The CSI Journal on
Computer Science and Engineering
Vol. 14, No. 2, 2017
Pages 32-49
**Regular Paper**

# Simulation of Fuzzy Intelligent Agents in Stone-Age Virtual Ecosystem

Ali Ahmadi[1] Bahman Aminipour[2]

[1]Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran
[2]Department of Electrical and Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

## Abstract

Artificial life (A-Life) is a relatively new research paradigm which investigates the life-like systems, their processes, and their evolution, through the use of simulations. Several A-Life simulators have been already presented in the literature. These simulators are mostly designed for studying the behaviors of living organisms based on a world of agents, rules and objects. In these simulations, the agent's experience along with simple reinforcement learning algorithms are used to produce complex behaviors. However, such worlds are not suitable for cognitive studies focused on complex human-like behaviors, mostly because of their slow calculation speed and discrete model structures. This paper proposes a new simulator, called Stone-age, for simulating and evaluating the behavior of the agents in the environment. Generally speaking, Stone-age simulates a form of ecosystem, or an environment of life, in which a series of primitive human beings are living in an artificial world with prehistoric creatures and objects. Due to its flexible structure and complex learning algorithms, we believe that Stone-age is a suitable tool for cognitive simulations in studying the emergence of complex human-like behaviors. This study is mainly focused on approaches and algorithms that can enhance the knowledge level of the agents and improve their decision-making process.

**Keywords:** Fuzzy Logic, Artificial Life (A-Life), Evolutionary Reinforcement Learning, Stone-Age Ecosystem, Intelligent Agent, Multi-Agent Environment.

## 1. Introduction

Artificial life (also known as "A-Life") is an interdisciplinary field of research that is aimed at understanding the living systems by observing the dynamic principles in their process and evolution, and artificially synthesizing them on another physical medium, such as a computer. Von Neumann designed the first artificial-life model (without referring to it as such) when he created his famous self-reproducing, computation-universal cellular automata [31]. At about the same time, Wiener started applying information theory and the analysis of self-regulatory processes (homeostasis) to the study of living systems [30].

A-Life focuses specifically on those complex systems that involve life, and these typically involve adaptation and learning. Though A-Life differs from artificial intelligence, the two are connected through A-Life's deep roots in computer science, especially artificial intelligence (AI) and machine learning. At the moment, A-Life uses three different kinds of synthetic methods. "Soft" A-Life creates computer simulations or other purely digital constructions that exhibit life-like behavior. "Hard" A-Life produces hardware implementations of life-like systems.

A-Life's main challenges come into three broad categories concerning life's origin, its evolutionary potential, and its connection to mind and culture [29]. The branch of A-Life that is discussed in this paper is mainly concerned with the ecosystem simulation. Ecosystem simulation is related to the mutual behaviors and reactions shown by the creatures with complex comportments. Such creatures are

usually capable of eating, fighting, reproducing and communicating with the others. Moreover, the environment where these creatures live in is a high level environment, making it possible for them to perform all the functions defined for their behaviors. LEE (Latent Energy Environments) [1], ERL (Evolutionary Reinforcement Learning) [2, 3], Echo [4] and Zamin [5] are examples of such environments.

Besides the above models for Ecosystem simulation, there exist some related works in the literature. Gras et al. [6] presented an individual-based predator-prey model with each agent behavior being modeled by a fuzzy cognitive map (FCM), allowing the evolution of the agent behavior through the epochs of the simulation. The FCM enables the agent to evaluate its environment (e.g., distance to predator or prey, distance to potential breeding partner, distance to food, energy level) and its internal states (e.g., fear, hunger, curiosity), and to choose several possible actions such as evasion, eating, or breeding.

Devaurs et al. [7] have developed an individual-based evolving predator–prey ecosystem simulation that exploits fuzzy cognitive map for modeling individual behaviour along with an evolutionary growth mechanism. The simulation constitutes an adaptive system involving a behavioural model with feedback effects and short term memory, a large number of interacting individuals, and a multi-level resource system. They simulated the species abundance patterns observed in the communities based on Fisher's logseries. Some preliminary results are provided about how their simulation is supporting ecological field results.

Dorin et al. in [8] investigated trends in A-Life that have led to a predominance of simulations incorporating artificial evolution acting on generic agents. By incorporating low-level models of the three frameworks of energetics, matter and evolution into virtual ecosystems they introduce pathways for ecologically relevant research in A-Life, particularly in domains where the interactions of the abiotic and biotic play a role.

Ouannes et. al. [9] proposed a virtual ecosystem environment with basic physical law and energy concept containing 3D virtual creatures foraging for the food. A genetic algorithm with an artificial neural network were implemented together to guarantee the correctness of actions. The creatures are said to be able to reach multiple food sources during the simulation time.

Golestani et al. [10] tried to analyze the results of an individual-based ecosystem simulation to evaluate its complexity. Four methods have been used for this analysis: Higuchi fractal dimension, correlation dimension, largest Lyapunov exponent and P&H method. They finally have found a deterministic and chaotic behavior in ecosystem simulation.

Benes et. al. [24] presented a biologically-based technique for interactive and efficient modeling of virtual plants and plant ecosystems. The virtual plants communicate actively with the environment and attempt to generate an optimal spatial distribution that dynamically adapts to neighboring plants, obstacles, light, and gravity.

The main objective of this study is to simulate an A-Life in an environment consisting of intelligent beings that behave in a human-like, logical way. To achieve this goal, we first propose a new artificial environment based on Zamin model, called Stone-age. Although the basis for Stone-age is Zamin, major modifications and improvements have been made in it to achieve a higher flexibility. Moreover, to enhance the level of intelligence in the agents, we have utilized the dynamic fuzzy Q-learning (DFQL) algorithms along with methods for tuning the membership functions and creating fuzzy rules. The importance and necessity of this research lies in the fact that most of the existing simulations in the literature include simple creatures, with short-term memories and low cognition of their surroundings and their own internal state. Moreover, the creatures in these simulations are mostly incapable of processing their memories. To address the mentioned issues, this study is aimed at increasing the level of intelligence in these beings by using more effective learning algorithms.

The main contributions of this paper are: (a) application of the combinatorial algorithms based on fuzzy logic and evolutionary reinforcement learning in the simulation of primitive human behavior (b) integration of new parameters such as mental pressure, anger, sexual desire, self-confidence, etc., to enhance the intelligence levels and increase human-like behaviors.

The rest of this paper is organized as follows. In section 2, we first provide a brief introduction about the previous platforms for A-Life simulations. Then we describe the Zamin model which is the basis for the proposed model. defql Finally, Section 5 concludes the paper.

# 2. The Zamin Model

As mentioned before, several platforms for A-Life simulations have been presented [5]. However, none of them are fast and flexible enough for all aspects of cognitive study in A-Life. Most of these models mainly differ in their flexibility, type of simulation, and the tools they have used to provide their beings. Echo and LEE, are relatively simple models that use genetic algorithm (GA) for the evolution and generation of the new populations of beings [1, 4, 25]. The GA fitness function in such models is defined as a function of the amount of energy of the beings. Furthermore, each intelligent being is equipped with a trainable neural network (NN) for decision-making processes [11]. The structure of ERL environment is also similar to previous models; except for that it consists of a wider range of beings. Moreover, the brain of each agent is composed of two feed-forward NNs, whose weights are preliminarily amounted by means of their parents' genes [3, 4, 22]. The first NN is responsible for selecting the best possible function, whereas the second NN indicates the fitness value of the current agent. The output of the second NN is used by the first NN for selecting the next function.

In order to provide a realistic simulation of the real world, Halavati and Bagheri introduced Zaminmodel [5]. Zamin uses ERL as its basis and further improves it. Zamin employs internal sensors for each intelligent being which makes them able to understand their current situation and thus improve their decision-making. In another work, the authors further extended the basic model by replacing the simple decision making rules with fuzzy rules and proposed Zamin II [28]. Their results indicate a considerable improvement over the basic approach. The pleasure-based learning system in Zamin II enables the intelligent beings to store decision-making

rules related to pleasurable actions as new experiences in their memory.

In general, learning methods in an agent with fuzzy control have been focused on three points:

Tuning the membership functions: The approaches presented in this group enable the intelligent agent to tune its membership functions toward a better decision-making in the learning process.

Generating and tuning the IF-THEN rules in the fuzzy rules database: The algorithms studied in this domain have so far dealt with approaches for learning the structure of the database, qualitative and quantitative tuning of the rules and also algorithms to generate rules, dynamically.

Learning and the tuning of the fuzzy inference system: The relevant algorithms that have been studied so far have generally been combinations of reinforcement learning algorithms and function approximations serving to tune the fuzzy database.

One of the most prominent algorithms in the literature is SARSA [12, 13], which is regarded as an on-policy temporal difference algorithm. In SARSA, the agents use the given policy in order to decide about the next function selection. Policy-dependent algorithms usually consider the previous experiences into account when making decisions (Exploitation). Thus, they enjoy more safety; however, they lose the opportunity of encountering potentially better situations beyond their policy [13]. Another widely used algorithm is Q-Learning [15, 19, 20, 21]. In contrast to SARSA, this algorithm is off-policy, i.e. the optimum policy is learned by the agent regardless of the action taken. The Q-Learning algorithm may behave beyond its given policy at times, thus discovering potentially better policies (Exploration). In many studies, tracing merit functions have been used in order to enhance SARSA and Q-learning algorithms. To explain how the tracing merit function works, if a certain state is observed in a time step, the amount of the function is added by one unit for that state; for the next time steps, however, the amount of the function is reduced by one until the concerned state is encountered again. In other words, such an improvement turns one-step time differences ($TD_0$) into multi-step time differences ($TD_\lambda$).

All the studies mentioned above have a major feature in common, that is, they are based on tables. In other words, all of these methods use a search table to maintain the previous values of their value functions [12, 14]. In problems with a large number of sensors and continuous variables, the agent constantly encounters with previously unseen situations. In such problems, the only way for learning is to generalize the previously observed situations to unobserved situations. Therefore, in many previous studies, a generalization of SARSA and Q-learning algorithms has been used based on the function approximation via gradient descent [16].

