter is not widely used, since FCMs in their original form are not convenient for this purpose, and modifications that could support this role are not yet available. However, many of the works mentioned above at least slightly touched on the problem of FCMs for autonomous systems. Moreover, some models already process agents with an FCM, for instance the Artificial Life Model (ALModel), the design of which forms part of this paper, and the EcoSim model (Gras et al. 2011).

EcoSim, an artificial life model, uses an FCM to process the behavior of individuals, and is similar to the ALModel. The simulation also allows for the evolution of values in the FCM, which enables the adaptation of behavior. Individuals can choose from a limited set of basic actions, and select the optimal one for the current situation. The development and behavior of the population emerge from interactions between agents. EcoSim contains two types of species, predator and prey, which both evolve to increase their chances of success against the rival species. The decision-making of both the prey and predator species differs in terms of its actions and observed properties. High-level and low-level control are combined in a single map, which generates a very complex FCM (26 concepts). Despite this complexity, the behavior is focused only on reproduction and the management of food and energy. The patterns generated in this way are visually close to those of cell-based models. EcoSim aims to observe the emergence and evolution of the population, rather than to realistically simulate the behavior of individuals (Khater and Gras 2012).

3. Fuzzy Cognitive Maps

FCMs are dynamic systems of concept nodes with a complex network of relations. The system is changing through iterations (time steps) according to a set of relations. The original FCMs are based on a collection of concept nodes *C* and the relations between these nodes. The adjacency matrix *R* in Eq. (1) is a general matrix representation of an FCM as the matrix of relations between *c* concept nodes (Kosko 1986).

$$R = \begin{pmatrix} w_{11} & \cdots & w_{1c} \\ \vdots & \ddots & \vdots \\ w_{c1} & \cdots & w_{cc} \end{pmatrix} \qquad (1)$$

Each value w_{ij} represents the influence of node c_j on node c_i. This may be any real number between −1 (strong negative causality) and 1 (strong positive causality). If $w_{ij} = 0$, then node c_j has no direct influence on node c_i. If $w_{ij} > 0$, then a high value of node n_j raises the level of node c_i; if $w_{ij} < 0$, then a high level of node c_j lowers the value of node c_i.

In addition to the matrix *R*, an FCM needs the truth values of nodes, which change at every time step. These values form a vector C^t (a one-dimensional array defined in Eq. (2)).

$$C^t = \begin{pmatrix} c_1^t \\ c_2^t \\ \vdots \\ c_k^t \end{pmatrix} \qquad (2)$$

where c_i^t is the value of node c_i at time step *t*. As a fuzzy truth value, c_i^t is always a real number ranging between 0 (definitely not true) and 1 (definitely true). If the value exceeds these limits after computation, then it is immediately reduced to within the valid range. The sizes of all components are constant. The vector C^t is updated at every time step, and the content of *R* is static. Eq. (3) shows the computation of one iteration *t* based on the original definition model, while Eq. (4) represents a version in which the product of multiplication is added to the previous value, known as the incremental model (Motlagh et al. 2015).

$$C^t = f\left(R \cdot C^{(t-1)}\right) \qquad (3)$$

$$C^t = f\left(C^{(t-1)} + R \cdot C^{(t-1)}\right) \qquad (4)$$

In both equations, the activation function *f* represents a transformation of values. There are many diverse types of nonlinear functions that can be used (e.g. sigmoid, hyperbolic, step and other functions). The primary task for this function is to keep the values within the fuzzy range. The model described in this paper mostly uses a simple linear transformation unless a value exceeds the range (see Eqs. (5, 6, 7)).

$$\forall x \in [0, 1]: f(x) = x \qquad (5)$$

$$\forall x > 1: f(x) = 1 \qquad (6)$$

$$\forall x < 0: f(x) = 0 \qquad (7)$$

The original FCMs handle all concept nodes in the same way, which causes a few issues in systems which contain elements of different type. For example, if FCM considers a few inputs and then decides whether to trigger an action, then a trigger node has to be processed differently than the other nodes. Besides choosing a sufficient level of the node to trigger an action, it usually has to recognize only two states of node: an action is performed or not. Such differences in the processing and interpretation of nodes make FCM confusing and less modular. This is the reason to use a modification which is built to handle such situations.

4. Fuzzy Cognitive Maps for Autonomous Agents

This section presents a new approach to design called the AA-FCM (fuzzy cognitive maps for autonomous agents), which makes the creation of FCMs for more complex systems with various types of concept nodes more comprehensive, effective and extensible. This new method of processing of nodes and optimization in the computation offers several advantages that would be impossible to achieve with the general classical approach. The proposed extension introduces three main categories of concept nodes: Needs, Activities, and States.

4.1. Structure

Let *N* denote the set of nodes from *C* which were identified as a *Needs*; similarly, *A* represents set of *Activity* nodes, and *S* is the set of *State* nodes. In the design phase, this approach requires the classification of all concepts into these classes (see Eqs. (8, 9)). Each node is in exactly one class, meaning that classes are disjoint sets (see Eq. 10).

$$C = (N \cup A \cup S) \tag{8}$$

$$\forall c_i \in (N \cup A \cup S) \tag{9}$$

$$(N \cap A) = \emptyset; (N \cap S) = \emptyset; (A \cap S) = \emptyset \tag{10}$$

For a better explanation of the algorithms, the original general variables are replaced to represent the classification. Using the classification above, set C^t is decomposed into three sets N^t, A^t, and S^t (see Eq. 11). Along with the sets of concept nodes, the variables of particular nodes and their values are changed from the general variable c_i to n_i, a_i, or s_i following the same pattern: *n* for *Needs*, *a* for *Activities*, and *s* for *States* (see Eq. 12). The total number of concept nodes is *c*, *n* represents the number of *Need* nodes, *a* the number of *Activity* nodes, and *s* the number of *State* nodes (see Eq. 13).

$$C^t = N^t \cup A^t \cup S^t \tag{11}$$

$$\begin{pmatrix} c_1^t \\ c_2^t \\ c_3^t \\ c_4^t \\ c_5^t \end{pmatrix} \rightarrow \begin{pmatrix} n_1^t \\ n_2^t \\ a_3^t \\ s_4^t \\ s_5^t \end{pmatrix} \tag{12}$$

$$c = n + a + s \tag{13}$$

4.2. The Needs Concept Class

Needs are the first class of concept nodes. A designed system usually has at least one purpose that it is trying to fulfill, and thus a measure of its success in this effort may be useful. Alternatively, it may be designed to observe a variable and to perform an action repeatedly, to keep the variable under control. If this action is costly in terms of resources or time, it is not convenient to take action too often, and the system may therefore use a time delay or sensitive balancing. In such cases, *Need* nodes are the optimal choice for representing these concepts. For example, this class can be used for the needs of individuals in an artificial life model or a simulated level of satisfaction with memory management in a system.

The main difference from the other nodes is their behavior during computation, since their computation is based on the incremental model (see Eq. 4). If there are no active influences on a node from others, then it holds its value, behaving exactly like a concept c_i in a definition model that has the relation to itself $w_{ii} = 1$. However, a *Need* node is also capable of other features: adjusting the granularity of the simulation, the addition of true positive causality from a node n_i to itself, self-evaluation, and considering the importance of activities. Eq. (14) shows the computation of a value n_i^t. This equation uses the basic transformation *f* (see Eqs. (5, 6, 7)) and can be extended using the granularity parameter *g* which will be described later (see *Section 5.1. Granularity*).

$$n_i^t = f\left\{ n_i^{t-1} + \left[\left(\sum_{j=1}^{n} w_{ij} \cdot n_j^{t-1} \right) + \left(\sum_{j=(\mathrm{n}+1)}^{(n+a)} w_{ij} \cdot a_j^{t-1} \right) + \left(\sum_{j=(\mathrm{n}+\mathrm{a}+1)}^{(n+a+s)} w_{ij} \cdot s_j^{t-1} \right) \right] \right\} \tag{14}$$

The value of a *Need* node represents the level of need to do something. Hence, zero represents the ideal level at which an agent does not have to do anything to satisfy this need. If the level approaches one, then the agent has failed to satisfy the need, and should take an appropriate action as soon as possible. The exact critical level of a *Need* node that initiates a corresponding activity obviously depends on the settings of the relations in matrix *R*.

4.3. The Activities Concept Class

The next class of concept nodes, *Activities*, represents all possible actions that a system can perform. If an agent with an AA-FCM is not simply a passive observer and is required to react, manage, or affect its environment or itself in any way, there are two possible solutions. The first approach involves another mechanism outside of an FCM which reads values from the FCM and makes a decision (Nachazel 2018); the second one locates the actions directly inside the AA-FCM, which then holds the decision-making responsibility.

As actions, these concepts are either active or are not; therefore, after computation, the *Activity* nodes have only two possible values: zero (*false*; the action is inactive) or one (*true*; an agent is performing the action) (see Eq. 15). However, during their computation, these values have the full fuzzy range of between zero and one. They are also calculated using an algorithm similar to that used for general nodes in the original FCMs (based on the definition model; see Eq. 16).

$$\forall\, a_i^t \in \{0; 1\} \qquad (15)$$

$$a_i^t = f_a\left\{\left(\sum_{j=1}^{n} w_{ij}\cdot n_j^{t-1}\right)+\left(\sum_{j=(\mathrm{n}+1)}^{(n+a)} w_{ij}\cdot a_j^{t-1}\right)+\left(\sum_{j=(\mathrm{n}+\mathrm{a}+1)}^{(n+a+s)} w_{ij}\cdot s_j^{t-1}\right)\right\} \qquad (16)$$

The values are rounded using a simple algorithm (the transformation f_a), which decides on activities based on the fuzzy values acquired from the computation. Depending on whether parallel activities are available, it selects only the activity with the highest value (see *Section 5.3. Disabling parallel activities*) or performs all activities reaching a certain critical level a_c (the default is 0.5; see Eqs. (17, 18)).

$$\forall x \geq a_c\!: f_a(x) = 1 \qquad (17)$$

$$\forall x < a_c\!: f_a(x) = 0 \qquad (18)$$

In cases where the purpose of the *Activity* nodes is to provide a fuzzy value to express the intensity of an action, the basic *f* transformation (Eqs. (5, 6, 7)) can be used. Transformation functions can be adjusted to the demands of the system without causing any problems with the rest of the design.

In general, FCMs are not suitable for a combination of high-level decision-making (what should be done) and low-level computation (how it should be done); this combination often requires too many variables in one structure, which causes performance issues. For example, in an artificial life model, the AA-FCM decides that an individual should search for food. This decision is then forwarded to actuators, which check the target location in the environment and find a path. This pathfinding would be extremely ineffective if managed by the AA-FCM along with the other concept nodes, but could be solved by another FCM dedicated to pathfinding. However, in any case, low-level operations should be separated from high-level decision-making in order to ensure the clarity of the model, and the most importantly, adequate performance.

4.4. The States Concept Class

The third class of concept nodes, *States*, is very similar to the general concept in the original FCMs. The way in which it is calculated is virtually the same (see Eq. (19)), and its purpose does not have the narrow focus of the classes presented above.

$$s_i^t = f\left\{\left(\sum_{j=1}^{n} w_{ij}\cdot n_j^{t-1}\right)+\left(\sum_{j=(\mathrm{n}+1)}^{(n+a)} w_{ij}\cdot a_j^{t-1}\right)+\left(\sum_{j=(\mathrm{n}+\mathrm{a}+1)}^{(n+a+s)} w_{ij}\cdot s_j^{t-1}\right)\right\} \qquad (19)$$

Besides general concepts, the *State* nodes are the optimal choice for external inputs. If an agent needs to be able to perceive an attribute of the environment and take it into account during the decision-making process, it requires a dedicated *State* node in its AA-FCM. As a property of the environment, the value of this attribute cannot interact with any node in the AA-FCM directly; instead, it is updated by sensors. The inserted values obviously have to be transformed to the fuzzy range, and all relations to this node in matrix R are therefore equal to zero, allowing its entire calculation to be omitted.

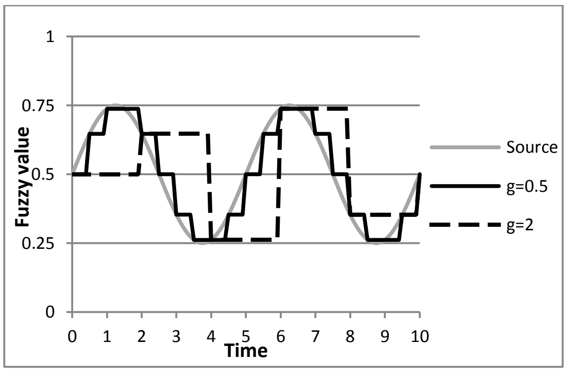


Fig. 1 Comparison of different settings of granularity

5. Additional Features

While the previous section described the core of the proposed method, this section provides optional attachments that are very useful for certain systems but unnecessary for others. Although a few of these are dependent on each other (the dependencies are noted), the core design presented above can be implemented without the following features.

5.1. Granularity

Since agents process an FCM through iteration in discrete time steps, the selection of the period between computations of the FCM is a crucial issue regardless of whether the system deals with continuous real-time or discrete time steps. The values of an FCM in a dynamic environment are always valid only within the particular interval for which the FCM was designed. In general, a shorter interval generates better reactions of agents, but higher performance demands. This is very similar to sampling in signal processing; when samples are too distant, much of information between them can be lost.

Figure 1 shows how an AA-FCM perceives a continuous variable with different settings of the granularity in continuous time. The AA-FCM with a granularity of 0.5 is four times more demanding in terms of performance, but is also more precise than one with a granularity of 2.0. The parameter *g* is a multiplier representing the length of an interval between computations, which is relative to the basic length of the interval for which a model was originally designed.

Note that in a single system, the granularity parameter of agents may vary, meaning that they can have different reaction times. In a model with continuous time, the designer chooses how often an FCM updates its values; in a discrete model, the problem instead lies in deciding how much virtual time (or how many time steps) elapses between the computations or how much the environment changes in a single time step. After an FCM is designed, any change to the length of this interval invalidates certain values related to dynamic phenomena in the environment. However, with the proposed solution, the influence of granularity on concept classes is evident; *Need* nodes are obviously affected; *Activity* nodes are not, since a decision could be made at any time; and *State* nodes typically are not, but since they have a broader use, some may be affected in certain cases.

The effect of granularity on *Need* nodes causes differences in their growth (or decrease). This can be easily compensated for with a simple enhancement of the calculation, and the nodes always adjust to the current simulation speed. Eq. (20) shows the addition of the granularity parameter *g*, which adjusts the size of the difference of the value per step.

$$n_i^t = f\left\{n_i^{t-1} + \boldsymbol{g}\cdot\left[\left(\sum_{j=1}^{n} w_{ij}\cdot n_j^{t-1}\right)+\left(\sum_{j=(\mathrm{n}+1)}^{(n+a)} w_{ij}\cdot a_j^{t-1}\right)+\left(\sum_{j=(\mathrm{n+a}+1)}^{(n+a+s)} w_{ij}\cdot s_j^{t-1}\right)\right]\right\} \tag{20}$$

The values of the *Need* nodes reach $(1/g)$ times more computations per virtual time unit than they would have with g = 1. For example, in an artificial life model, an individual escaping from a predator has a depleted stamina after three time steps with the default granularity; with the granularity at 0.5, the individual manages to run over six computations. Of course, in both cases, it runs the same length and for the same amount of virtual time, but due to the granularity, it has twice as many opportunities to reconsider the activity or the direction of its escape.

5.2. Faster reactions of agents

As can be seen in Eq. (3), the original FCMs compute the current values by using the values of the previous iteration. This procedure inevitably causes a delay between a stimulus and the corresponding reaction (i.e. one time step or the interval between recomputations of the FCM). Depending on how often the FCMs are recomputed, this delay may cause problems if a short reaction time is essential for proper operation of a system. For example, a monitoring system should react to a fire immediately, as soon as sensors detect it, rather than waiting for the next iteration to take action.

Using these different classes of concept nodes, the computation of an FCM can be divided into three parts, which can then be rearranged in any order. Moreover, some parts can even consider values of the current iteration from parts that have already been computed. The best order has shown to be as follows: first the *Need* nodes, then the *State* nodes and finally the *Activity* nodes. *Need* nodes do not have to correspond with the most current values; since they use an incremental model (see Eqs. (4, 14, 20)), their values are primarily based on their own previous values and actions performed in the previous time step. *State* nodes may be based on the current values of needs, but also contain external inputs that have to be considered in the decision-making as soon as possible. Finally, *Activity* nodes, as the decision-making part of the model, should access the latest values in order to give the best possible reaction to the current situation. The implementation of this feature requires only the replacement of c_j^{t-1} with c_j^t for concept classes that have been already computed. Eqs. (21, 22) show adjusted expressions for the order recommended above. The equation for *Need* nodes is not affected, since no other current values are yet available for time *t*.

$$s_i^t = f\left\{\left(\sum_{j=1}^{n} w_{ij} \cdot n_j^t\right) + \left(\sum_{j=(n+1)}^{(n+a)} w_{ij} \cdot a_j^{t-1}\right) + \left(\sum_{j=(n+a+1)}^{(n+a+s)} w_{ij} \cdot s_j^{t-1}\right)\right\} \tag{21}$$

$$a_i^t = f_a\left\{\left(\sum_{j=1}^{n} w_{ij} \cdot n_j^t\right) + \left(\sum_{j=(n+1)}^{(n+a)} w_{ij} \cdot a_j^{t-1}\right) + \left(\sum_{j=(n+a+1)}^{(n+a+s)} w_{ij} \cdot s_j^{t}\right)\right\} \tag{22}$$

For example, in an artificial life model, individuals have two concept nodes: the *Danger* state and the *Escape* activity. When an individual recognizes a dangerous situation, it should immediately escape rather than wait until the next time step to take the action. Tables 1 and 2 show both approaches; obviously, an individual with the AA-FCM with this feature has a much better chance of escaping and surviving.

Table 1
The delay in reaction time in the original FCM

Iteration	1	2	3	4	5	6	7	8
Danger	0	1	0	0	1	1	1	0
Escape	0	0	1	0	0	1	1	1

Table 2
Immediate reactions in the AA-FCM

Iteration	1	2	3	4	5	6	7	8
Danger	0	1	0	0	1	1	1	0
Escape	0	1	0	0	1	1	1	0

5.3. Disabling parallel activities

In many systems, the individual actions are independent of each other; however, there are cases where an agent is limited to one action per time step, since different activities require different positions of the agent or several of them utilize a single actuator. The original FCM method cannot restrict this without another algorithm that processes and adjusts the values of specific nodes. If any process or value is bound to specific nodes by its position in an FCM, then the algorithm has to be adjusted after any change. This is confusing and a less modular approach.

The AA-FCM uses a simple adjusted transformation algorithm for its *Activity* nodes, meaning that it is not bound to specific nodes, which allows simple changes in concepts without disrupting the algorithm. This feature is useful in artificial life models, and also more generally in simulations of living creatures. For example, as simulated individuals, these agents should not be able to search for food, drink and sleep at the same time step.

The following equations (Eqs. (23-26)) describe the adjusted transformation function f_a for non-parallel activities.

$$a_i^t = f_a(x_i^t) \qquad (23)$$

$$x_i^t = \max(X^t) \wedge x_i^t > a_c \qquad (24)$$

$$a_i^t : f_a(x_i^t) = 1 \qquad (25)$$

$$\forall a_j^t : f_a(x_j^t) = 0; i \neq j \qquad (26)$$

where x_i^t is the value of the *Activity* node *i* at time step *t* before the transformation f_a. Vector X^t then includes all values x_i^t. When disabling of parallel activities is required, the algorithm finds the *Activity* node with the highest value after all nodes are calculated: if the value exceeds a critical level a_c, then the activity is performed; otherwise, the agent does nothing (or returns to its default state). Within this single cycle, the algorithm finds the most necessary activity from vector *A* and rounds the values to zero or one.

5.4. Primary State, fitness and constant increments

In any field, an evaluation of a system is a critical topic, regardless of whether it is done for designers of the system or purposes of autonomous learning; the development and progress of any system always depend on some kind of feedback. Since the *Need* nodes in the AA-FCM serve as indicators of success in particular tasks, their values can easily be converted into a general evaluation measure of the success (fitness) of an FCM.

In order to integrate the fitness into an AA-FCM, a *Primary State* (*PS*) node is added to the *State* nodes. Only *Need* nodes that are included in the evaluation of the system can affect the *PS* node. If the recommendation for the setting of the *Need* nodes is met (a value of zero indicates maximal satisfaction and no action is required), then all these relations are negative. Therefore, the higher the value of a *Need* node, the lower the fitness of an AA-FCM. Since fitness ranges from zero to one, its base value before computations has to be set to one, as the *Need* nodes reduce this value during its calculation. The relation of the *PS* node to itself w_{pp} is also equal to one, where *p* is the position of this node in an AA-FCM. Otherwise, if *Need* nodes are inverted, then their relations to the *PS* node are positive, the *PS* node starts at zero and $w_{pp} = 0$. It is possible to mix both approaches, but then the appropriate base value is harder to find.