With the progress of Q-learning in fuzzy inference systems, Fuzzy Q-Learning (FQL) algorithms have been used in numerous fields. FQL is a reinforcement approach for learning the rules in fuzzy inference [15]. In this approach, variables used in the simple Q-learning algorithm are replaced with fuzzy sets. The important point of a fuzzy inference system is that its rule-base does not require using a search table. Each fuzzy rule has an amount Q proportionate to its value related to a specific function in a specific situation, the fuzzy inference system rules will thus be improved by means of Q-learning algorithms and updating the amounts for Q. In this model [16-18], the fuzzy inference system will consist of rules as follows:

$$R^i: \text{if}\langle S^i \text{and action} = a^i\rangle \text{then}\langle q\_value = q(S^i, a^i)\rangle \quad (1)$$

in which $\langle S^i\rangle$ means $\langle x_1 \text{is} s_1^i \text{and} x_2 \text{is} s_2^i \text{and} \dots \text{and} x_n \text{is} s_n^i\rangle$, $s_k^i$ denotes the fuzzy language expressions and $x_k$ denotes input variables (states).

In many FQL approaches, the actions are selected using an Exploration/ Exploitation Policy (EEP). Different approaches used different policies to maximize the amount of knowledge gained by learning system. The Boltzmann exploration and pseudo-stochastic methods [16] as well as ε-greedy are examples of those policies [12]. In some FQL systems where usual EEPs aren't suitable enough, evolutionary methods, i.e. Genetic Algorithms, are used to improve the decision making process [16].

The problem we may encounter in FQL methods is when we come across the situations that are unknown to the system. It means that if the system inputs are incompatible with the rules in the knowledge-base, the decision-making, and consequently the agent's learning – process will be disrupted. One solution to this problem is to design a system which is able to dynamically generate and tune the rules according to new situations. This was the solution that guided many studies toward dynamic fuzzy Q-learning. Apart from tuning the fuzzy inference system parameters based on previous approaches, DFQL is also capable of tuning its own inference system structure in a self-learning, dynamic manner [20].

As mentioned before, we have selected the Zamin model as the basic for implementation of our proposed Stone-age. Therefore, we first describe the Zamin and its characteristics in detail.

## 2.1. Description of Zamin

Zamin virtual world is a fast and extensible medium based on ERL which has many advantages compared to the previous models. Zamin has a checkerboard-like structure which consists of three different species: (a) intelligent agents which are called Aryos. These agents are capable of learning and are the basic species in the project. (b) Sentinels that act the same as hunters in ERL. (c) Plants which grow randomly in different parts of the earth and agents feed on them.

**a) Life Cycle of Agents in Zamin World**

Each agent has an energy index, which either degrades over time or increases by feeding on plants or animals' corpses. If the energy level drops below a threshold it may lead to agent's death. Therefore, the agents aim to increase their energy in order to increase their lifetime.

**b) Agents' Evolution**

Most features of agents in Zamin are not stable. Thus they are defined in a data structure format called genetic code. After reproduction, the parent's genetic code will be mutated with a certain probability and will be transmitted to its offspring.

**c) Feasible Actions**

Aryos are allowed to select one of the following feasible actions:

- Shifting to one of the three cells ahead
- Changing the direction
- Eating
- Reproducing
- Attacking a creature in the opposite cell
- Standing steady

Moreover, agents in Zamin have a set of internal features such as fatigue, food taste and physical strength. Maximum level of each one of these features is stored in a genetic code format and will be transferred to the offspring in case of reproduction. For example, the agents with higher maximum level of fatigue usually have offspring without fatigue.

**d) Agents' Sensors**

Every Aryos have two kinds of sensors. The first sensor, called internal sensor, enables the agent to be aware of its inner state at any moment. Inner state of agents involves energy levels, fatigue, food taste and physical strength. Internal sensor also receives information like the direction of movement, last performed action, and agent's age. The second sensor is external sensor which receives data related to the nearest object located within the agent's sight.

**e) The Pleasure-Based System in Agents**

Every Aryos uses a system based on its own pleasure to conduct the agent's unsupervised learning system. This system shows the satisfaction degree of each agent, i.e. the degree that an agent is satisfied with its own current state at any moment.

**f) Decision-Making System**

Decision-making in each agent is performed based on different rules. Each principle is in (S, A, P) or (State, Action, Pleasure) format which represents the degree of pleasure in each agent as a numerical value P, in experienced state S and performed action A. During the decision-making process, the current state S is compared with all rules, providing a compliance level of P which is the criterion for action selection.

**g) Agents' Learning**

After selecting the next action in decision-making process, the amount of pleasure received from the sensors will be incorporated with premise (P part) of chosen decision-making rule. Thereby, the system will save the consequence of its action, which helps it to make better decisions in the similar situations in future.

**h) Sentinels' Performance**

The performance of the sentinels differs from that of the agents only in two points. First, decision-making rules in sentinels are fixed and they have no learning or genetic code. Second, their aim is only to search and hunt agents, thereby increasing their energy.

## 2.2. Shortcomings of the Zamin Model

The Zamin model has the following disadvantages:

(1) Learning process in Zamin is solely based on the experiences acquired by the agents. In other words, the experience of the agent is represented in the form of an if-then fuzzy rule and will be saved in the agent's memory. But, the agents are unable to perform any process with their own knowledge or experiences. Therefore, producing new rules in their own knowledge database would only be done in case of acquiring a new experience by agents themselves.

(2) The reinforcement learning method in Zamin is very simple and it is only based on the pain or pleasure level.

(3) Zamin's creatures have relatively primitive level of understanding and reasoning, and they suffer from lack of many human traits like curiosity, self-confidence and sexual desire. Therefore, these creatures are unable to react like human in numerous situations. They also have little flexibility in their behaviors because of their single-objective goal, i.e. to get more and more pleasure.

(4) The discrete structure of Zamin model restrains the movement of the creatures to the checkerboard-like lattice structure.

# 3. Stone-Age Model

Stone-age model consists of a checkerboard-like structure in which the creatures and objects are located in different cells. These creatures include mature humans (men and women), children, domestic and wild animals, plants, water, shelter and etc. Although the stone-age model is composed of a limited number of cells, humans and creatures are able to move in all possible directions. Moreover, they can be placed anywhere in the model. In other words, the movement of creatures is completely continual and is not restricted to center of the cells. In Stone-age world, time is considered as an environmental parameter. Thus, darkness or daylight affects the decision-making of the agents.

In fact, intelligent agents in Stone-age are similar to primitive humans. These creatures' minds are based on fuzzy control and they attempt constantly to survive.

In order to implement this program, XNA Game Studio 4 Net was used. This model is capable of creating image type objects and making animation from images. Each one of defined objects has its own specific image and animation in the program, which are updated at specific intervals. An instance of a board created by this program in XNA is shown in figure 1.


Figure 1. Stone-age GUI in XNA game studio

In order to make creatures animated in this model, an image file containing all possible motions is used. The number of video frames existing in the image depends on the

extension of object's movement. An instance of this image is shown in figure 2.

Figure 2. All necessary frames for animating a human being object

Stone-age program includes three modules that are GUI, logic, and agent. Figure 3 shows the relation between these modules.

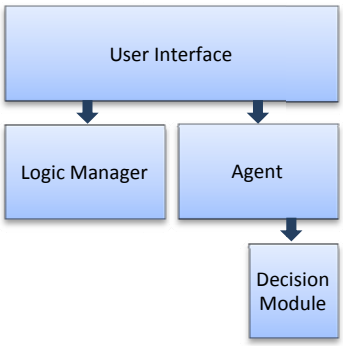


Figure 3. Main modules of stone-age program

The user interface module includes all classes corresponding to graphical objects. It also has an instance of logic manager module as well as several instances of agent class. Logic manager module has some classes for management of environment's logic. The main class of user interface uses this module for all necessary graphical updates in a logical framework. Agent class represents an intelligent agent with fuzzy logic which receives the current state from user interface with the help of its sensors. It will apply final action on the model after performing all necessary processing. Pseudo code and the flowchart of this action are as shown in figures 4 and 5.

The Logic Manager. Get State (map) function determines the current state in a data structure format according to the world environment. Agent. Observe (S) procedure receives the current state and registers visible information in agent's mind with respect to its extent of view. Then the best action would be selected by Agent. Get Action () through the current state processing. Afterwards, Logic Manager. Calculate Reward (map, A) calculates the reward of this action and Perform Action (A) applies the chosen action to the environment in a graphical way. Finally, Agent. Update Knowledge (S_prime, R) updates agent's database by DFQL algorithm and saves it.

### a) Agents' Life Cycle in Stone-Age World

Similar to Zamin, in Stone-age each agent has an energy index, which will either decrease over time or increase by feeding on plants or of animals' corpses. If the energy level drops below a threshold it may lead to agent's death. Therefore, the agents aim to increase their energy to further increase their lifetime. Moreover, agents in Stone-age are able to move in one of the four main directions including: North, South, East and West or four halfway directions including north-west, north-east, south-west and south-east.

```
User Interface: Update Method ()
{
Foreach Agent Do
    {
          State S = Logic Manager. Get State (map);
          Agent. Observe (S);
          Action A = Agent. Get Action ();
          Reward R = Logic Manager. Calculate Reward
         (map, A);
          map = Perform Action (A);
      State S_prime = Logic Manager. Get State (map);
      Agent. Update Knowledge (S_prime, R);
       }
}
```

Figure 4. The pseudo code for decision making in stone-age model

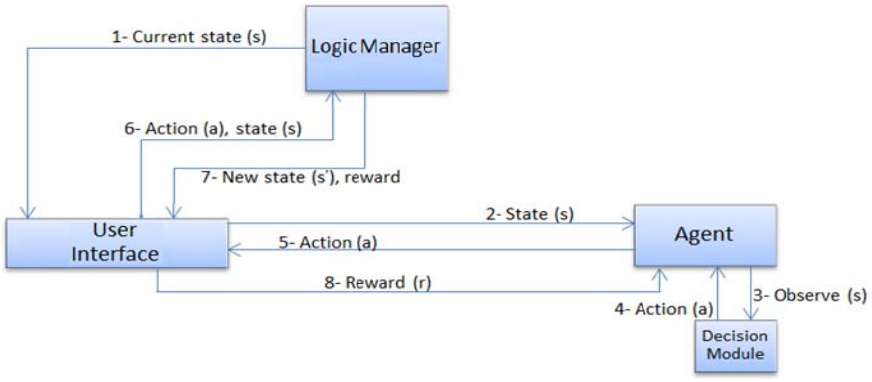


Figure 5. Flowchart of decision making process for agents in Stone-age

**b) Agents' Evolution**

For reproduction in Stone-age, male and female should have necessary requirements. For this reason, both agents should have sexual desire, high energy and they should also be located near to each other according to their direction of movement. After reproduction, a certain amount of energy along with genetic features will be transferred from parent to their offspring. Most human features are not stable and would be defined in a structure format of genetic code as Zamin. After reproduction, three tasks are performed: (1) the parents' genetic codes will be combined, (2) a mutation with a certain probability will be performed on the resultant genetic code, and (3) the obtained genetic code will be transferred to the offspring. Agents' genetic code is stable during their lifetime and can only be mutated at their birth time.

**c) Feasible Actions**

Different types of feasible actions that can be performed by an agent are available in Stone-age. All of them are stored in a database. Table 1 defines all possible actions that could be done by a primitive human being. Agents have a range of sight which will be changed by their genetic code. This range of sight for each agent includes cells of Stone-age in which the agent can see the objects and organisms surrounding it.

Table 1. Feasible actions which could be done in Stone-age

| Action name |
|---|
| Standing |
| Walking in each one of eight directions |
| Running in each one of eight directions |
| Feeding on plants and animals |
| Attacking enemy |
| reproduction |
| sleeping |

Agents in Stone-age have some internal features like energy, fatigue and physical strength (Table 2). A basic level of some of these features is stored in a gene format in agents' genetic code and will be transferred to the offspring, in case of reproduction.

**d) Agents' Sensors**

Each agent has two kinds of sensors. The first sensor, which is called internal sensor, enables the agent to be aware of its inner state at any moment. Table 2 lists the inner states for an agent. The second sensor is external sensor which receives data about the nearest object located within the agent's sight. For example, if this object is a plant, then the received information is the distance of plant to the agent and its energy level. But if other agent was seen in this area, the received information will be the visible part of agent's inner state.

**e) Decision-Making System**

The decision-making process in each agent is done based on fuzzy rules. The format of each rule is as follows:

$$R_i: \text{if}\langle S^i\rangle \text{then}\langle a[i,1] \text{with} q[i,1]\rangle \text{ or}\langle a[i,2] \text{with} q[i,2]\rangle \text{or or}\langle a[i,J] \text{with} q[i,J]\rangle \quad (2)$$

where $\langle S^i\rangle$ denotes:
$\langle x_1 \text{ is} s_1^i \text{ and } x_2 \text{ is} s_2^i \text{ and } \ldots \text{and} x_n \text{ is } s_n^i\rangle$. $s_k^i$ is a fuzzy set and $x_k$ is an input variable.

Each agent has several predefined rules. The number of the rules can be increased by learning process. Following is an example of a fuzzy rule used in this system.

If $<S_1^1$: Wolf is near> and $<S_2^1$: Weather is dark> and $<S_3^1$: Fire is near>
Then $<a_1^1$: Run> with $<q_1^1$: 0.4> or $<a_2^1$: Sit-down> with $<q_2^1$: 0.6> or $<a_3^1$: Sleep> with $<q_3^1$: 0.2> (3)

Table 2. Agents' features in stone-age

| **Feature** | **Feature function** | **Genetic code** | **Genetic code function** |
|---|---|---|---|
| Energy | Will increase in case of feeding and will decrease in case of daily activities. | Max Energy | Will increase/ decrease every time energy level exceeds a threshold/ drops below a threshold. |
| Health | Will increase in case of sleeping and will decrease in case of daily activities, attacking, escaping and etc. | Max Health | Will increase/ decrease every time health level exceeds a threshold/ drops below a threshold. |
| Stress | Will increase in case of facing high risk situations and will decrease in case of sleeping and reproduction | Base Stress | Will increase/ decrease every time stress level exceeds a threshold/ drops below a threshold. |
| Anger | Will increase in case of seeing same sex in specific circumstances and will decrease in case of successful attacking or sleeping | Base Anger | Will increase/ decrease every time anger level exceeds a threshold/ drops below a threshold. |
| Strength | Will increase in case of feeding or sleeping or attacking and will decrease in case of being hungry or sleepy | Base Strength | Will increase/ decrease every time strength level exceeds a threshold/ drops below a threshold. |
| Sexual desire | Will increase in case of seeing opposite sex in a fixed rate and will decrease in case of reproduction | - | - |
| Consciousness | Will decrease in case of sleeping and will increase in case of doing various actions | - | - |
| Age | - | Max Age | Will calculate as a function of other genetic codes at the time of birth. |

where $S_i^1$ are fuzzy sets, $a_i^1$ are actions and $q_i^1$ are the corresponding q-values. Table 3 shows examples of initial rules in each agent's database.

Figure 6 depicts examples of defined membership functions in the fuzzy controller.

### f) Static Learning

Regarding Eq. (2), the inferred action a(x) for input vector $x = (x_1, x_2, \ldots, x_n)$, is calculated based on rule fulfillment $\beta^i(x)$ and corresponding action $a^i$ as below:

Table 3. Examples of initial rules for each agent in stone-age

| Condition1 | Condition2 | Condition3 | Conclusion |
|---|---|---|---|
| Food Distance Is Near | Agent Energy Is Critical | - | Approach Food |
| Food Distance Is Far | Agent Energy Is Critical | - | Approach Food |
| Food Distance Is Near | Agent Energy Is Low | Food Attractiveness Is High | Approach Food |
| Food Distance Is Near | Predatory Distance Is Far | Food Attractiveness Is High | Approach Food |
| Food Distance Is Close | Agent Energy Is Critical | - | Eat Food |

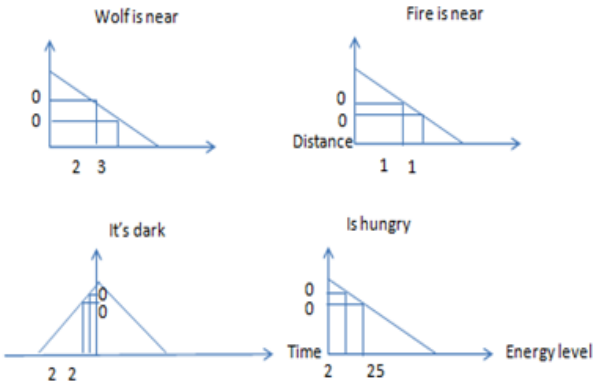


Figure 6. Some examples of agents' membership functions in stone-age

$$a(x) = \frac{\sum_{i=1}^{M} \beta^i(x) * a^i}{\sum_{i=1}^{M} \beta^i(x)} \quad (4)$$

The corresponding Q-values are also calculated as below:

$$Q(x, a) = \frac{\sum_{i=1}^{M} \beta^i(x) * q(S^i, a^i)}{\sum_{i=1}^{M} \beta^i(x)} \quad (5)$$

The fuzzy inference system consists of rules as described in Eq. (2). Since the learning system can choose one out of J actions for each rule, we call $a[i,j]$ the $j_{th}$ possible action in rule i and $q[i,j]$ its corresponding Q-value.

In order to explore a set of possible actions and acquire experience through reinforcement, the actions are selected using EEP. To maximize the amount of knowledge gained, we used a modified version of ε-greedy method in which the possibility of exploration or exploitation depends on the confidence level of agents, i.e. higher confidence level leads to a higher chance of exploration by trying new actions while the low confidence usually results in choosing actions with the highest Q-value.

Let $a[i, j^+]$ be the selected action in rule i using an EEP and the corresponding Q-value $q[i, j^+]$. Also consider $i^*$ such that $q[i, j^*] \leq \max_{1 \leq j \leq J} q[i, j]$. Then the actual Q-value of the inferred action a, is as follows:

$$Q(x, a) = \frac{\sum_{i=1}^{M} \beta^i(x) * q[i, j^+]}{\sum_{i=1}^{M} \beta^i(x)} \quad (6)$$

And the value of state x:

$$V(x) = \frac{\sum_{i=1}^{M} \beta^i(x) * q[i, j^*]}{\sum_{i=1}^{M} \beta^i(x)} \quad (7)$$

After selecting the next action in decision-making process, pleasure level received by sensors will be given to fuzzy controller as the reward level. In this stage, fuzzy inference system will update values of value function using TD (λ) methods. The difference between the "old" and the "new" $Q(x, a)$ is considered as an error signal $\delta = r + \gamma V(y) - Q(x, a)$ that can be used to update the Q-Values:

$$\Delta q[i, j^+] = \alpha \, \delta \, e[i, j] \quad (8)$$

where $\alpha$ is a learning rate and $e[i, j]$ is the eligibility trace computed as below:

$$e[i, j] = \begin{cases} \lambda \gamma e[i, j] + \frac{\beta^i(x)}{\sum_{i=1}^{M} \beta^i(x)} \text{if} j = j^+ \\ \lambda \gamma e[i, j] \text{otherwise} \end{cases} \quad (9)$$

In the above equation $\beta^i(x)$ is the firing strength of $i^{th}$ rule. The pseudo-code of the fuzzy Q-learning algorithm is given in figure 7.

```
Observe the state x
Repeat For each Rule:
    Choose action a(x) using some EEP
    Compute a(x) and corresponding q-value Q(x, a)
    Apply the action a(x), observe the new state y
    Receive the reinforcement (r)
    Update q-values using Δq[i, j+] = α δ e[i, j]
End repeat.
```

Figure 7. Stone-age fuzzy q-learning algorithm

### g) Dynamic Learning

In order to increase the effectiveness of the learning system in facing new situations, the agents are equipped with the capability of processing their experiences. As the result, the agents are not only able to improve the parameters of their fuzzy inference system, but also able to improve the structure of their inference system in a self-regulatory and dynamic manner [17, 23]. This optimization process includes modifying the existing rules or adding some new rules.

ϵ-Completeness criterion [23] is a measure that determines the necessity of creating a new rule in case of facing a new input variable X. For any input in the operating

range, if there are not any fuzzy rules with matching degree (or firing strength) less than $\epsilon$, more fuzzy rules should be employed to accomplish the input space. First, Mahalanobis distance between the current observation state X and centers of the membership functions for an existing fuzzy rule $R_j$, i.e. $C_j = \{c_{1j}, c_{2j}, \dots, c_{kj}\}$, is calculated according to:

$$f(R_j) = {}^{1}/_{md(j)} \tag{10}$$

In the above equation, $f(R_j)$ is the firing strength of $j^{th}$ fuzzy rule and md(j) is the Mahalanobis distance between input vector $\bar{X} = [x_1, x_2, \dots, x_n]^T$ and centers vector $\bar{C}_j = [c_{1j}, c_{2j}, \dots, c_{kj}]^T$ where k is the number of membership functions in $R_j$.