For example, Table 3 shows a part of the matrix *R* of the AA-FCM in an artificial life model. This part contains three *Need* nodes and the *Primary State* node. The last row in the table contains the relations of all nodes to the *PS* node. Since the value of the *Need* nodes decreases with increasing satisfaction, these relations are negative, and *PS* node is set to 1.0 before computation begins. In this example, *Hunger* and *Thirst* are more important to success than *Reproductive Need*; they therefore have a much higher negative impact on the fitness of an individual.

Table 3
An example of matrix R of an FCM with fitness

	Hun.	**Thir.**	**Repro. N.**	**Prim. S.**	**…**
Hunger	0	0	0	0.05	
Thirst	0	0	0	0.1	
Reproductive Need	0	0	0	0.01	
Primary State	-0.5	-0.5	-0.2	1	
…					…

If the *PS* node starts at 1.0, then it can be used as a constant node, i.e. serving as a constant increment to any node. For the *PS* node s_p and a node c_i, the relation w_{ip} guarantees that a steady increase (or decrease) is added to node c_i in every computation. This relation is especially useful for the stable growth of *Need* nodes. For instance, the last column of Table 3 shows the positive relations of the *PS* node to the *Need* nodes. These relations simulate a constant increase in the needs over time.

5.5. Necessity

The necessity of actions provides the AA-FCM with another useful measure for decision-making. If several *Need* nodes have high values, then the decision-making prefers actions that relate to the most vital need. This feature is useful for the decision-making process in an AA-FCM with disabled parallel activities. For example, in the artificial life model, an individual with values of both *Hunger* and *Reproductive Need* of 1.0 will prefer activities that lead to the meeting of a more critical need. In the case shown in Table 3, the individual would select feeding rather than reproduction, since the *Hunger* need affects the *PS* more than the *Reproductive Need* does.

This feature uses states to evaluate the necessity of the *Need* nodes. The *PS* node is recommended, since at least one fitness value is required. More states can represent different fitness functions of the system, and the necessity feature covers even this possibility. At the first step, a designer identifies those *State* nodes that are used as fitness values and compares their importance to the system. The constant d_i represents these evaluations for all *State* nodes in the form of fuzzy values, where d_i = 1.0 means that *State* node s_i has the maximal importance and vice versa (see Eq. 27). For example, Eq. (28) shows the importance values for the AA-FCM with a *PS* node, a less important *State* node and a *State* node that does not serve as a fitness value (for instance an external input). Note that index i does not start at one for d_i values, since i represents position of the node in an AA-FCM that also contains other types of nodes.

$$\forall d_i \in [0,1] \tag{27}$$

$$\begin{aligned} d_5 &= 1.0 \\ d_6 &= 0.3 \\ d_7 &= 0.0 \end{aligned} \tag{28}$$

During initialization of the system, the necessity of *Need* nodes is calculated according to their influences on the *State* nodes and the corresponding values d_i (see Eq. (29)). The coefficients are also adjusted by an increment k_e that shifts their values to give a mean of 1.0 (see Eqs. (30, 31)).

$$e'_i = k_e + \sum_{j=(n+a)}^{(n+a+s)} \left(-w_{ji} \cdot d_{[j-(n+a)]}\right) \tag{29}$$

$$k_e = 1 - \frac{\sum_{i=0}^{n} (e'_i - k_e)}{n} \tag{30}$$

$$e_i = k_e + e'_i \tag{31}$$

Then, e_i is the final necessity coefficient of *Need* node n_i and e'_i is the necessity value e_i without the compensation of the offset k_e. In the computation of an AA-FCM, the necessity is used to calculate the *Activity* nodes from the *Need* nodes. Eq. (32) shows the placement of the necessity coefficient; otherwise, the equation is the same as Eq. (16).

$$a_i^t = f_a \left\{ \left(\sum_{j=1}^{n} \boldsymbol{e_j} \cdot w_{ij} \cdot n_j^{t-1} \right) + \left(\sum_{j=(n+1)}^{(n+a)} w_{ij} \cdot a_j^{t-1} \right) + \left(\sum_{j=(n+a+1)}^{(n+a+s)} w_{ij} \cdot s_j^{t-1} \right) \right\} \tag{32}$$

5.6. Summary of the solution

When the core of the proposed solution is enriched with all of the features mentioned above, Figure 2 illustrates the computation of the AA-FCM and the recommended order of its calculations.

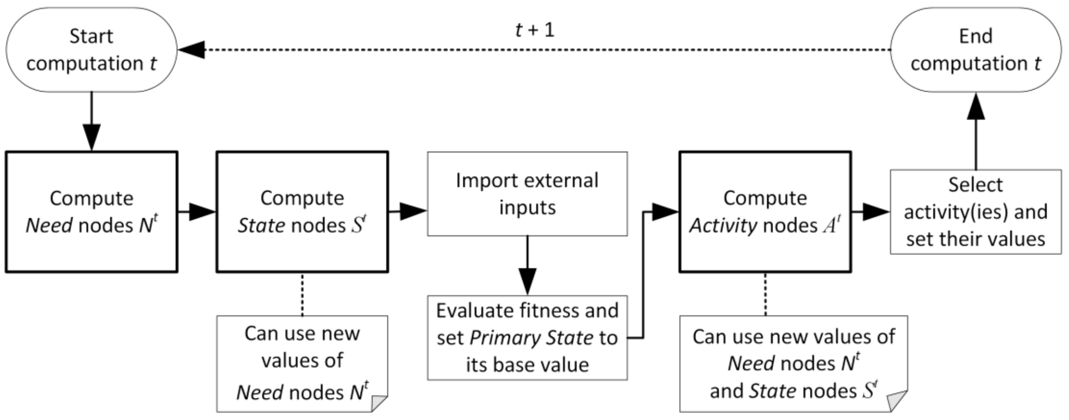

Start computation t
$t + 1$
End computation t
Compute *Need* nodes N^t
Compute *State* nodes S^t
Import external inputs
Compute *Activity* nodes A^t
Select activity(ies) and set their values
Can use new values of *Need* nodes N^t
Evaluate fitness and set *Primary State* to its base value
Can use new values of *Need* nodes N^t and *State* nodes S^t

Fig. 2 Summary of computation of the FCM for autonomous agents with the features described above

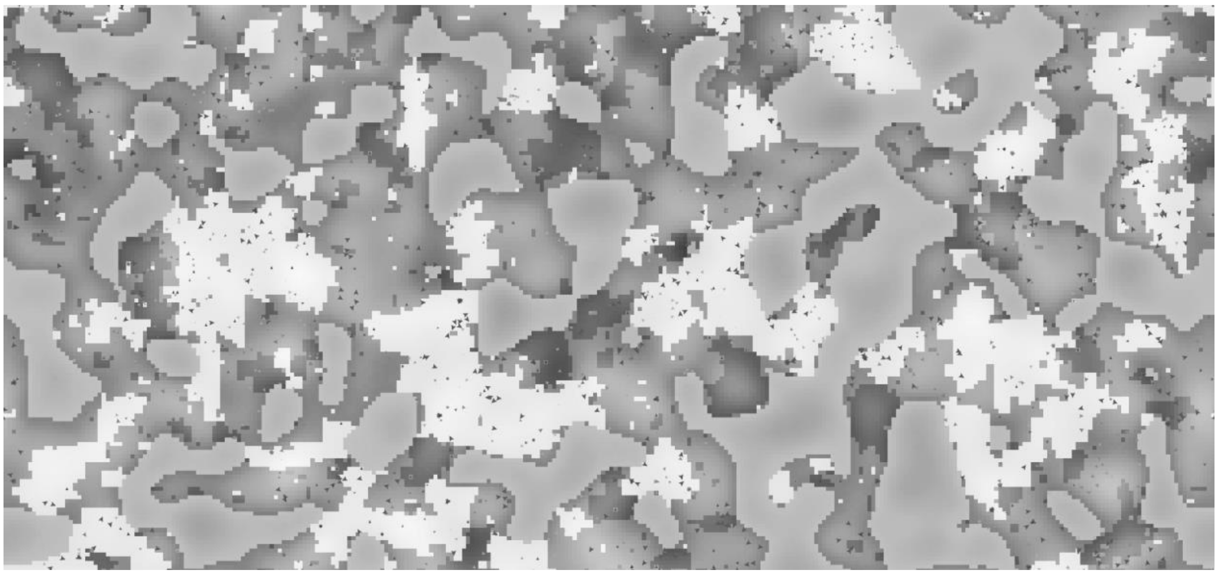


Fig. 3 Screenshot of the environment of the artificial life model (Nachazel 2018)

6. AA-FCM in an Artificial Life Model

To demonstrate the concept and test its abilities, the proposed solution was implemented into an artificial life model (ALModel) featuring a randomly generated two-dimensional environment with resources and thousands of individuals (see Figure 3). The model was built on the platform NetLogo 5, and is available for download at (Nachazel 2016). The simulation runs in discrete time steps (ticks), and uses an AA-FCM as an artificial intelligence method. Each individual makes a decision about its activity at the beginning of every tick. Parallel activities are disabled, meaning that individuals cannot perform more than one activity per time step.

The proposed approach manages whole agents' behavior. A single AA-FCM simulates needs, processes information from sensors, and performs decision-making. Each aspect of artificial life usually requires more than one node. For instance, the food management of individuals uses one node for the level of need (*Hunger*), two nodes for activities (*Feeding*, *Searching for Food*) and a *State* node as an external input, which allows currently available food supplies to be taken into consideration. More concept nodes in any FCM usually mean more possibilities, but also that the FCM becomes more demanding in terms of performance.

Hence, only the core areas of artificial life were implemented in the model. The behavior of individuals involves food, water, fatigue, danger, and reproduction. Artificial life requires evolution, and one of its elements is selection, which emerges through competition for resources. In this model, food and water are these vital resources. Fatigue is closely connected with the availability of resources, and its level affects the general performance of the individual. Reproduction is the next essential part of evolution. Finally, the *State* node *Danger* with the activity *Escape* enables individuals to escape from predators; this is the only defense for most of the individuals.