First we center $\bar{X}$ and $\bar{C}_j$ on the arithmetic mean of each variable. Let $\hat{X}$ and $\hat{C}_j$ be the centered vectors, then the covariance matrix of each vector is calculated as:

$$Cov_{X_j} = \frac{1}{n}\hat{X}^T\hat{X} \quad \text{and} \quad Cov_{C_j} = \frac{1}{k}\hat{C}_j^T\hat{C}_j \tag{11}$$

The pooled covariance matrix of the two vectors is computed as the weighted average of their covariance matrices:

$$Cov_j = \frac{n}{n+k}Cov_{X_j} + \frac{k}{n+k}Cov_{C_j} \tag{12}$$

Finally the Mahalanobis distance md(j) is simply calculated as the quadratic multiplication of mean difference of $\bar{X}$ and vector $\bar{C}_j$ and inverse of pooled covariance matrix:

$$md(j) = \sqrt{(M_X - M_{C_j})^T Cov_j^{-1}(M_X - M_{C_j})} \tag{13}$$

where $M_X$ and $M_{Cj}$ are arithmetic means of $\bar{X}$ and $\bar{C}_j$, respectively, and $Cov_j$ is the pooled covariance matrix of the two vectors. If md(j) becomes greater than a threshold $k = {}^{1}/_{\epsilon}$ or, $f(R_j)$ becomes smaller than ${}^{1}/_{k} = \epsilon$, creating a new rule is required in inference system.

But it should be noted that $\epsilon$-Completeness criterion itself is not an adequate condition to determine whether to create a new rule or not. In spaces in which the efficiency of DFQL estimation is lower than the expected, it is required to create new rules. For this reason, another index TD error criterion is used which is shown by $\xi$ here. In fact, the value of this index represents the time difference error of Q-Learning algorithm which is calculated as follows [22]:

$$\hat{\xi}_{t+1} = r_{t+1} + \gamma V_t(s_{t+1}) - Q_t(s_t, a_t) \tag{14}$$

where r is reward and $\gamma$ is the learning rate. The method for updating this index in each time step is as follows:

$$\xi_{t+1}^{i} = [(H - \alpha_t^i)\xi_t^i + \alpha_t^i(\hat{\xi}_{t+1})^2]/H\,; \quad H > 0 \tag{15}$$

where $\alpha_t^i$ is the firing strength of $i^{th}$ rule in time t which represents the level that $R_i$ rule affects TD error. Constant H is the learning rate for $\xi$ and usually takes a value from 10 to 100 [22]. Finally, if the value of $\xi^J$ is greater than a threshold like $k_\xi$, it means that rule $R_J$ will satisfy the TD error condition and therefore, it is necessary to create a new rule.

When system needs to create or change fuzzy rules, the received reward should be calculated in a specific interval. In addition, necessary changes should be made in agents' knowledge database. These changes are performed in two steps which we will describe consequently.

1) Adjusting Membership Functions

All membership functions will be updated at the end of the time intervals. This operation can be further explained with the following example:

Assume that the membership function of Food is very close is set to be updated. Also, assume that a rule was defined as follows:

IF food is very close THEN the reward is high

Now, if the average firing strength for this rule in all favorable states (states in which reward of the performed action is greater than a threshold, experimentally between 0.6 and 0.8) would be 0.85 and the average firing strength in all problematic states (states in which reward of performed action is lower than a threshold, usually between 0.2 and 0.4) would be 0.56, then it is concluded that more compliance with this rule will result in higher reward from environment. Therefore, adjusting the membership function will be in the direction of increasing membership degree for food is very close (Figure 8).

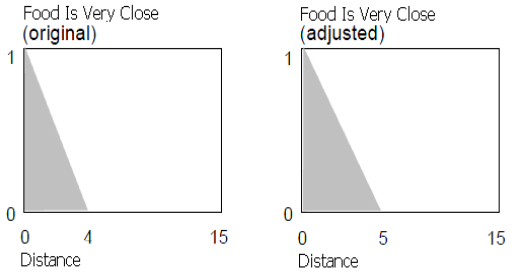


Figure 8. Adjusting membership functions in Stone-age

2) Generating New Rules Using a Combination of Existing Rules

In order to generate new rules, we need to introduce a communication state matrix. This matrix is used to identify the communication level of two linguistic variables. The variables with a communication level greater than a predefined threshold would be selected as candidates to create new rules. The method is explained with the following example:

Suppose we want to calculate the communication level of two linguistic variables Fire is Close (FIC) and Predatory is Near (PIN). Also, assume that five favorable states (states in which reward of performed action is greater than a threshold) of $S_1$ to $S_5$ were saved at the end of a specific time interval. Values of membership functions corresponding to these two linguistic variables are listed in table 4.

As can be seen in table 4, FIC has a membership value of 0.9 in $S_3$ and it is the only value which is greater than the average (it is in minority compared to other four states), therefore FIC is counted as a special case in $S_3$.

On the other hand, PIN has a membership value lower than the average value in $S_3$ and $S_5$ (it is in minority compared to other three states), and is considered as special

cases in these two states. In one of these states, i.e. $S_3$, FIC is special one too; therefore communication level of these two linguistic variables is ½ or 0.5.

Table 4. Communication state matrix for two linguistic variables FIC and PIN

| | S1 | S2 | S3 | S4 | S5 | Avg. |
|---|---|---|---|---|---|---|
| FIC | 0.5 | 0.6 | 0.9 | 0.5 | 0.6 | 0.62 |
| PIN | 1 | 0.8 | 0.6 | 0.9 | 0.7 | 0.80 |

Finally, if this communication level becomes greater than a threshold level, rules like those in the following will be created for all performed actions.

If <Predatory is near> and <Fire is close>
Then <Run> with <$q_0$> or <Sit-down> with <$q_0$> (16)

where $q_0$ is the initial q value. To validate the newly created rule, a testing process needs to be performed to remove the potentially invalid actions. For this purpose, if an agent encounters a state where the newly created rule is fired, actions which cannot be performed will be removed from the rule. Clearly, if all actions of a rule have been removed, the rule is nullified.

**h) Hybrid Learning Algorithm**

Static learning enables agent to update q values in order to acquire experience through reinforcement. From the other hand, dynamic learning allows agent to became more flexible in facing new situations. Therefore, in order to increase the effectiveness of the learning system both types of methods are necessary. The learning algorithm in Stone-age is a combination of both static and dynamic learning methods discussed above. As explained before, the necessity of creating new rules is determined by measuring ϵ-Completeness and TD-error criteria. In another word, creation of new rules depends on the satisfaction levels of these criteria.

1) Satisfaction of ϵ-Completeness Criterion

Assume that at the end of a specific period, an agent encounters a new state that is not satisfying the ϵ-Completeness criterion. Thus creating a new rule is required. In this case, candidates for creating a communication state matrix would be the linguistic variables of the newly encountered state and every other existing linguistic variable. Therefore, new rules could be generated due to the communication levels of the linguistic variables of the new state.

2) Satisfaction of TD-Error Criterion

Again, assume that at the end of a specific period, there are one or more rules that cannot satisfy the TD-error criterion. In this case candidates for creating a communication state matrix would be the linguistic variables of those rules and every other existing linguistic variable. Therefore, if the communication levels become greater than a threshold, new rules with an extent of similarity to the original rules could be produced. Figure 9 illustrates the flowchart of the proposed hybrid learning algorithm.

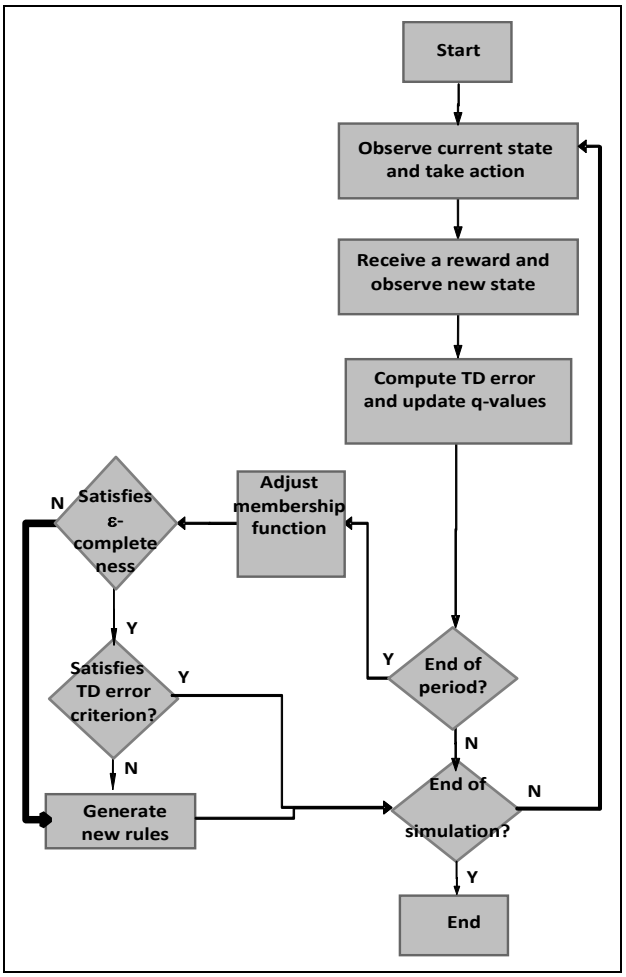


Figure 9. The flowchart of the hybrid learning algorithm

**i) Sentinels' Performance**

Same as in Zamin, sentinels in Stone-age have the same performances as agents' one with two major differences. Firstly, decision-making rules and secondly, their aim to search and hunt agents.

# 4. Experimental Study

In order to investigate the Stone-age, the proposed model was implemented using C# (Dot-Net framework 3.5). The initial state of the Stone-age environment, which was used for training, is a two dimensional 40×40 checkerboard-like lattice structure area as shown in figure 10. Various objects like trees, fire, caves and cliffs have been placed randomly over the area. To begin the training phase, a few agents (humans) from both genders are located at random cells. A couple of predators also placed on the map. Predators cannot reproduce but after they die, there is a chance that new predators will be created again. Time-dependant feature of the environment causes the trees to fertilize again after being consumed as food resources. Caves can be used as safe shelters during sleep time, and fire keeps the wild animals away.

At the beginning of the training phase, the agents' inference systems are initialized with default fuzzy rules and membership functions. The information is stored for each agent in some data files, separately. The initialized rule-base is shown in table 5.

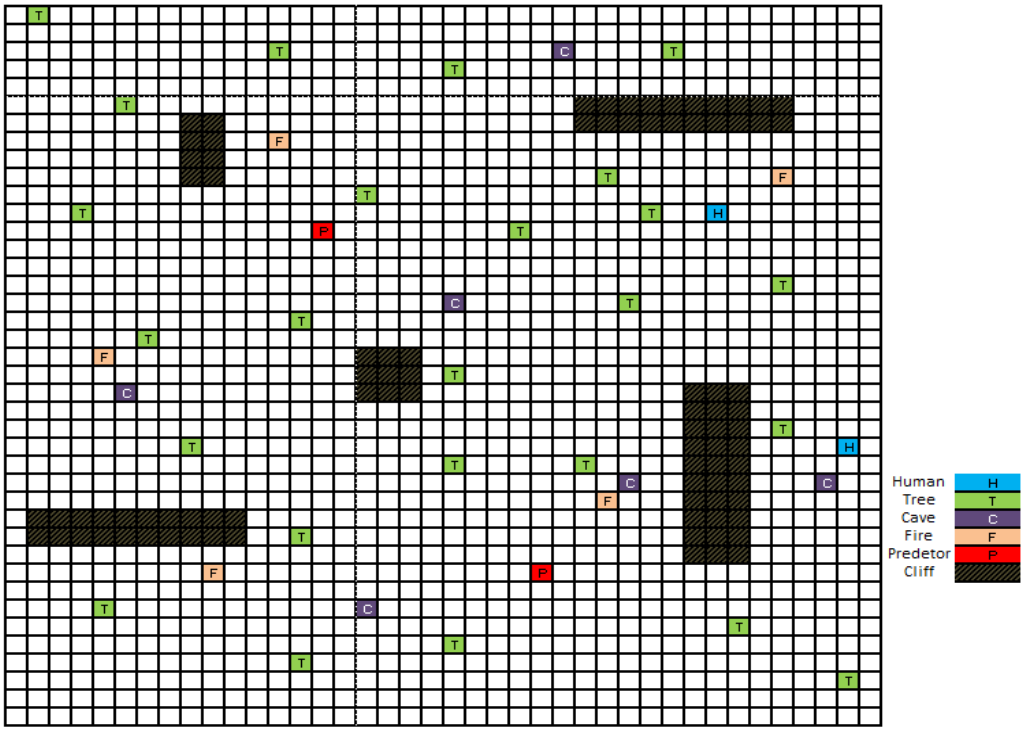


Figure 10. Stone-age environment

Table 5. Initialized rules of each agent in Stone-age

| | **Condition1** | **Condition2** | **Condition3** | **Conclusion** |
|---|---|---|---|---|
| 1 | Food Distance Is Near | Agent Energy Is Low | - | Approach Food |
| 2 | Food Distance Is Far | Agent Energy Is Critical | - | Approach Food |
| 3 | Food Distance Is Close | Agent Energy Is Critical | - | Eat Food |
| 4 | Food Distance Is Close | Agent Energy Is Low | - | Eat Food |
| 5 | Fire Distance Is Near | Predatory Distance Is Near | - | Approach Fire |
| 6 | Fire Distance Is Far | Predatory Distance Is Near | - | Move away Predatory |
| 7 | Fire Distance Is Close | Predatory Distance Is Near | - | Stand |
| 8 | Predatory Distance Is Close | Agent Strength Is High | - | Attack Predatory |
| 9 | Predatory Distance Is Close | Agent Health Is High | - | Attack Predatory |
| 10 | Environment Is Night time | Cave Is Near | - | Approach Cave |
| 11 | Environment Is Night time | Cave Is Close | - | Sleep |
| 12 | Agent Strength Is Low | Food Distance Is Near | - | Approach Food |
| 13 | Agent Energy Is High | Environment Is Day time | - | Explore |
| 14 | Agent Health Is High | Environment Is Day time | - | Explore |
| 15 | Agent Arousal Is high | Opposite Sex Is Near | - | Approach opposite Sex |
| 16 | Agent Arousal Is high | Opposite Sex Is Close | Age Is Medium | Reproduce |
| 17 | Agent Temper Is High | Same Sex Is Close | - | Attack Same Sex |
| 18 | Agent Strength Is Low | Predatory Distance Is Close | - | Move away Predatory |
| 19 | Predatory Distance Is Far | Environment Is Day time | - | Explore |
| 20 | Agent Health Is Low | Cave Is Near | - | Approach Cave |

To study how these agents function in such an environment, three different scenarios have been implemented. In the first scenario, the number of agents, predators and shelters (fires and caves) is balanced so the difficulties in the environment became an easy challenge for agents. In the second scenario, the number of predators has been increased. Third scenario contains even more predators and also the number of shelters (fires and caves) has been reduced greatly. In the following sections the performance and the behavioral results of the agents in each scenario is examined.

## 4.1. The First Scenario

In this scenario only a few number of threats exists in the environment. The initial number of objects is shown in table 6.

After running the application for 510 minutes, the results represented a major growth in the population while the resources stayed stable. Obtained results are shown in table 7 and the graph of population, total birth, and total death over time is illustrated in figure 11.

Results shown in above diagrams represent that total birth number consequently overtakes death and as a result, there is a relative success of agents in increasing their lifetime as well as survival of their generation.

Table 6. Initial objects of the first scenario

| Object Name | Initial Number |
|---|---|
| Human | 2 |
| Predator | 2 |
| Fire | 8 |
| Cave | 8 |
| Tree | 20 |

Table 7. Quantitative comparison between population and resources over time in the first scenario

| Time | Total birth | Total death | Population | Wild animals | Food sources |
|---|---|---|---|---|---|
| 0 | 2 | 0 | 2 | 2 | 20 |
| 5 | 2 | 0 | 2 | 2 | 20 |
| 10 | 2 | 0 | 2 | 2 | 20 |
| 20 | 3 | 0 | 3 | 2 | 19 |
| 30 | 3 | 0 | 3 | 1 | 21 |
| 60 | 6 | 0 | 6 | 1 | 22 |
| 90 | 8 | 3 | 5 | 1 | 24 |
| 120 | 10 | 2 | 8 | 2 | 23 |
| 150 | 14 | 7 | 7 | 2 | 26 |
| 180 | 17 | 7 | 10 | 2 | 29 |
| 210 | 22 | 14 | 8 | 2 | 28 |
| 240 | 26 | 15 | 11 | 3 | 28 |
| 270 | 33 | 23 | 10 | 2 | 26 |
| 300 | 40 | 24 | 16 | 3 | 24 |
| 330 | 49 | 34 | 15 | 2 | 21 |
| 360 | 62 | 36 | 26 | 2 | 20 |
| 390 | 79 | 49 | 30 | 1 | 18 |
| 420 | 101 | 52 | 49 | 2 | 20 |
| 450 | 128 | 83 | 45 | 1 | 17 |
| 480 | 161 | 86 | 75 | 1 | 18 |
| 510 | 202 | 131 | 71 | 1 | 16 |

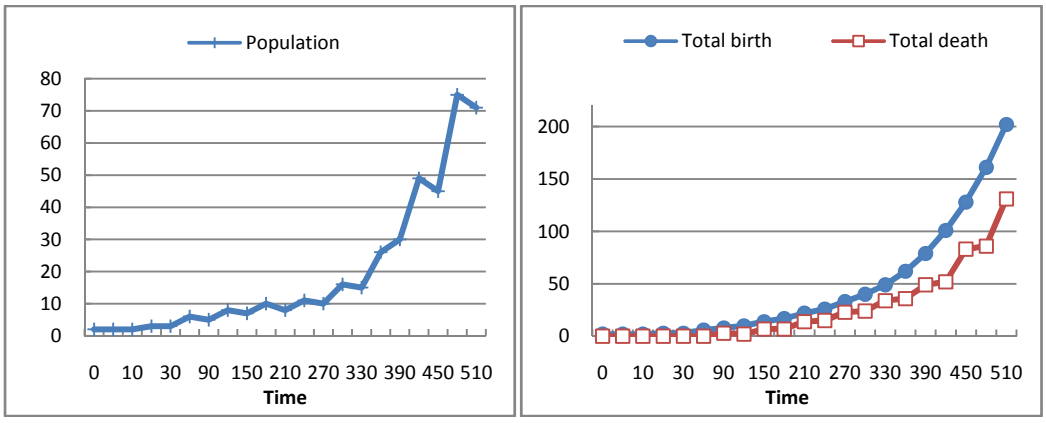


Figure 11. Graphical comparison between population, total birth and total death over time in the first scenario

Table 8. Comparison of average fitness of agents' genetic code over the time in the first scenario

| Time (minute) | Max health | Max energy | Base strength | Base anger | Base Stress | Max age |
|---|---|---|---|---|---|---|
| 0 | 50 | 50 | 50 | 50 | 50 | 50 |
| 5 | 50 | 50 | 50 | 50 | 50 | 50 |
| 10 | 50 | 50 | 50 | 50 | 50 | 50 |
| 20 | 50 | 50 | 50 | 50 | 50 | 50 |
| 30 | 50 | 50 | 50 | 50 | 50 | 50 |
| 60 | 50 | 50 | 50 | 50 | 50 | 50 |
| 90 | 49.7 | 50.2 | 50 | 49.9 | 49.7 | 50 |
| 120 | 49.7 | 50.2 | 50 | 49.7 | 49.3 | 50.3 |
| 150 | 49.9 | 50.5 | 50 | 49.2 | 49.5 | 50.9 |
| 180 | 49.9 | 50.5 | 50 | 49.4 | 48.6 | 51.7 |
| 210 | 50.3 | 50.8 | 50 | 48.9 | 48.2 | 52.4 |
| 240 | 50.3 | 50.8 | 50 | 48.6 | 47.7 | 54 |
| 270 | 50.5 | 51 | 50.4 | 48.2 | 48.1 | 54.8 |
| 300 | 50.7 | 51 | 50.4 | 48.4 | 48 | 55.6 |
| 330 | 50.9 | 51.6 | 50.7 | 47.9 | 47.6 | 55.7 |
| 360 | 50.9 | 51.7 | 50.7 | 47.7 | 47.4 | 57.7 |
| 390 | 51.2 | 51.1 | 51.2 | 47.5 | 47.1 | 58.2 |
| 420 | 51.4 | 50.7 | 50.9 | 47.1 | 47.3 | 58.9 |
| 450 | 51.8 | 50.9 | 51.5 | 47.3 | 47 | 58.6 |
| 480 | 52 | 51.1 | 51.5 | 46.8 | 46.8 | 59.9 |
| 510 | 52.3 | 52.5 | 52.3 | 47 | 46.6 | 61 |