Table 4 contains all of the concept nodes distributed into classes. *Need* nodes are set in the recommended way, meaning that a value of one represents the highest need, and zero means maximal satisfaction. *State* nodes that represent the availability of a resource or partner are inverted to the lack of these entities, so a negative relation is used instead of a positive one; in other words, the relation "*Food availability* causes *Feeding*" is replaced by "*Lack of food* prevents *Feeding*," which is a more accurate representation.

Table 4
Classification of concept nodes in the ALModel

Need nodes	**Activity nodes**	**State nodes**
Hunger (n_1)	Feed (a_5)	Primary State (s_{13})
Thirst (n_2)	Drink (a_6)	Lack of Food (s_{14})
Fatigue (n_3)	Sleep (a_7)	Lack of Water (s_{15})
Reproductive N. (n_4)	Reproduce (a_8)	Lack of Partners (s_{16})
	Search for Food (a_9)	Danger (s_{17})
	Search for Water (a_{10})	
	Search for Partner (a_{11})	
	Escape (a_{12})	

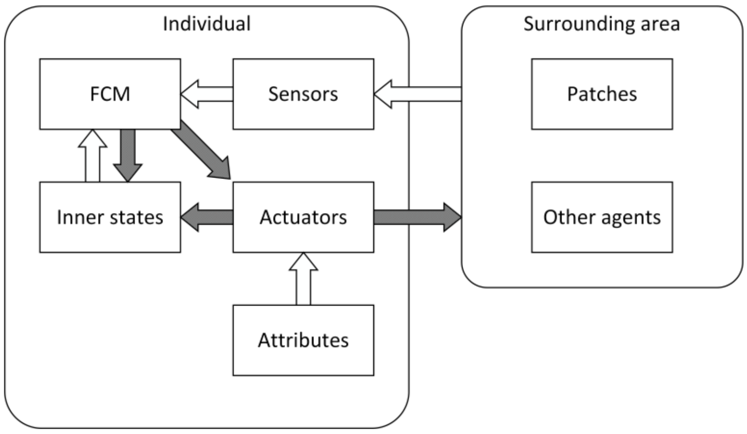


Fig. 4 Diagram of an individual in the ALModel (Nachazel 2015)

Figure 4 depicts an individual in the model; white arrows represent a flow of information, while dark arrows express a direction of influence or method call. The AA-FCM reads data from sensors and inner states, and then makes a decision according to the current situation and sends the request to actuators. It manages behavior at a higher level, and the actuators try to fulfill the command within a given environment. The actuators perform the selected action while taking into consideration the individual's vicinity and attributes; for example, the AA-FCM decides to search for food, and the actuators then direct the individual to the closest food source, if this is within sight, or alternatively to the location where the individual fed last time. If there is an obstacle (water) in the way, the actuators need to perform more difficult path-finding than just heading directly to the target location.

Table 5
Initial matrix of relations *R* in the ALModel

	n_1	n_2	n_3	n_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	s_{13}	s_{14}	s_{15}	s_{16}	s_{17}
Hunger	0	0	0	0	-1	0	-0.005	0.3	0	0	0	0	0.01	0	0	0	0
Thirst	0	0	0	0	0.05	-1	-0.005	0.3	0	0	0	0	0.01	0	0	0	0
Fatigue	0.001	0.001	0	0	0	0	-0.2	0.3	0.005	0.005	0.005	0.1	0.005	0	0	0	0
Reproductive N.	0	0	0	0	0	0	0	-1	0	0	0	0	0.002	0	0	0	0
Feed	1	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0	-0.2
Drink	0	1	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	-0.2
Sleep	0	0	0.8	0	0	0	0.2	0	0	0	0	0	0	0	0	0	-0.8
Reproduce	-0.2	-0.2	-0.2	1	0	0	0	0	0	0	0	0	0	0	0	-1	-0.8
Search for Food	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Search for Water	0	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S. for Partner	-0.2	-0.2	-0.2	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0
Escape	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Primary State	-0.5	-0.55	-0.35	-0.1	0	0	0	0	0	0	0	0	1	0	0	0	0
Lack of Food	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lack of Water	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lack of Partners	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Danger	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6
The decision-making section of matrix *R* of an individual at a later stage of a model run (generation 89)

	n_1	n_2	n_3	n_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	s_{13}	s_{14}	s_{15}	s_{16}	s_{17}
Feed	0.8	-0.2	0.2	0.2	-0.3	0.0	-0.1	-0.2	0.0	0.1	-0.1	0.1	0.0	-0.9	0.2	-0.2	-0.5
Drink	-0.1	1.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.4	-0.2	-0.1	-0.1	0.1	-0.4	0.2	-0.2
Sleep	-0.2	0.1	0.9	0.0	0.1	0.2	-0.1	0.0	0.1	-0.2	0.1	0.4	0.1	0.2	0.1	0.0	-0.8
Reproduce	0.0	-0.2	-0.5	1.0	0.0	-0.3	0.0	-0.2	-0.1	0.3	0.2	-0.1	-0.2	-0.1	-0.2	-0.8	-0.9
S. for Food	0.3	0.1	0.1	0.0	0.0	-0.3	0.3	0.1	-0.1	0.0	-0.1	0.0	-0.2	-0.1	0.0	-0.2	-0.1
S. for Water	0.2	0.8	0.0	0.1	0.2	-0.1	0.1	0.2	0.2	0.0	0.0	-0.2	-0.1	-0.1	0.0	0.2	0.0
S. for Partner	-0.1	-0.5	-0.1	0.7	0.2	0.2	0.2	-0.1	-0.2	-0.1	0.0	-0.3	0.0	0.2	0.0	-0.2	0.3
Escape	0.2	0.0	0.0	-0.2	-0.1	0.1	-0.1	0.0	0.1	-0.4	0.0	0.0	-0.1	-0.1	0.2	0.2	0.9

Table 5 shows the initial settings of matrix *R*. The top section (four rows of *Need* nodes) contains all relations to the *Need* nodes. This section represents a set of rules that determine the increase in the needs and the effects of activities in the model. These values are affected by the granularity setting of a model run.

The middle section (eight rows of *Activity* nodes) represents the decision-making part of the AA-FCM. In the ALModel, this section changes dynamically during a model run, through evolution. Its initial setting is therefore not essential, because the responsibility for correct behavior moves from the designer to the evolution of the model. Regardless, due to the dynamic environment, the optimal behavior at the beginning usually differs from the optimal behavior at later stages of a model run. Moreover, various species often require different behavior patterns that better suit their attributes and type of diet.

Finally, the bottom section (five rows of *State* nodes) contains mostly zeros, since four of these *State* nodes are external inputs. The *PS* node begins computation at a value of 1.0, and is then lowered by *Need* nodes. The AA-FCM uses this node for the fitness of individuals and the necessity feature. Since this is the only fitness node, its importance value d_1 is 1.0, while the values of the other *State* nodes are 0.0.

Table 6 shows the middle section of the AA-FCM after 89 generations. This particular matrix *R* represents a member of a successful small omnivore species. A few relations did not prove to be useful and lost their influence on behavior, for example the negative relation from *Hunger* to the activity *Reproduce* or the positive relation of activity *Sleep* to itself. On the other hand, several new relations arose. For instance, this species learned over generations that taking a rest is beneficial immediately after escaping a predator. Also, while searching for water, this species are less likely to flee if they spot a predator. Since predators often gather near water sources, thirsty individuals have to come closer to them than they would normally allow in any other situation.

Figure 5 shows 100 time steps of a single individual in the model. In this case, the granularity was set to 0.6. The top plot in the figure depicts the development of the *Need* nodes and fitness. The bottom plot explains the changes in the values in the plot above with a log of the activities that this individual performed during the observed 100 time steps. During the first 25 time steps, the individual struggled to find resources; *Need* nodes were incerasing, and its fitness was decreasing. This individual finally found resources, rested and reproduced. Reproduction raised its *Need* nodes due to the corresponding values set in matrix *R* (see Table 5, column a_8). Except for *Sleep*, *Escape*, and *Searching*, all activities floor the value of the corresponding *Need* node in a single time step.

Individuals are capable of surviving over hundreds of thousands of time steps, increasing the population and evolving. Verifying the solution on a larger scale of an entire population is problematic, because the *PS* node enables only the evaluation of individuals. In the simulation, this fitness node is not essential for the survivability of the entire species. Due to natural selection and the aggressive competition in the dynamic environment, successful species tend to prefer the survivability of the species as a whole over individual satisfaction; however, the ALModel offers various other measures.

The ratio of deaths caused by a lack of resources to total deaths offers one possibility for verifying the adaptation of the AA-FCM in the model. Obviously, this ratio depends on the availability of resources in the dynamic environment. However, Figure 6 shows the development of the ratio during a model run with a stable environment. In order to cancel out any other influences except adaptation, this environment maintained resources at stable levels and contained a constant number of individuals from only one species. The decrease in this ratio proves that the artificial intelligence of the individuals is capable of adaptation, even at the population scale.

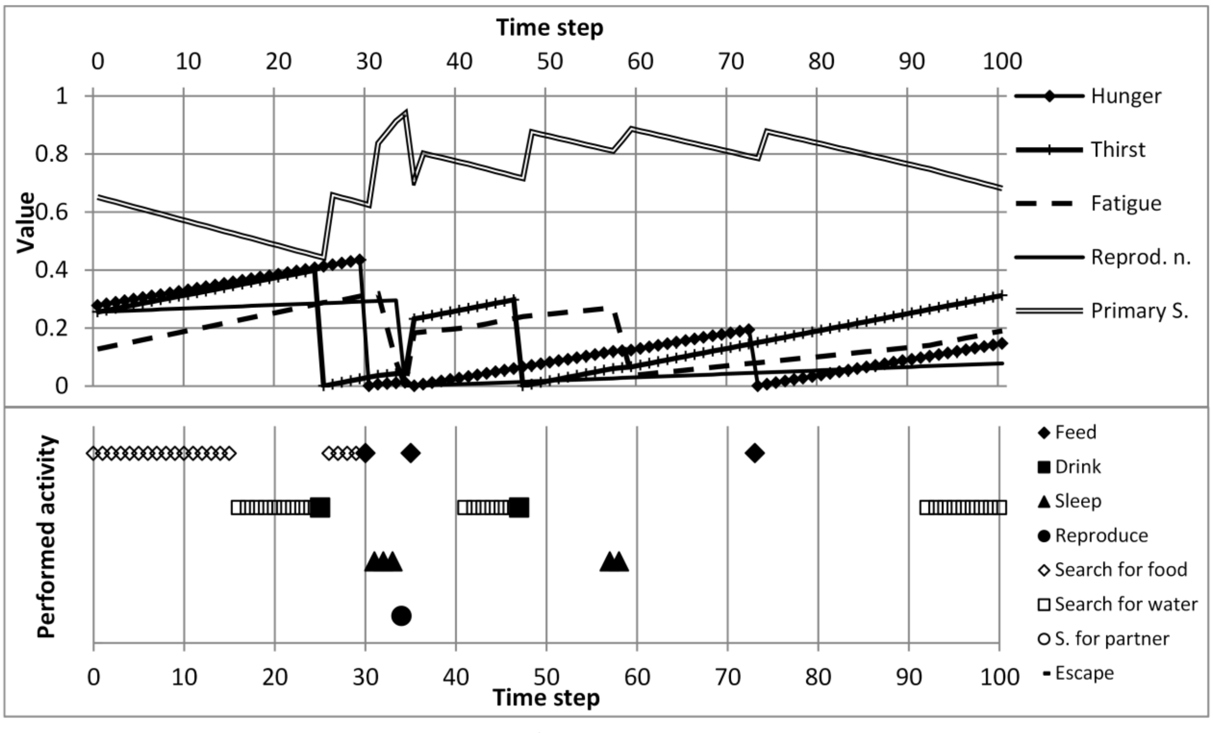


Fig. 5 *Need* and *Activity* nodes of an individual in the ALModel over 100 time steps

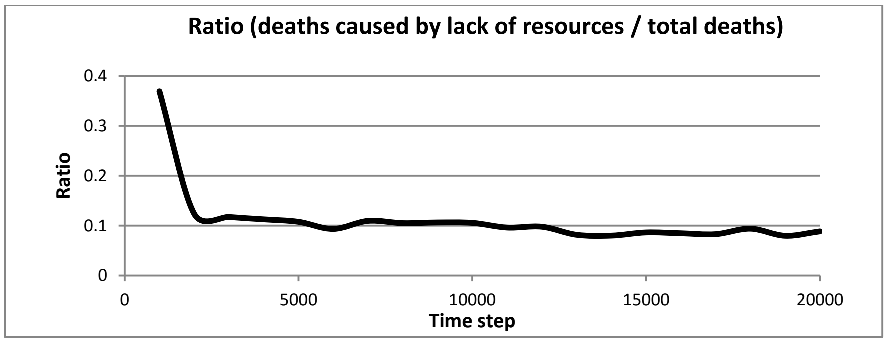


Fig. 6 Development of the ratio of deaths caused by lack of resources to total deaths

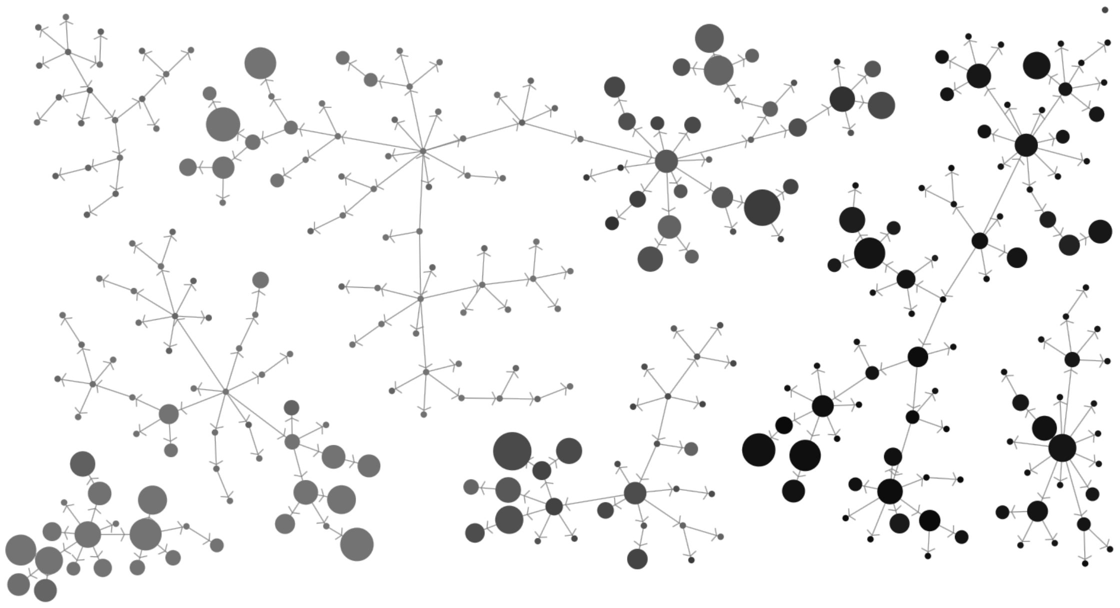


Fig. 7 Phylogenetic trees of species generated from a single run of the artificial life model

With evolutionary adaptation, the genomes of individuals (their attributes and behavior) develop during a model run. When the genome of an offspring is sufficiently differentiated from the original genome of its species, it founds a new species, and this process generates a phylogenetic tree. Figure 7 shows phylogenetic trees of species produced in a single model run, in which each bubble represents a different species. The arrows point to the species that evolved from the species on the other end of the arrow, and sizes of the bubbles represent the numbers of individuals that were classified as members of that species at the moment of creating the graph (small dots correspond to extinct species). The brightness of the bubbles depicts a certain attribute of a species; in this case, it is the type of diet (a black bubble means a carnivore species; a brighter one a herbivore species). The experimental results presented here prove that this decision-making method allows for rational behavior with multiple objectives, learning and interactions.

7. Discussion and Future Work

This approach offers a wide scale of possible usages and modifications for adjusting to a specific environment. In the past, the basis of this approach was developed for an ambient intelligence system in the limited virtual environment of Second Life (unfortunately, this has not yet been published in English, and the thesis in (Nacházel 2012) is the only source). It was later extended with evolutionary principles, allowing individuals to learn over generations (Nachazel 2018, 2015). The modularity of the AA-FCM method allows for the modification that replaces the decision-making part of AA-FCM with AHP (Nachazel 2018); this modification is focused on the computational speed of processing, and while it is significantly faster, it slightly lowers the complexity, leading to the poorer learning possibilities of the method. This may be advantageous for some applications, especially in large scale models with thousands of individuals. Due to its modularity, other modifications may be possible for specific objectives such as a network of FCMs, dynamic matrix sizes, and so on.

This method is currently being implemented for the simulation of human activity, which can aid in experiments with smart systems of ambient intelligence. For this purpose, it was connected to a schedule planner to better simulate human routines and planning. The result is cooperation between the dynamic schedule, which covers all planned activities, and the AA-FCM, which handles needs, unplanned activities and emergencies.

For more complex systems, the transformation of a large FCM into several smaller ones seems to be an appropriate way of modeling complex structures with a low density of relations. A group of FCMs structured as a hierarchy or a network recomputes all concept nodes much more quickly than one large FCM with the same number of nodes. Different levels of control can also be separated, with clear responsibilities. In combination with the AA-FCM modification, this distribution undoubtedly has great potential in the modeling of autonomous agents.

8. Conclusion

The paper proposes a new approach to the design of FCMs for autonomous agents. The AA-FCM modification distributes concepts into three classes according to their purpose within the map, and these classes differ in their calculation and interpretation. This design offers several advantages and features that are very useful for various systems dealing with a dynamic environment.

This paper provides the mathematical background of the proposed design and gives several examples for clarity. In addition to the core of the AA-FCM, it presents many optional attachments that provide additional help in designing autonomous agents. The paper also presents the complete design of the AA-FCM for the artificial life model, together with testing of this modification in the final model. This testing verified the solution and all of its features, proving both its functionality and evolutionary adaptation. Although this adaptation is not a part of the proposed modification, FCMs usually require a certain learning or training process. This paper verifies that the AA-FCM supports evolutionary algorithms. Moreover, due to its clear classification and modularity, any combination with other algorithms or extensions is even easier to implement than with the original general FCM.

The current development of the modification focuses on its combination with other methods which can provide additional features or improve existing ones. There are already modifications of this method; a combination with AHP improves the performance of a model with thousands of agents, and with the addition of schedule management, AA-FCM can believably simulate human activities, including both routine and unplanned, spontaneous activities.

In terms of its complexity and difficulty of development, AA-FCM lies between simple rule sets and neural networks. It is complex, powerful and has a high learning ability while retaining human readability and avoiding the “black box” problem of neural networks. Moreover, its modularity offers great potential for a wide range of uses in the modeling of autonomous agents.

9. Acknowledgments

Support from the Specific Research Project "Socio-economic models and autonomous systems" in Faculty of Informatics and Management, University of Hradec Králové, is gratefully acknowledged.

References

Acampora, G., Loia, V., & Vitiello, A. (2011). Distributing emotional services in Ambient Intelligence through cognitive agents. [journal article]. *Service Oriented Computing and Applications, 5*(1), 17-35, doi:10.1007/s11761-011-0078-7.

Ahmadi, S., Yeh, C. H., Papageorgiou, E. I., & Martin, R. (2015). An FCM-FAHP approach for managing readiness-relevant activities for ERP implementation. [Article]. *Computers & Industrial Engineering, 88*, 501-517, doi:10.1016/j.cie.2015.07.006.

Barón, H., Crespo, R., Pascual Espada, J., & Martínez, O. (2014). Assessment of learning in environments interactive through fuzzy cognitive maps. *Soft Computing*, 1-14.

Furfaro, R., Fink, W., & Kargel, J. S. (2012). Autonomous real-time landing site selection for Venus and Titan using Evolutionary Fuzzy Cognitive Maps. *Applied Soft Computing, 12*(12), 3825-3839, doi:http://dx.doi.org/10.1016/j.asoc.2012.01.014.

Giabbanelli, P. J., & Crutzen, R. (2014). Creating groups with similar expected behavioural response in randomized controlled trials: a fuzzy cognitive map approach. [journal article]. *BMC Medical Research Methodology, 14*(1), 1-19, doi:10.1186/1471-2288-14-130.

Gras, R., Golestani, A., Hosseini, M., Khater, M., Farahani, Y., Mashayekhi, M., et al. Ecosim: an individual-based platform for studying evolution. In *European Conference on Artificial Life, 2011* (pp. 284-286)

Hsieh, Y.-H., Chen, I.-H., & Yuan, S.-T. (2013). FCM-based customer expectation-driven service dispatch system. [journal article]. *Soft Computing, 18*(2), 359-378, doi:10.1007/s00500-013-1063-1.

Jayashree, L. S., Palakkal, N., Papageorgiou, E. I., & Papageorgiou, K. (2015). Application of fuzzy cognitive maps in precision agriculture: a case study on coconut yield management of southern India's Malabar region. [journal article]. *Neural Computing and Applications, 26*(8), 1963-1978, doi:10.1007/s00521-015-1864-5.

Jones, R. T., Connors, E., Mossey, M., Hyatt, J., Hansen, N., & Endsley, M. (2011). Using fuzzy cognitive mapping techniques to model situation awareness for army infantry platoon leaders. *Computational and Mathematical Organization Theory, 17*(3), 272-295, doi:10.1007/s10588-011-9094-6.

Jose, A. (2010). Dynamic Fuzzy Cognitive Maps for the Supervision of Multiagent Systems. In M. Glykas (Ed.), *Fuzzy Cognitive Maps: Advances in Theory, Methodologies, Tools and Applications* (pp. 307-324): Springer Berlin Heidelberg.

Khater, M., & Gras, R. (2012). Adaptation and Genomic Evolution in EcoSim. In T. Ziemke, C. Balkenius, & J. Hallam (Eds.), *From Animals to Animats 12: 12th International Conference on Simulation of Adaptive Behavior, SAB 2012, Odense, Denmark, August 27-30, 2012. Proceedings* (pp. 219-229): Springer Berlin Heidelberg.

Kosko, B. (1986). Fuzzy cognitive maps. *International Journal of Man-Machine Studies, 24*(1), 65-75, doi:http://dx.doi.org/10.1016/S0020-7373(86)80040-2.

Kyriakarakos, G., Patlitzianas, K., Damasiotis, M., & Papastefanakis, D. (2014). A fuzzy cognitive maps decision support system for renewables local planning. *Renewable and Sustainable Energy Reviews, 39*, 209-222, doi:10.1016/j.rser.2014.07.009.

Laureano-Cruces, A. L., & Rodriguez-Garcia, A. (2011). Design and implementation of an educational virtual pet using the OCC theory. [journal article]. *Journal of Ambient Intelligence and Humanized Computing, 3*(1), 61-71, doi:10.1007/s12652-011-0089-4.

Mendonça, M., Arruda, L. V. R. d., & Neves-Jr, F. (2014). Cooperative Autonomous Agents Based on Dynamical Fuzzy Cognitive Maps. In I. E. Papageorgiou (Ed.), *Fuzzy Cognitive Maps for Applied Sciences and Engineering: From Fundamentals to Extensions and Learning Algorithms* (pp. 159-175): Springer Berlin Heidelberg.

Mendonça, M., de Arruda, L. V. R., Chrun, I. R., & da Silva, E. S. Hybrid Dynamic Fuzzy Cognitive Maps Evolution for autonomous navigation system. In *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2-5 Aug. 2015 2015* (pp. 1-7). doi:10.1109/FUZZ-IEEE.2015.7337855.

Motlagh, O., Jamaludin, Z., Tang, S. H., & Khaksar, W. (2015). An agile FCM for real-time modeling of dynamic and real-life systems. [journal article]. *Evolving Systems, 6*(3), 153-165, doi:10.1007/s12530-013-9077-6.

Nachazel, T. Optimization of Decision-Making in Artificial Life Model Based on Fuzzy Cognitive Maps. In *Intelligent Environments (IE), 2015 International Conference on, 15-17 July 2015 2015* (pp. 136-139). doi:10.1109/IE.2015.28.

Nachazel, T. (2016). NetLogo User Community Models: ALModel. http://ccl.northwestern.edu/netlogo/models/community/ALModel.

Nachazel, T. (2018). Analytic hierarchy process in artificial life model based on fuzzy cognitive maps. *Journal of Ambient Intelligence and Smart Environments, 10*, 127-141, doi:10.3233/AIS-180480.

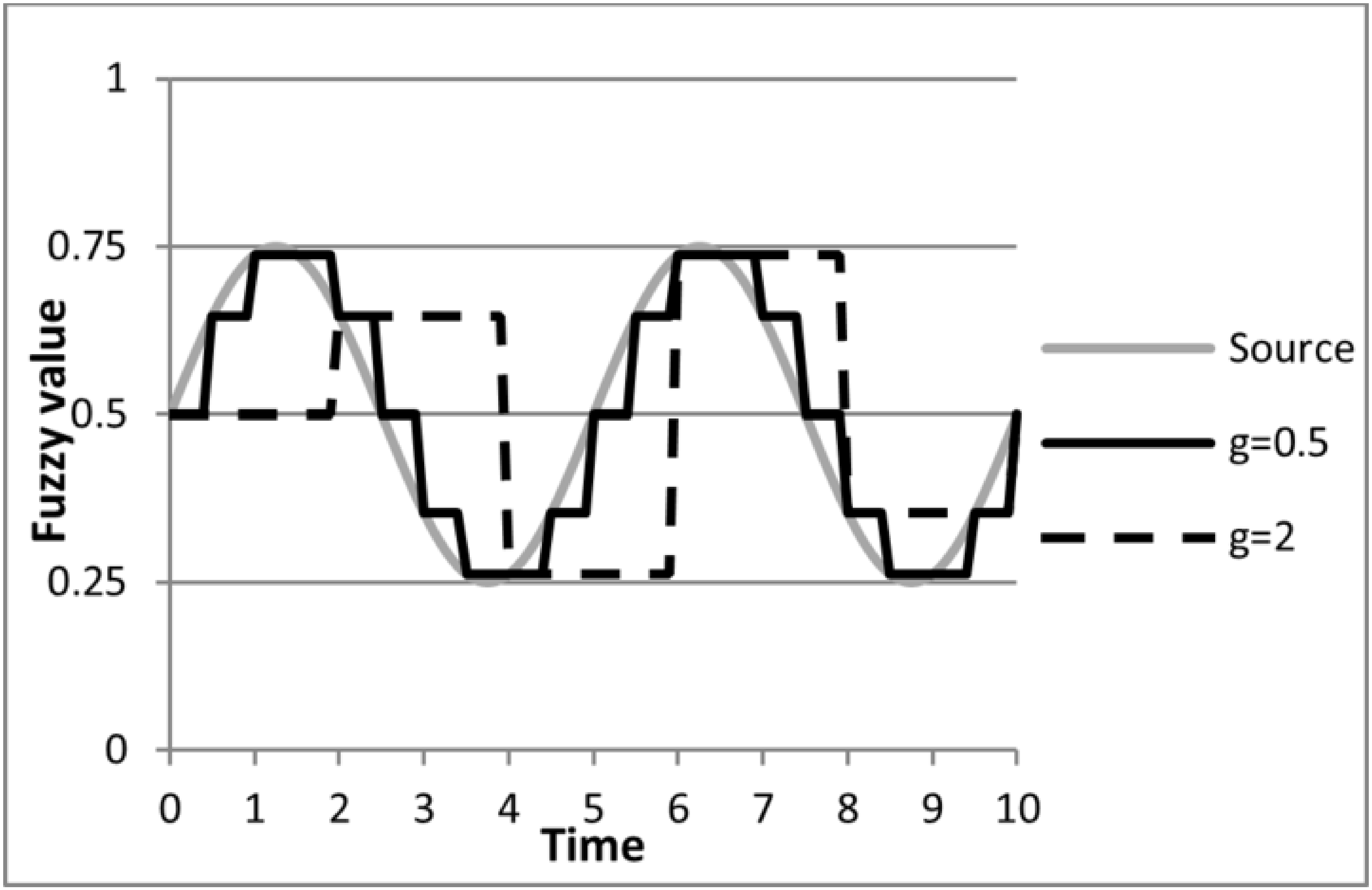
Nacházel, T. (2012). *Inteligentní systémy ve virtuálním prostředí*. Bachelor thesis, University of Hradec Králové, Hradec Králové.

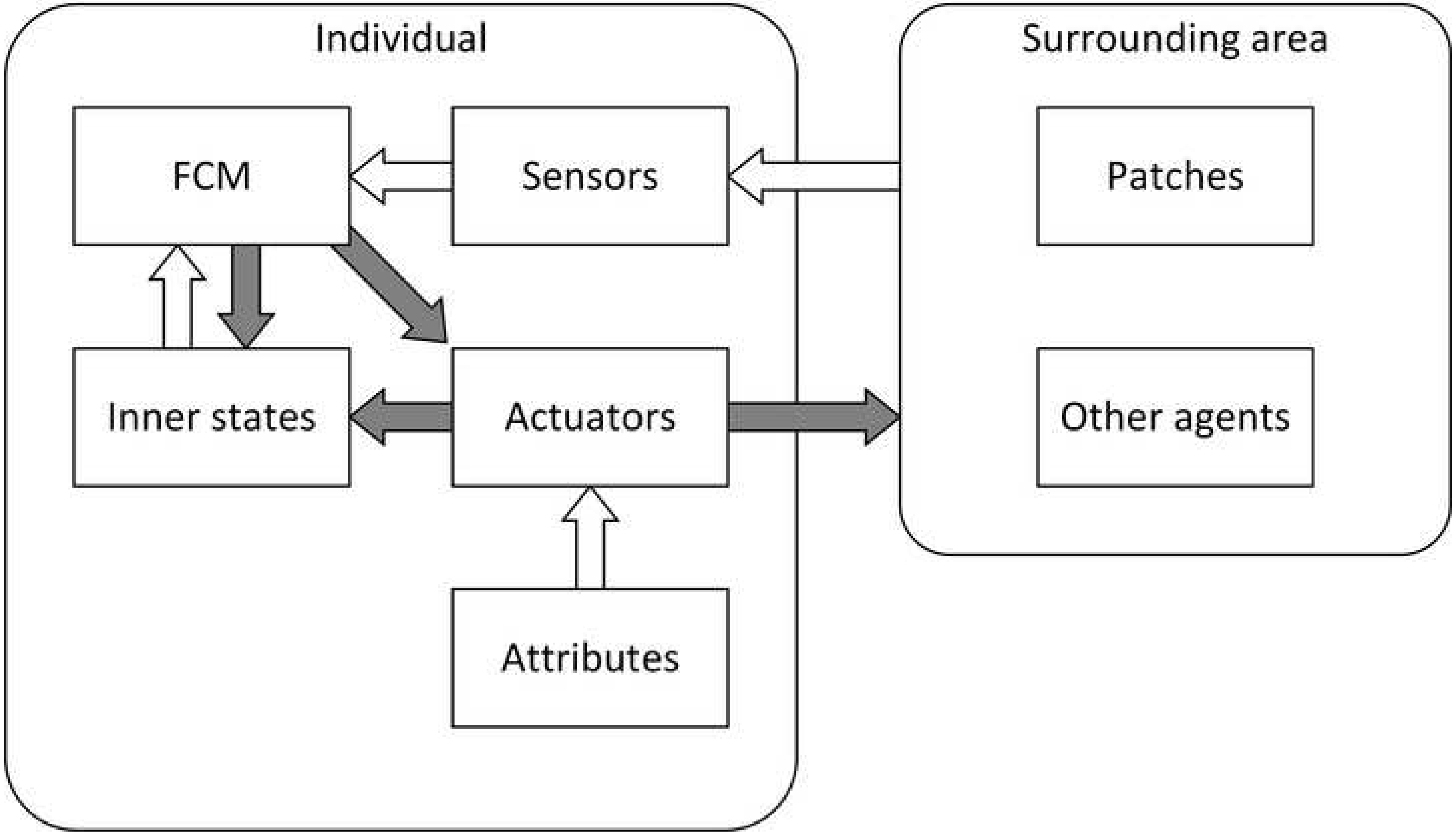
Senniappan, V., Subramanian, J., Papageorgiou, E. I., & Mohan, S. (2016). Application of fuzzy cognitive maps for crack categorization in columns of reinforced concrete structures. [journal article]. *Neural Computing and Applications*, 1-11, doi:10.1007/s00521-016-2313-9.