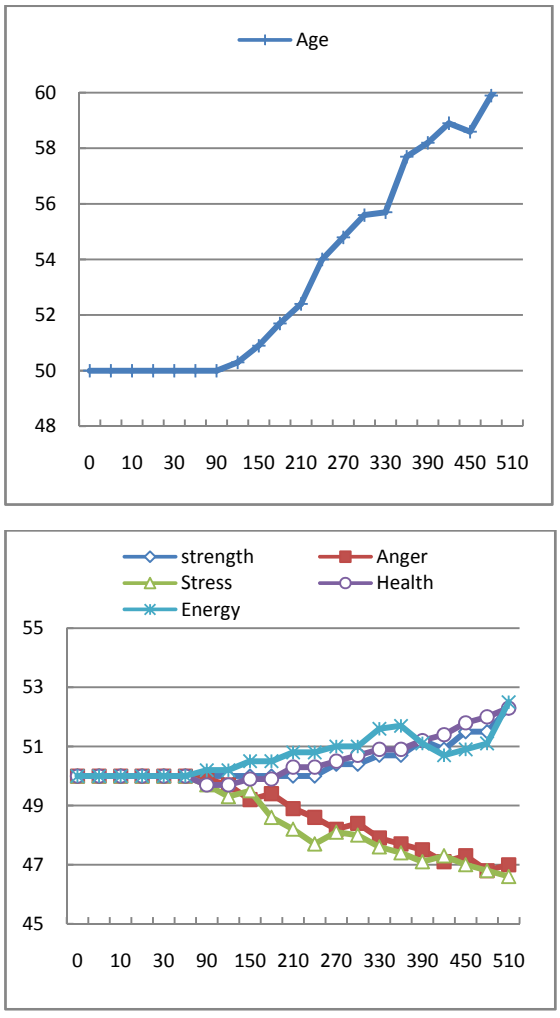


Figure 12. Graphical comparison of average fitness of agents' genetic code over the time in the first scenario

According to the genetic code results shown in above diagrams, there is also a relative success of agents in finding food resources and sleeping properly and as a result increasing their Health, Energy and Strength as well as reducing their base anger and stress. Also, there are some noticeable fluctuations in energy plot which can be explained by rule-base learning capability of the agents.

Each agent has some fixed and pre-defined fuzzy rules in database. Dynamic learning enables intelligent agents to update their fuzzy inference system in case of facing new occurrences. This update process involves modifying membership functions parameters or creating new rules. Table 9 lists the new learned rules that are common between most of the agents and passed through generations.

Table 9. Common learned rules by DFQL algorithm in the first scenario

| Row | Time (minute) | New rules |
|---|---|---|
| 1 | 90 | Agent Energy Is High & Agent Health Is High => Explore |
| 2 | 120 | Predatory Distance Is Near & Food Distance Is Near => Approach Food \| Move Away Predatory |
| 3 | 240 | Agent Energy Is Critical & Food Distance Is Far => Approach Food |
| 4 | 330 | Agent Energy Is High & Predatory Distance Is Near => Explore |

New learned rules indicate that in this scenario most of the agents are willing to explore. Exploration may lead to finding new food resources or mates to reproduce but also could reduce the energy level. Rules number 1 and 4 could explain the reduction of the energy level in figure 12 and rule number 3 could explain its increment afterward.

## 4.2. The Second Scenario

In this scenario the number of agents and predators are slightly increased but there are still a fairly large number of shelters. The initial number of objects is shown in table 10.

Table 10. Initial objects of the second scenario

| Object Name | Initial Number |
|---|---|
| Human | 4 |
| Predator | 8 |
| Fire | 10 |
| Cave | 10 |
| Tree | 20 |

After running the application for 510 minutes, again the results represented increasing in population and stability in resources. Obtained results are shown in table 11 and the graph of population, total birth and total death over the time is indicated in figure 13.

In comparison with the first scenario, although there are some minor fluctuations in the population curve and a slightly reduction of total population which is the result of dangerous environment, results shown in above diagrams represent the relative success of agents in increasing their lifetime in this scenario.

Evolution of the genetic code shown in above diagrams represents that despite the fluctuations of anger and stress curves and total increment of stress level, incremental progress of total age stayed somehow stable. It indicates that growth of strength level and final reduction of anger brings a general stability to lifetime of the agents.

Table 11. Quantitative comparison between population and resources over the time in the second scenario

| Time (minute) | Total birth | Total death | Population | Wild animals | Food sources |
|---|---|---|---|---|---|
| 0 | 4 | 0 | 4 | 8 | 20 |
| 5 | 4 | 0 | 4 | 8 | 20 |
| 10 | 4 | 0 | 4 | 8 | 19 |
| 20 | 5 | 0 | 5 | 8 | 20 |
| 30 | 5 | 1 | 4 | 7 | 20 |
| 60 | 6 | 1 | 5 | 7 | 22 |
| 90 | 9 | 3 | 6 | 7 | 20 |
| 120 | 11 | 3 | 8 | 7 | 26 |
| 150 | 14 | 8 | 6 | 7 | 24 |
| 180 | 18 | 8 | 10 | 6 | 22 |
| 210 | 24 | 14 | 10 | 7 | 22 |
| 240 | 28 | 18 | 10 | 7 | 24 |
| 270 | 34 | 25 | 9 | 6 | 27 |
| 300 | 39 | 25 | 14 | 5 | 26 |
| 330 | 48 | 35 | 13 | 4 | 23 |
| 360 | 57 | 36 | 21 | 5 | 20 |
| 390 | 70 | 49 | 21 | 6 | 21 |
| 420 | 84 | 49 | 35 | 8 | 20 |
| 450 | 101 | 68 | 33 | 7 | 17 |
| 480 | 132 | 69 | 63 | 8 | 21 |
| 510 | 173 | 102 | 71 | 6 | 19 |

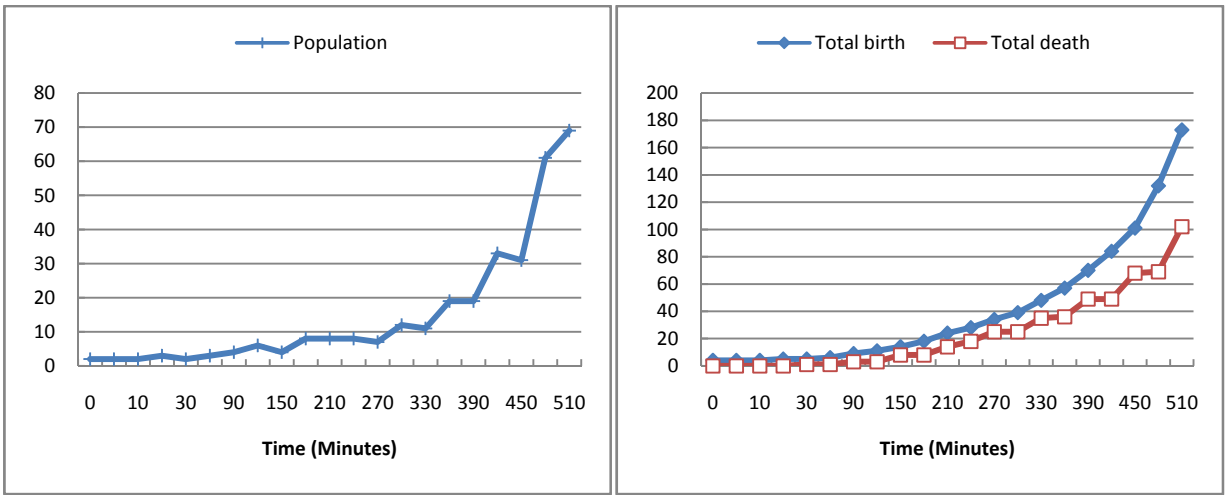


Figure 13. Graphical comparison between population, total birth and total death over the time in the second scenario

Table 12. Comparison of average fitness of agents' genetic code over the time in the second scenario

| Time | Max health | Max energy | Base strength | Base anger | Base Stress | Max age |
|---|---|---|---|---|---|---|
| 0 | 50 | 50 | 50 | 50 | 50 | 50 |
| 5 | 50 | 50 | 50 | 50 | 50 | 50 |
| 10 | 50 | 50 | 50 | 50 | 50 | 50 |
| 20 | 50 | 50 | 50 | 50 | 50 | 50 |
| 30 | 50 | 50 | 50 | 50 | 50 | 50 |
| 60 | 50 | 50 | 50.2 | 50 | 50 | 50 |
| 90 | 49.7 | 50.2 | 50.2 | 49.8 | 49.7 | 50.2 |
| 120 | 49.7 | 50.2 | 50.2 | 49.7 | 49.3 | 50.6 |
| 150 | 49.9 | 50.3 | 50.2 | 49.5 | 49.5 | 51.1 |
| 180 | 49.9 | 50.6 | 50.4 | 49.3 | 48.6 | 51.4 |
| 210 | 50.3 | 50.8 | 50.3 | 49.1 | 48.2 | 53 |
| 240 | 50.3 | 51.2 | 50.3 | 49.3 | 48.8 | 54.1 |
| 270 | 50.5 | 51.4 | 50.8 | 50 | 49.5 | 53.7 |
| 300 | 50.7 | 51.2 | 50.7 | 49.5 | 49.1 | 53.2 |
| 330 | 50.9 | 51.7 | 50.8 | 49.1 | 48.9 | 54 |
| 360 | 50.9 | 51.6 | 52.4 | 50.2 | 49.6 | 55.4 |
| 390 | 51.2 | 52.3 | 52.5 | 49.9 | 49.9 | 55.1 |
| 420 | 51.4 | 52.1 | 52.9 | 50.1 | 50.7 | 56.2 |
| 450 | 51.8 | 52.4 | 53 | 49.6 | 50.6 | 55.6 |
| 480 | 52 | 52.6 | 52.9 | 49.1 | 50.5 | 57 |
| 510 | 52.3 | 52.4 | 53.2 | 49.3 | 51.4 | 57.9 |

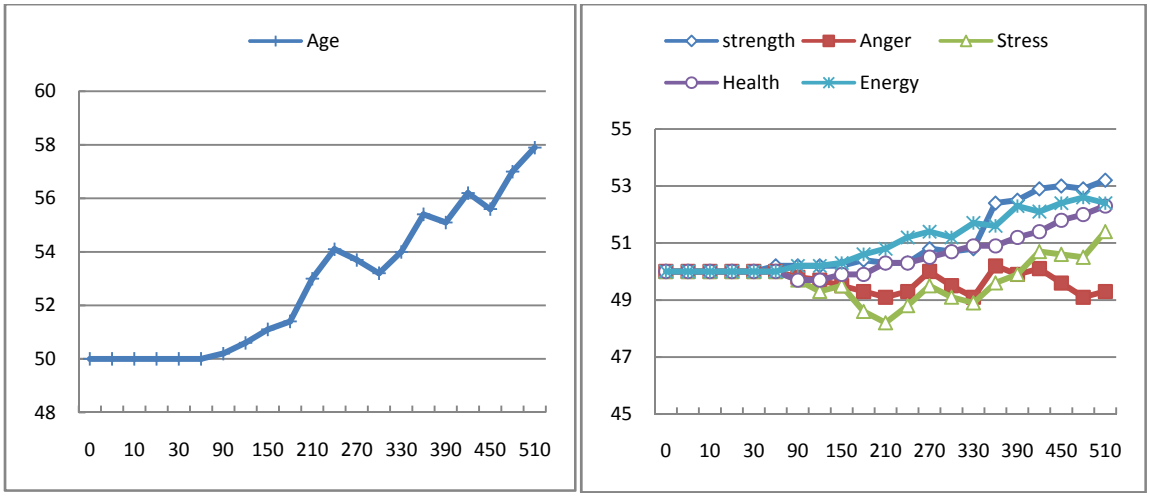


Figure 14. Graphical comparison of average fitness of agents' genetic code over the time in the second scenario

Table 13. Common learned rules by DFQL algorithm in the second scenario

| Row | Time (minute) | New rules |
|---|---|---|
| 1 | 90 | Agent Energy Is High & Agent Health Is High => Explore |
| 2 | 120 | Environment is Night & Predatory Distance Is Near => Sleep \| Move Away Predatory |
| 3 | 240 | Agent Health Is Low & Predatory Distance Is Close => Move Away Predatory |
| 4 | 330 | Agent Health Is Low & Fire Distance Is Near => Approach Fire |

New learned rules in this scenario indicate that most of the agents are willing to avoid dangerous situations. Avoiding from dangers may help agents to survive but escaping these situations will continuously increase stress level. The last three rules explain stressful behavior of the agents.

## 4.3. The Third Scenario

In this scenario the number of agents and predators are increased even more while the number of shelters is reduced greatly. The Initial number of objects in this scenario is shown in table 14.

Table 14. Initial objects of the third scenario

| Object Name | Initial Number |
|---|---|
| Human | 8 |
| Predator | 15 |
| Fire | 4 |
| Cave | 4 |
| Tree | 20 |

Again, after running the application for 510 minutes, the results represented a relative growth in population. However, the resources stayed stable. Obtained results are shown in table 15 and figure 15.

In comparison with the previous scenarios, results show that there are more fluctuations in the population plot, a major growth of total death number and a noticeable reduction of total population which is the result of the dangerous environment. However, total increment of the population plot represents relative success of agents in increasing their lifetime and making themselves more compatible with the difficult conditions in this scenario.

Evolution of the genetic code shown in above diagrams represents that although at first there is a major reduction of total age and a noticeable reduction of health and energy levels, these plots regain their stability afterward. It indicates that major growth in strength level and great reduction of stress level brings a relative stability to lifetime of the agents.

New learned rules in this scenario indicate that most of the agents are willing to fight. Attacking predators or other humans could help an agent to gain strength but on the other hand it may lead to destructive anger. Rules number 2 and 3 explain the raging behavior of agents and rule number 3 shows an urgent need to regain health.

Table 15. Quantitative comparison between population and resources over the time in the third scenario

| Time (minute) | Total birth | Total death | Population | Wild animals | Food sources |
|---|---|---|---|---|---|
| 0 | 8 | 0 | 8 | 15 | 20 |
| 5 | 8 | 0 | 8 | 15 | 20 |
| 10 | 8 | 0 | 8 | 15 | 21 |
| 20 | 9 | 0 | 9 | 15 | 20 |
| 30 | 9 | 1 | 8 | 15 | 19 |
| 60 | 12 | 1 | 11 | 15 | 19 |
| 90 | 18 | 9 | 9 | 15 | 19 |
| 120 | 23 | 10 | 13 | 15 | 19 |
| 150 | 28 | 20 | 8 | 14 | 18 |
| 180 | 27 | 20 | 7 | 12 | 16 |
| 210 | 32 | 26 | 6 | 13 | 17 |
| 240 | 38 | 27 | 11 | 14 | 19 |
| 270 | 42 | 32 | 10 | 12 | 22 |
| 300 | 49 | 33 | 16 | 13 | 21 |
| 330 | 60 | 41 | 19 | 15 | 22 |
| 360 | 71 | 44 | 27 | 17 | 19 |
| 390 | 83 | 63 | 20 | 16 | 16 |
| 420 | 91 | 63 | 28 | 15 | 15 |
| 450 | 112 | 81 | 31 | 16 | 16 |
| 480 | 131 | 81 | 50 | 15 | 14 |
| 510 | 152 | 108 | 44 | 15 | 15 |

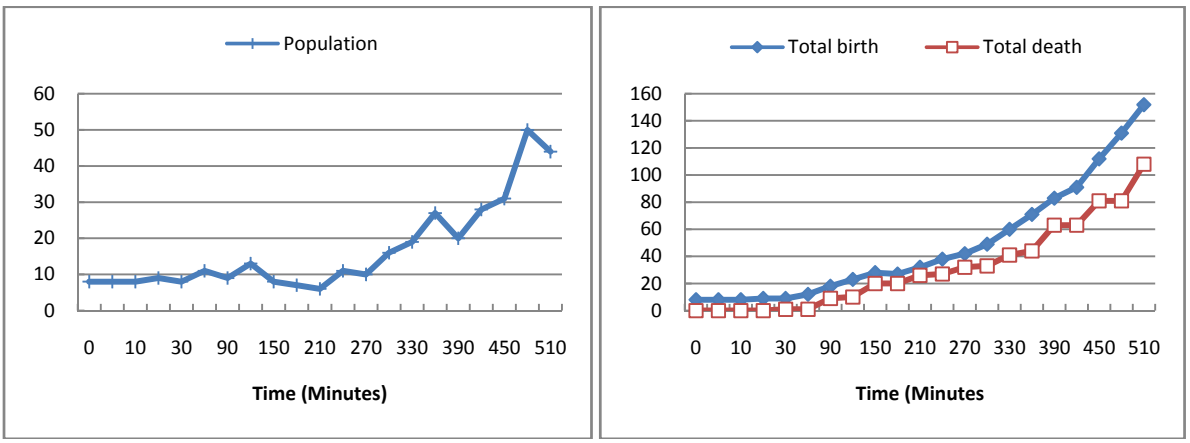


Figure 15. Graphical comparison between population, total birth and total death over the time in the third scenario

Table 16. Comparison of average fitness of agents' genetic code over the time in the third scenario

| Time (minute) | Max health | Max energy | Base strength | Base anger | Base Stress | Max age |
|---|---|---|---|---|---|---|
| 0 | 50 | 50 | 50 | 50 | 50 | 50 |
| 5 | 50 | 50 | 50 | 50 | 50 | 50 |
| 10 | 50 | 50 | 50 | 50 | 50 | 50 |
| 20 | 50 | 50 | 50 | 50 | 50 | 50 |
| 30 | 50 | 50 | 49.9 | 50 | 50 | 50 |
| 60 | 50 | 50 | 49.8 | 50 | 50 | 49.9 |
| 90 | 49.7 | 50.1 | 50 | 49.8 | 49.8 | 49.8 |
| 120 | 49.7 | 50.1 | 50 | 49.6 | 49.6 | 50.2 |
| 150 | 49.9 | 50.3 | 50 | 49.6 | 49.6 | 50.6 |
| 180 | 49.9 | 50.5 | 50 | 49.5 | 49.5 | 51 |
| 210 | 48.2 | 49.7 | 52.2 | 51.6 | 51.3 | 51.4 |
| 240 | 48.4 | 50.2 | 52.9 | 51.7 | 51.1 | 47.2 |
| 270 | 48.5 | 50.8 | 53.4 | 51.6 | 50.7 | 48.7 |
| 300 | 48.1 | 51.2 | 53.3 | 52.1 | 49.2 | 50.4 |
| 330 | 48.2 | 51.7 | 53.8 | 52 | 48.7 | 51.3 |
| 360 | 47.6 | 51.5 | 54.6 | 52.9 | 48.3 | 53 |
| 390 | 47.5 | 52 | 55.5 | 53.3 | 47.6 | 52.5 |
| 420 | 48 | 51.9 | 55.7 | 54 | 47.7 | 54.1 |
| 450 | 48.1 | 52.7 | 56.2 | 54.3 | 47.4 | 53.9 |
| 480 | 48.3 | 52.3 | 56.6 | 54.2 | 47.4 | 55.3 |
| 510 | 48.8 | 52.6 | 56.8 | 54.4 | 47.2 | 55.6 |

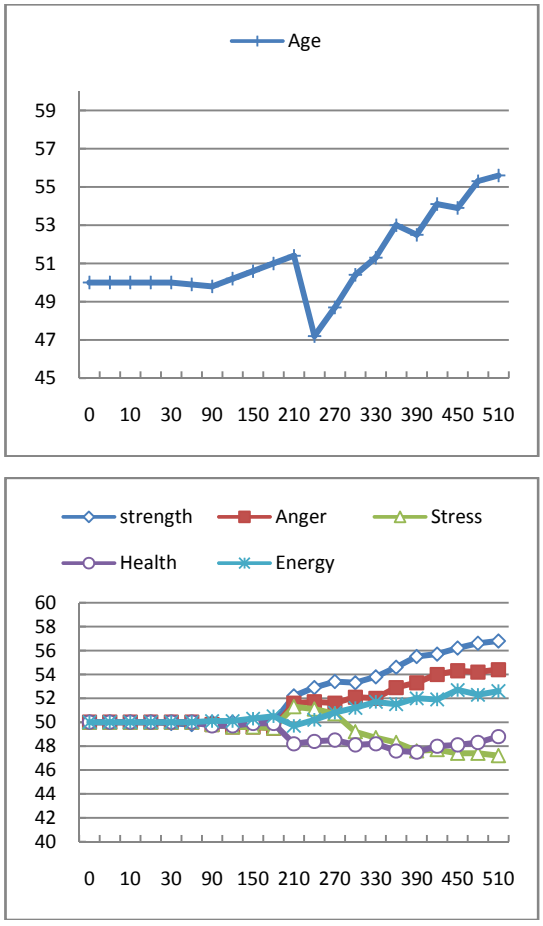


Figure 16. Graphical comparison of average fitness of agents' genetic code over the time in the third scenario

Table 17. Common learned rules by DFQL algorithm in the third scenario

| Row | Time (minute) | New rules |
|---|---|---|
| 1 | 90 | Agent Energy Is High & Predatory Distance Is Near => Explore |
| 2 | 120 | Agent Anger Is High & Predatory Distance Is Close => Attack Predatory |
| 3 | 240 | Environment Is Day & Predatory Distance Is Close => Attack Predatory \| Explore |
| 4 | 330 | Agent Health Is Low & Environment Is Night => Sleep |

- **Comparison with other Models**

As mentioned earlier in this paper, among the related works that we already addressed, Zamin is the only comparable model to Stone-age due to its capability in simulating an A-Life environment consisting of intelligent beings with human-like behaviors. The experimental results reported for Zamin model are about agents' age and change in population over the time [26-28]. In these experiments the average age of agents for two species of crisp and fuzzy are estimated and compared. Also, the magnitude of growth in population of crisp and fuzzy agents over the time is compared.