Stach, W., Kurgan, L., & Pedrycz, W. (2010). A divide and conquer method for learning large Fuzzy Cognitive Maps. *Fuzzy Sets and Systems, 161*(19), 2515-2532, doi:http://dx.doi.org/10.1016/j.fss.2010.04.008.

Stula, M., Krstinic, D., & Seric, L. (2011). Intelligent forest fire monitoring system. [journal article]. *Information Systems Frontiers, 14*(3), 725-739, doi:10.1007/s10796-011-9299-8.

Szwed, P., Skrzynski, P., & Chmiel, W. (2014). Risk assessment for a video surveillance system based on Fuzzy Cognitive Maps. [journal article]. *Multimedia Tools and Applications*, 1-24, doi:10.1007/s11042-014-2047-6.


1
0.75
0.5
0.25
0
Fuzzy value
0
1
2
3
4
5
6
7
8
9
10
Time
Source
g=0.5
g=2


Individual
FCM
Sensors
Inner states
Actuators
Attributes
Surrounding area
Patches
Other agents

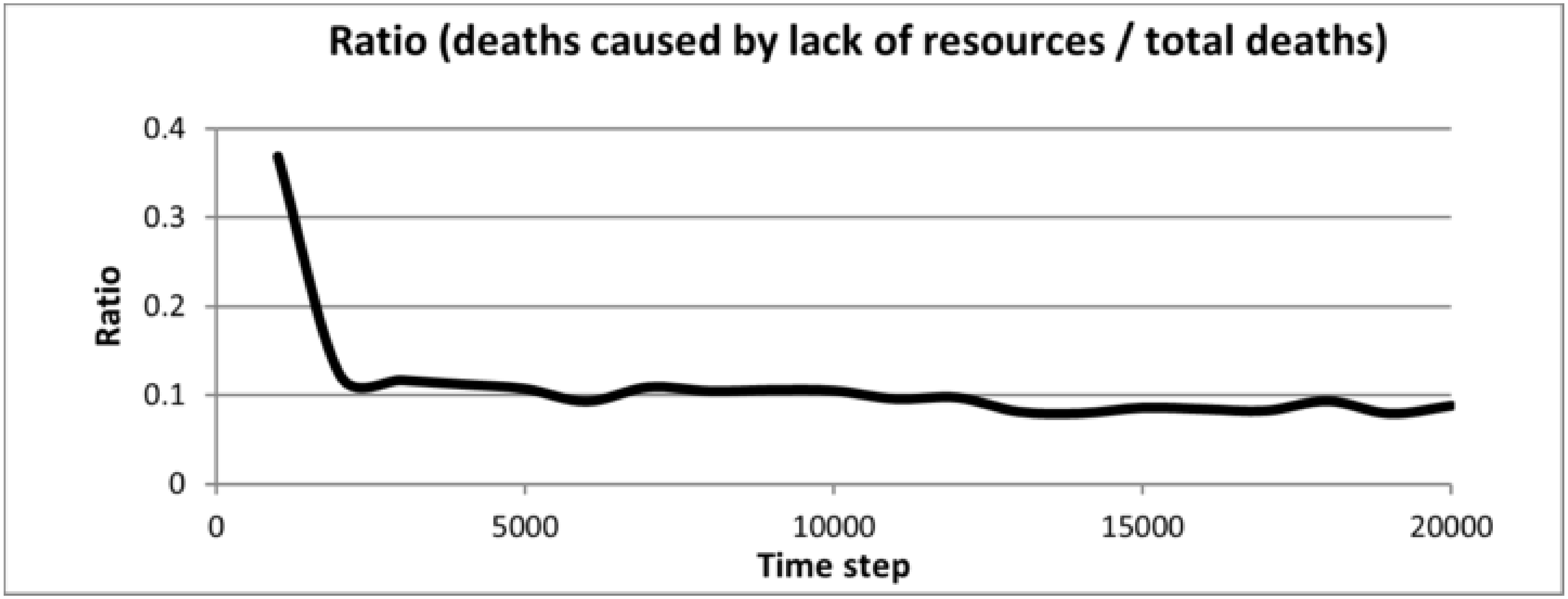

Ratio (deaths caused by lack of resources / total deaths)
Ratio
0.4
0.3
0.2
0.1
0
0
5000
10000
15000
20000
Time step

Fig3 model

Click here to download Figure ModelScreenshot.png

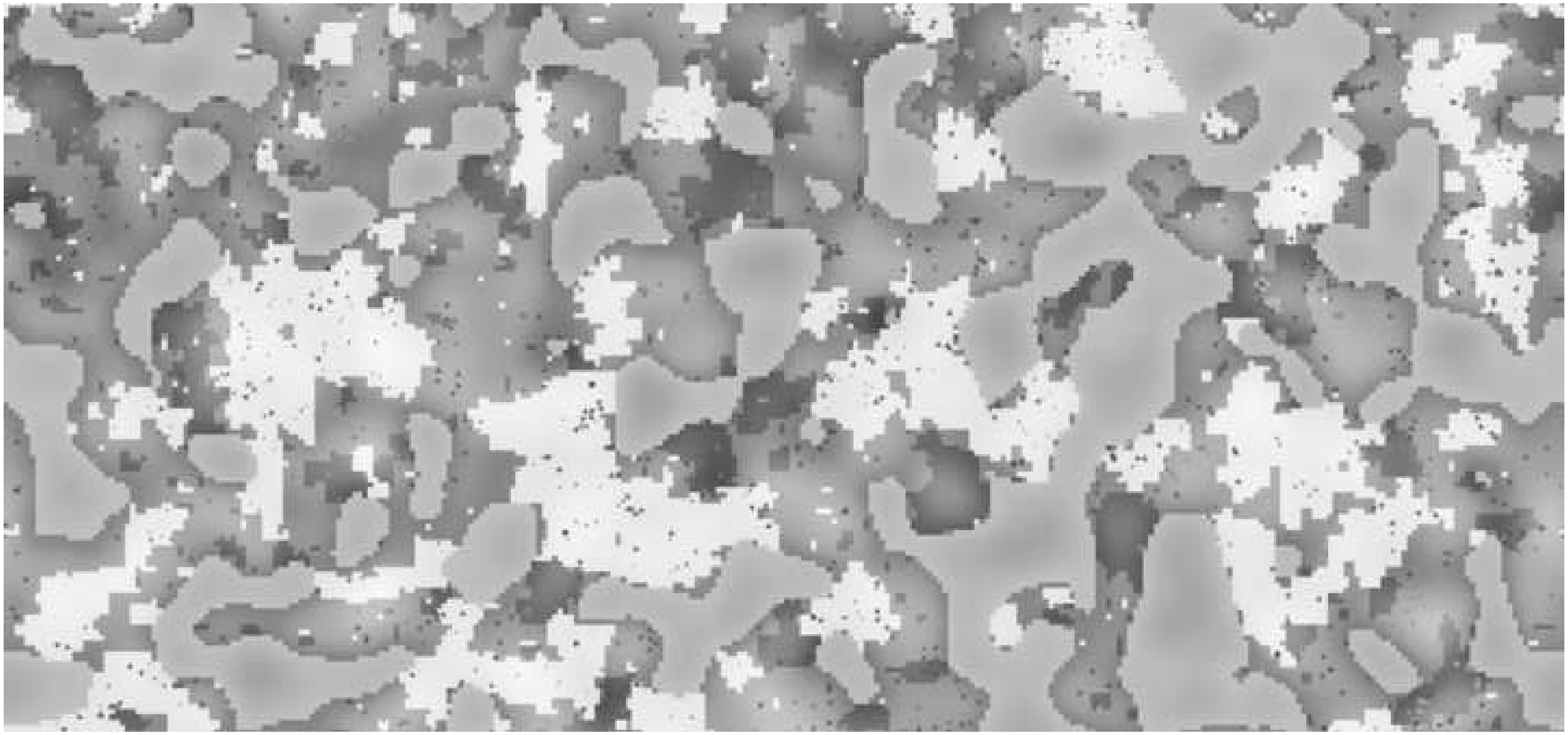
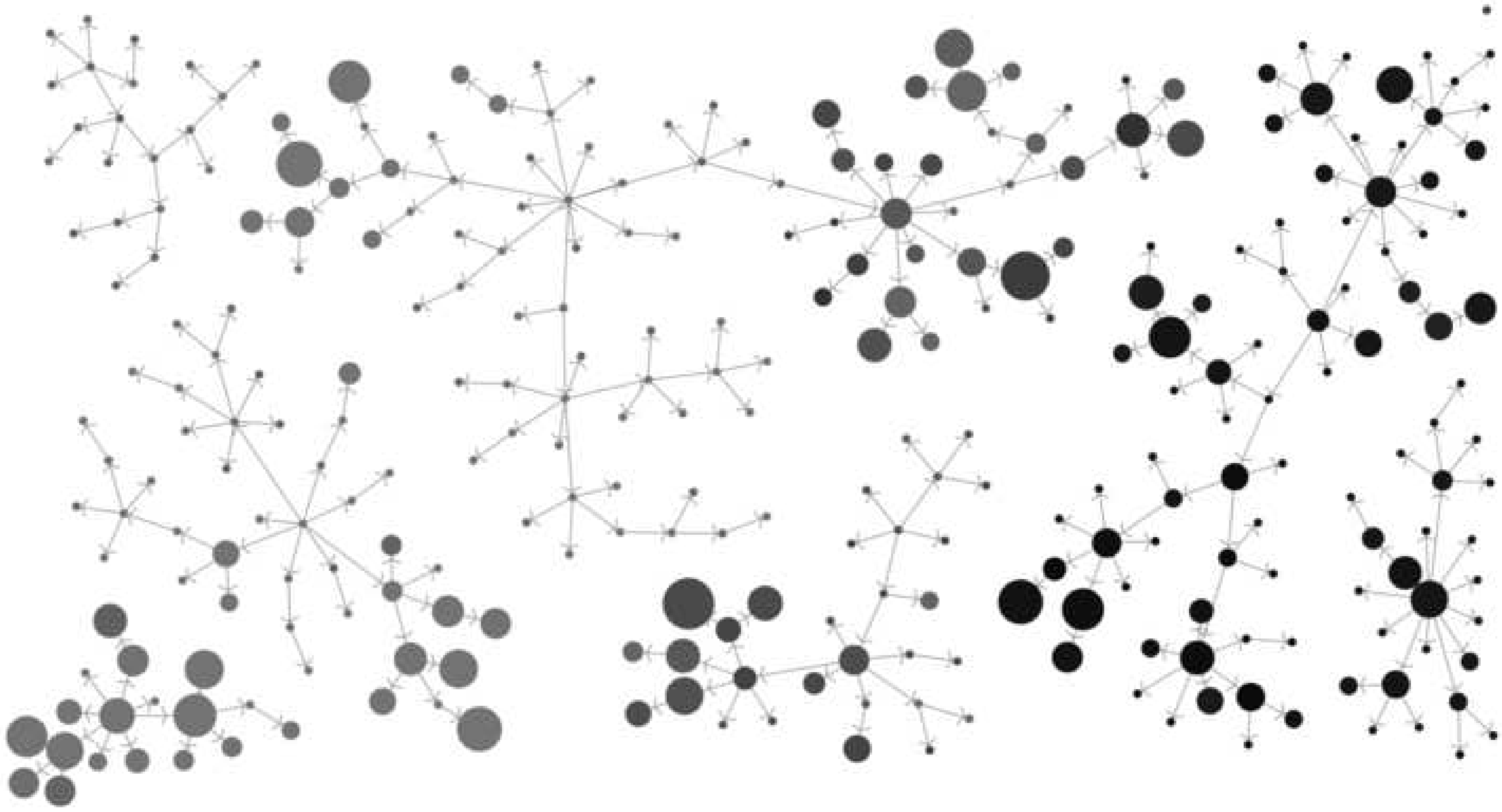


Fig7 phylogenetic trees

Click here to download Figure FCM-1800.png



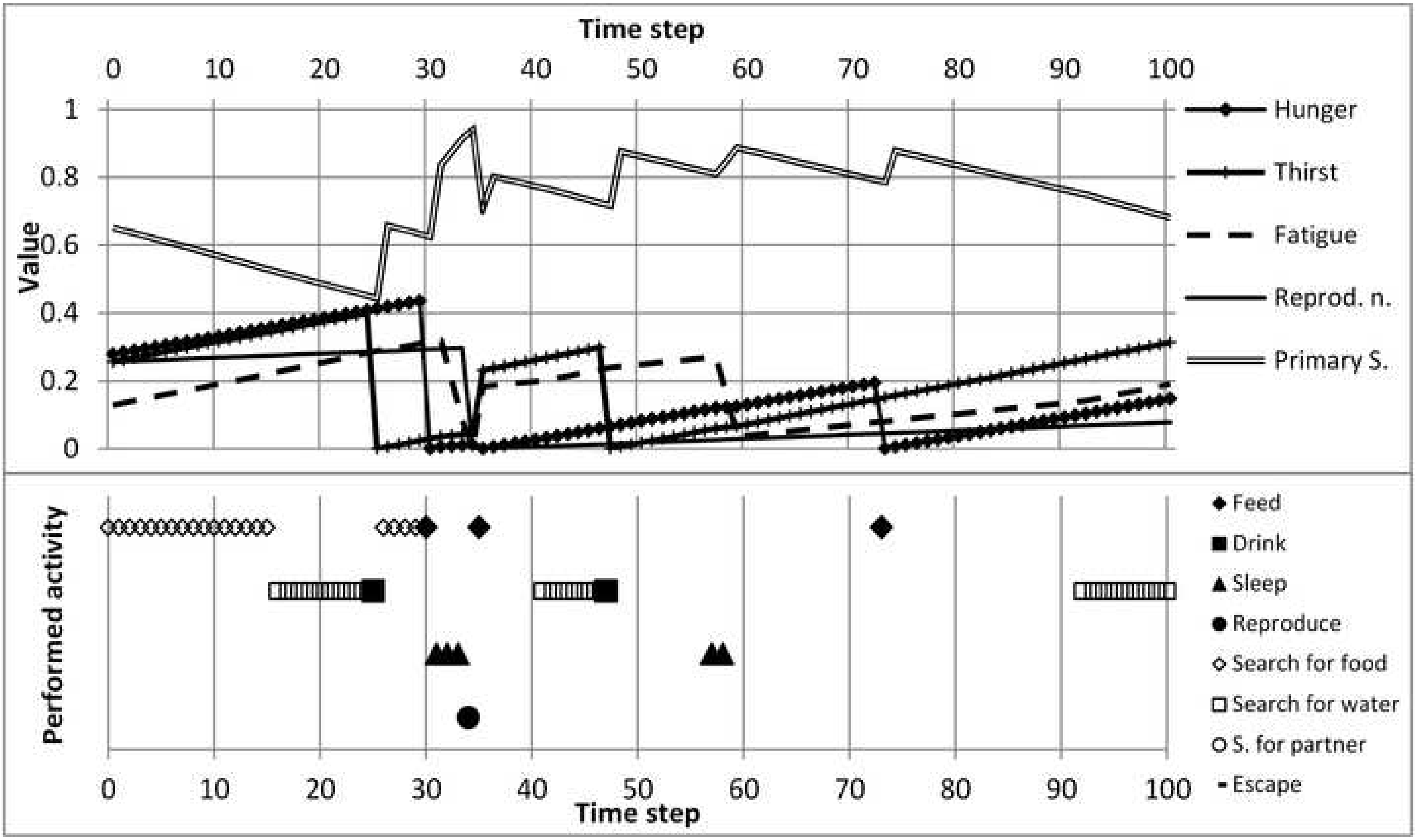
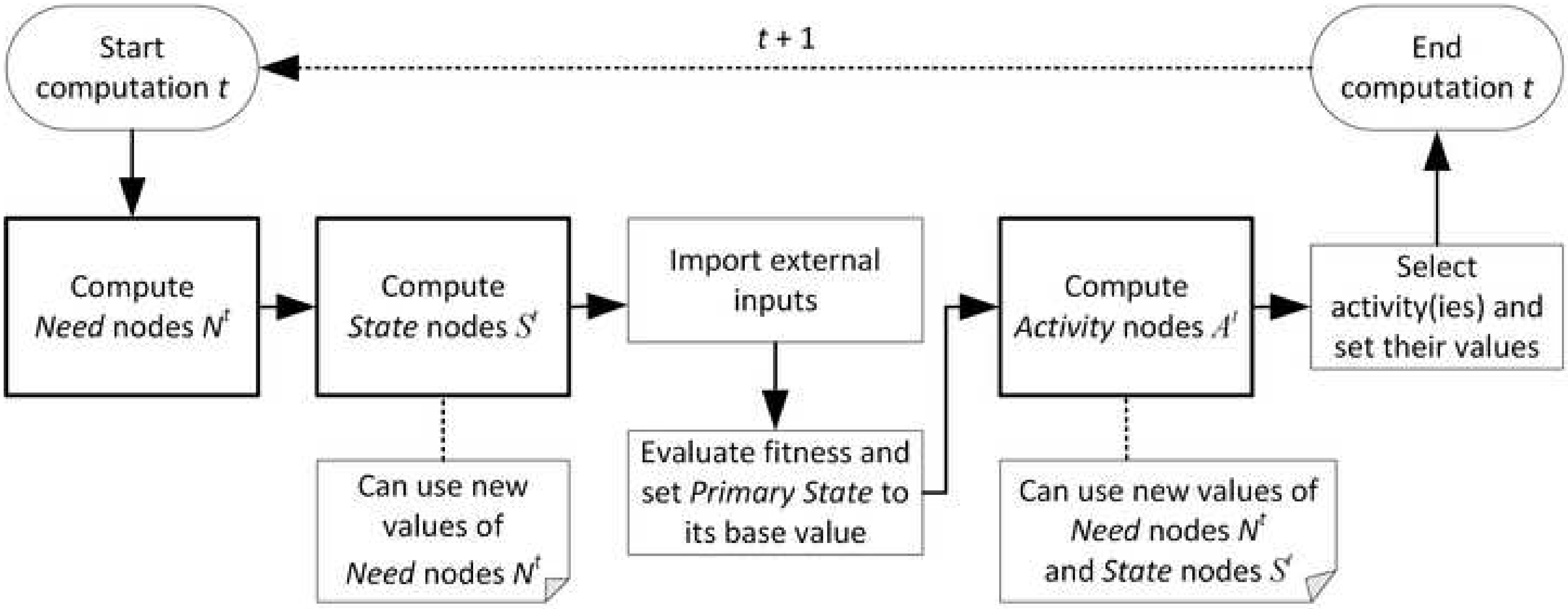

Time step
0
10
20
30
40
50
60
70
80
90
100
Value
1
0.8
0.6
0.4
0.2
0
Hunger
Thirst
Fatigue
Reprod. n.
Primary S.
Performed activity
Feed
Drink
Sleep
Reproduce
Search for food
Search for water
S. for partner
Escape
Time step

Fig2 algorithm

Click here to download Figure Procedure2.png


Start computation t
$t + 1$
End computation t
Compute *Need* nodes N^t
Compute *State* nodes S^t
Import external inputs
Evaluate fitness and set *Primary State* to its base value
Compute *Activity* nodes A^t
Select activity(ies) and set their values
Can use new values of *Need* nodes N^t
Can use new values of *Need* nodes N^t and *State* nodes S^t