As for making a comparison between the agents' age in two models of Stone-age and Zamin, since the initial age of each agent in Stone-age is stored as a value in its gene, and this pre-defined value is very affective in the results, it is difficult to have a real comparison between the results of two models.

As for a comparison between agents' population in two models, since the increase or decrease in population depends on various parameters like sentinels number, environment dimension, CPU speed, and so on, without having the exact value of such parameters in Zamin model, a real comparison could not be performed.

# 5. Conclusion

A new A-Life simulator, called Stone-age, is presented and studied in this paper. Stone-age is a virtual-world in which a number of agents, i.e. humans, are living in an environment with the aim to increase their lifetime. The main objective of this study is to improve the decision-making process in the agents using fuzzy control. Each Agent uses a fuzzy controller structure as a basic structure and tries to improve various parts of this controller to enhance the decision-making process and enhance the efficiency. Since this world is an example of implemented A-Life, the obtained results can be compared to human behavior to some extent. Therefore, this method enables us to design various personality parameters, and review behavioral results compared to real life experiences. Novelty of the proposed model could be summarized as follows:

1) Learning in Stone-age includes not only memorizing experiences, but also processing the knowledge database and modifying the existing rules or adding new rules. This capability is established by implementing Dynamic Fuzzy Q-Learning algorithm in agents which results in storing real-time information in the fuzzy inference system.
2) Stone-age agents are able to mimic sophisticated and human-like behaviors using various personality features.
3) Same as Zamin, Stone-age utilizes the reinforcement learning based on pleasure and pain. However, the pleasure and pain in Stone-age encompasses a wider range of concepts. From this point of view, an agent is no longer just a one-dimensional creature, but it has a higher flexibility, and consequently it can initiate more various reactions in case of facing similar events.
4) The results of all three scenarios discussed in section 4, show that difficulties of the environment could direct the behavioral characteristics of agents and affect their lifetime. Results also demonstrate that the learning ability of the agents help them to overcome the environmental hazards and adapt with the environment over the time.

# References

[1] F. Menczer, and R. K. Belew, "LATENT ENERGY ENVIRONMENTS A Tool for Artificial Life Simulations," Cognitive Computer Science Research Group University of California, San Diego, 1993.

[2] H. S. Seo, S. J. Youn, and K. W. Oh, "A Fuzzy Reinforcement Function For The Intelligent Agent To Process Vague Goals," in: *Fuzzy Information Processing Society*, Atlanta, GA, USA, pp. 29-33, 2000.

[3] F. Hoffmann, and G. Pfister, "A New Learning Method for the Design of Hierarchical Fuzzy Controllers Using Messy Genetic Algorithms," *In Proceedings Sixth Int'l Fuzzy Systems Association World Congress,* pp. 249-252, 1995.

[4] P. T. Harber, Terry Jones, Stephanie Forrest, *the Echology of Echo*, Massachusetts Institute of Technology, 1997.

[5] R. Halavati, and S. BagheriShouraki, "Zamin: An Artificial Ecosystem," *EurAsia-ICT*, pp. 1008-1016, 2002.

[6] R. Gras, D. Devaurs, A. Wozniak, and A. Aspinall, "An Individual-Based Evolving Predator-Prey Ecosystem Simulation Using a Fuzzy Cognitive Map as the Behavior Model," *Artificial Life*, vol. 15, no. 4, pp.423-463, 2009.

[7] D. Devaurs, and R. Gras, "Species Abundance Patterns in an Ecosystem Simulation Studied through Fisher's Logseries," *Simulation Modeling Practice and Theory*, vol. 18, pp. 100-123, 2010.

[8] A. Dorin, K. Korb, and V. Grimm, "Artificial-Life Ecosystems: What Are They And What Could They Become?", *Proceeding of ALIF Conf.*, pp. 173-180, 2008.

[9] N. Ouannes, N. Djedi, Y. Duthen, and H. Luga, "Toward The Construction Of A Virtual Ecosystem By Evolving Virtual Creature's Behaviours," *Int'l Conf. on Multimedia Computing and Systems (ICMCS)*, 2012.

[10] A. Golestani, and R. Gras, "Regularity Analysis of an Individual-Based Ecosystem Simulation," *CHAOS'20*, 043120, 2010.

[11] D. Nauck, F. Klawonn, and R. Kruse, "Combining Neural Networks And Fuzzy Controllers," *Fuzzy Logic In Artificial Intelligence*, pp. 35-45, 1993.

[12] R. S. Sutton, and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, pp. 313-438, 1998.

[13] V. Derhamia, V. Johari Majd, and M. Nili Ahmadabadi, "Exploration And Exploitation Balance Management In Fuzzy Reinforcement Learning," *Elsevier, Fuzzy sets and systems*, vol. 161, pp. 580-582, 2010.

[14] M. Irodova, and R. H. Sloan, "Reinforcement Learning and Function Approximation," *Department of Computer Science University of Illinois at Chicago*, pp. 2-4, 2005.

[15] A. Bonarini, A. Lazaric, F. Montrone, and M. Restelli, "Reinforcement Distribution In Fuzzy Q-Learning," *Elsevier, Fuzzy sets and systems*, vol. 160, pp. 1426-1434, 2009.

[16] P. Y. Glorennec, and L. Jouffe, "Fuzzy Q-learning," *Sixth Internat.Conf. On Fuzzy Systems (Fuzz-IEEE'97)*, Barcelona, Spain, pp. 659-662, 1997.

[17] H. Berenji, "A Reinforcement Learning-Based Architecture For Fuzzy Logic Control," *Int'l Journal Of Approximate Reasoning*, vol. 6, pp. 267-292, 1992.

[18] L. Jouffe, "Fuzzy Inference Systems Learning by Reinforcement Methods: Application to A Pig House Atmosphere Control", *IEEE Transactions on systems, Application and reviews*, vol. 28, pp. 338-346, 1998.

[19] D. Gu, and H. Hu, "Accuracy based fuzzy q-learning for robot behaviors," *IEEE Int'l. Conf. on Fuzzy Systems, Budapest, Hungary*, pp. 1455-1460, 2004.

[20] C. Deng, and M. J. Er, "Efficient Implementation of Dynamic Fuzzy Q-Learning," *Fourth Int'l. Conf. on Information, Communications and Signal Processing*, pp. 1854-1858, 2003.

[21] P. Caironi, and M. Dorigo, "Training and Delayed Reinforcements in Q-Learning Agents," *University Libre de Bruxelles, Belgium*, Tech. Rep. IRIDIA/94-14, 1994.

[22] C. Deng, and M. J Er, "Online Tuning Of Fuzzy Inference Systems Using Dynamic Fuzzy Q-Learning," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 34, pp. 1478-1489, 2004.

[23] C. Deng, and M. J. Er, "Automatic Generation of Fuzzy Inference Systems By Dynamic Fuzzy Q-Learning," *IEEE Int'l. Conf. on Systems, Man and Cybernetics*, pp. 3206-3211, 2003.

[24] B. Beneš, N. Andrysco, and O. Šťava, "Interactive Modeling of Virtual Ecosystems," *Eurographics Workshop on Natural Phenomena*, 2009.

[25] Q. Lu, Z. Peng, F. Chu, and J. Huang, "Design Of Fuzzy Controller For Smart Structures Using Genetic Algorithms," *Smart Materials and Structures*, vol. 12, pp. 979- 986, 2003.

[26] F. Hoffmann, and G. Pfister, "A New Learning Method for the Design of Hierarchical Fuzzy Controllers Using Messy Genetic Algorithms," *In Proceedings Sixth International Fuzzy Systems Association World Congress*, pp. 249-252, 1995.

[27] D. Nauck, and R. Kruse, "A Fuzzy Neural Network Learning Fuzzy Control Rules And Membership Functions By Fuzzy Error Back Propagation," *(NN-IEEE'93),* pp. 1022-1027, 1993.

[28] R. Halavati, and S. Bagheri Shouraki, "A Fuzzy Artificial World: Zamin II," *ICCS 2003*, pp. 601-609, 2003.

[29] M. A. Bedau, "Artificial Life: Organization, Adaptation and Complexity from the Bottom Up," *TRENDS in Cognitive Sciences*, vol.7, no.11, 2003.

[30] N. Wiener, *Cybernetics, or Control and Communication in the Animal and the Machine*, MIT Press, 1965.

[31] J. Von Neumann, "Theory of Self-Reproducing Automata," *Urbana Champagne: University of Illinois Press*, 1966.

**Ali Ahmadi** Received his B.Sc. in Electrical Engineering from Amir kabir University, Tehran, Iran in 1991 and his M.Sc. and Ph.D. degree in Artificial Intelligence and Soft Computing from Osaka Prefecture University, Japan in 2001 and 2004, respectively. He worked as a researcher in Research Center for Nano devices and Systems in Hiroshima University, Japan during 2004–2007. He has been with K. N. Toosi University of Technology, Tehran, Iran as assistant professor from 2007. His research interests include distributed intelligent systems, human-computer interaction, computational intelligence, adaptive and interactive learning models, virtual reality and artificial life.

**Email:** ahmadi@kntu.ac.ir

**Bahman Aminipour** studied Applied Mathematics from 2002 to 2004 in Ferdowsi University and Software Engineering in Sadjad University of Technology during 2004 to 2008 in Mashad, Iran. He received his B.Sc. in Software Engineering in 2008. He then took his M.Sc. degree in Artificial Intelligence at the Islamic Azad University - Science and Research branch, Tehran, Iran in 2012. He has been working as a software programmer, video game designer and R&D specialist since 2009 and his research interests include human-computer interaction, game architecture and design, game theory, and fuzzy learning models.

**Email:** bahman.aminipour@gmail.com

**Paper Handling Data:**

Submitted: 15.12.2016
Received in revised form: 05.10.2017
Accepted: 02.12.2017
Corresponding author: Dr. Ali ahmadi,
Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran.