Artif Life Robotics (2001) 5:142–147

ORIGINAL ARTICLE

Sung-Bae Cho

Applications of artificial life to developing robot and softbot

Received: November 21, 2000 / Accepted: May 30, 2002

Abstract Adaptation gives rise to a kind of complexity that greatly hinders our attempts to solve some of the most important problems currently posed by our world. Recently, there has been an attempt to build a complex adaptive system which is rich in autonomy and creativity, with the ideas and methodologies of artificial life (A-life). This article presents the concepts and methodologies of A-life, and shows two typical applications based on them. These systems not only develop new functionality spontaneously, but also grow and evolve their own structure autonomously. They have been applied to controlling a mobile robot and developing adaptive agents on the world-wide web.

Key words Artificial life · Complex adaptive systems · Robot · Softbot

1 Introduction

Intelligent systems can adaptively estimate continuous functions from data without specifying mathematically how outputs depend on inputs. System behavior is called *intelligent* if the system emits appropriate problem-solving responses when faced with problem stimuli. Recently, some researchers have tried to synthesize intelligent systems by using artificial life (A-life) technologies.

A-life research aims at studying man-made systems which exhibit behaviors characteristic of natural living systems. It complements traditional biological sciences which are concerned with the analysis of living organisms by attempting to synthesize life-like behavior within computers; extending the empirical foundation upon which biology is based from *life as we know it* to a larger picture of *life as it could be*.[1] The essential features of A-life models are as follows:

- they work with populations of simple programs, where no single program directs all the other programs;
- each program details the way in which a simple entity reacts to local situations in its environment, including encounters with other entities;
- there are no rules in the system that dictate global behavior, and higher behavior is therefore emergent.

It is the concept of emergent properties that shows the nature of A-life research. Emergent properties are exhibited by a collection of interacting entities whose global behavior cannot be reduced to a simple aggregate of the individual contributions of these entities. In artificial intelligence, conventional methods cannot reveal or explain the emergent properties because they are generally reductionist, i.e., they decompose a system into its constituent subsystems and then study these in isolation using a top-down approach.

However, A-life adopts a bottom-up approach, which starts with a collection of entities exhibiting simple and well-understood behavior patterns, and synthesizes these into more complex systems. Many technologies are used in A-life research, such as cellular automata, the Lindenmayer system, the genetic algorithm, neural networks, and so on, but the key idea is the evolutionary algorithm. In this sense, a practical goal of A-life can be redefined as finding a mechanism for an evolutionary process to be used in the automatic design and creation of artifacts. Figure 1 shows the main research areas in artificial life.

The genetic algorithm, one of the evolutionary algorithms, is a model of machine learning derived from the procedure of evolution in nature. This is done by creating a population of individuals that are represented by chromosomes. A chromosome can be thought of as a string of human genes. The individuals in the population go through evolution. This is an evolutionary procedure in which different individuals compete for resources in the environment.

S.-B. Cho (✉)
Department of Computer Science, Yonsei University, 134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, Korea
Fax +82-2-365-2579
e-mail: sbcho@cs.yonsei.ac.kr

This work was presented in part at the Fifth International Symposium on Artificial Life and Robotics, Oita, Japan, January 26–28, 2000

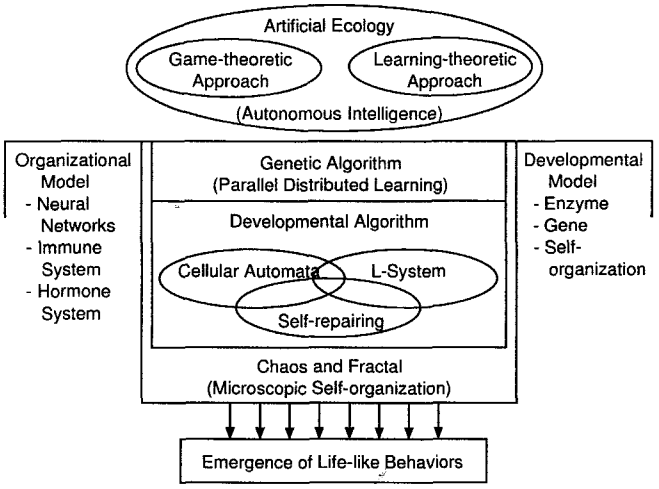

Fig. 1. The main research areas in artificial life

Better individuals are more likely to survive and propagate their genetic material to their offspring.

2 Application to a mobile robot

There have been several attempts to develop an artificial brain using engineering techniques. Among these, CAM-brain develops neural networks based on cellular automata by evolution. Owing to their particular features, cellular automata (CA) can be evolved very quickly on parallel hardware such as CAM-8 at MIT, or CBM at ATR.[2]

Evolutionary engineering (EE) is an approach used to evolve neural network modules with particular functions in order to develop an artificial brain. It has been extensively exploited to apply each neural network module to a specific problem. We have attempted to evolve a module of CAM-brain for the problem of robot control, especially the Khepera simulator. A simulation means that an appropriate neural architecture emerges to make the Khepera simulator navigate the given environment without bumping against walls and obstacles. This section shows the power of the model based on A-life technology by analyzing the robot behavior and corresponding neural networks evolved.

CAM-brain is an evolved neural network based on CA. This article uses one of the CAM-brain models, the CoDi model, and the process of developing neural networks and signaling among neurons. This process consists of two phases. One is a growth phase that builds the structure of the neural network. The other is a signaling phase that sends and receives signals among neurons.

2.1 Method

2.1.1 CA in CoDi

Cellular automata consist of a state, a neighborhood, and a program.[3] Each cell in the CA space of CAM-brain has one of four states: neuron, axon, dendrite, and blank. If the state of the cell is blank, it is an empty space. Blank cells do not participate in any cell interactions during the signaling of neural networks. Neuron cells collect neural signals from the surrounding dendrite cells. If the sum of the collected signals is greater than the threshold, then the neuron cells send the signals to the surrounding axon cells. These cells distribute signals originating from neuron cells. Dendrite cells collect signals and eventually pass them to neuron cells.[4]

Neighborhood cells are the surrounding cells (north, south, west, and east) in 2-D CA space (top and bottom are added in 3-D CA space). The state of each cell, and the program or rule deciding the state of each cell with the states of its neighbors, are decided by the chromosomes. One chromosome has the same number segments as cells, and can make one neural network. One segment corresponds to one cell, and can change a blank cell to a neuron cell. It also decides the directions in which to send received signals to neighborhood cells.

2.1.2 Growth phase

The growth phase organizes the neural structure and makes the signal trails among the neurons. Neurons are seeded in CA-space by the chromosome. The neural network structure grows by sending out two types of growth signal (axon and dendrite) to neighborhood cells. A neuron sends axon growth signals in two opposite directions, as decided by the chromosome, and dendrite growth signals in the four remaining directions.

The neighborhood cells become axons or dendrites according to the type of growth signal received. They then propagate the growth signal received to a neighborhood cell. Each axon cell and each dendrite cell belongs to only one neuron cell. Once the type of cell is decided, it never changes. The neural network is constructed and encoded to the chromosome, and it is then evolved by the genetic algorithm.[4]

2.1.3 Signaling phase

The signaling phase transmits a signal continuously from input cells to output cells. The signaling trails are formed by the evolved structure at the growth phase. Each cell has a different role according to its type. If the cell is a neuron, it gets a signal from neighborhood dendrite cells and gives a signal to neighborhood axon cells when the sum of the signals is greater than its threshold. If the cell is a dendrite, it collects data from the facing cell and eventually passes it to the neuron body. If the cell is an axon, it distributes data originating from the neuron body.

The position of the input and output cells in CA-space is decided in advance. At first, if input cells produce a signal, it is sent to facing axon cells which distribute that signal. Then, neighborhood dendrite cells belonging to other neurons collect this signal and send it to connected neurons. The neurons that receive the signal from dendrite cells send it to axon cells.

Finally, dendrite cells of the output neuron receive the signal and send it to the output neurons. The output value can be obtained from the output neurons. During the signaling phase, a fitness evaluation is executed. According to the given task, various methods can be used. This fitness is used for the evolution of the chromosome.

2.1.4 Mobile robot control

Applying CAM-brain to the control of a mobile robot[5,6] requires the following process. The neural network structure is made in the growth phase. In the signaling phase, sensor values from the Khepera simulator[7] are used as inputs to CAM-brain. CAM-brain transmits signals from input to output cells. As the output values of CAM-brain are input to the Khepera simulator, the robot moves. When the robot bumps against an obstacle or reaches its goal, its fitness is computed. Chromosomes are reproduced in proportion to the result of this evaluation.

There are two main problems with applying this model to controlling the robot. Because CAM-brain cannot utilize the activation values of the robot sensors perfectly, the activated range of the input neuron is varied according to the input value. In addition, because a delay time is needed until CAM-brain gives an output value, we must execute a dummy signaling phase until the signals from the input cells arrive at the output cells. This means that the robot can react appropriately in several situations.[7]

2.2 Results

The robot controller evolves in $5 \times 5 \times 5$ CA space to facilitate easy analysis. After the 21st generation, individuals with a fitness value of one keep appearing. Figure 2 shows the trajectory of a successful robot. This is less smooth than that obtained in our previous work, but this robot controller is smaller, which makes the analysis easier.

Figure 3 shows the architecture of the neural network evolved. The dotted arrows represent inhibitory connections, and the solid arrows represent excitatory connections. This diagram has been obtained by tracing the activation values of each neuron. There are 12 neurons, but neurons 8, 11, and 12 are not functional because they are not in the path of the input or output neurons. Neurons 2 and 10 are output neurons which produce the velocity of the left and right wheels, and neurons 3, 4, 5, and 6 are input neurons. Neuron 3 is for the front sensor of the robot, neurons 5 and 6 are for the left sensors of the robot, and neuron 4 is for the right sensor of the robot.

The architecture of the controller has direct connections from input to output neurons. These connections play a role in turning left and right: if neuron 5 has a high activation value (which means that there is no obstacle on the left side of the robot because the sensor values of the robot are scaled inversely), neuron 10 (right wheel) produces a positive signal (because neuron 5 fires an excitatory signal to neuron 10). If neuron 4 has a low activation value (there is an obstacle on the right side of the robot), neuron 2 (left wheel) produces 0 (because neuron 5 cannot fire neuron 2). The velocity of each wheel is decided according to the value of the output neurons. It becomes 5 (if the output value is positive), −5 (if negative), or 0 (otherwise). The values of the output neuron make the robot turn left. Similarly, the robot can turn right with an output from neuron 7.

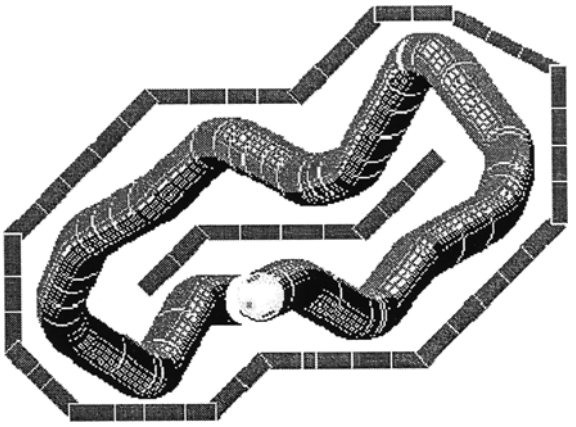

Fig. 2. The trajectory of a successful robot

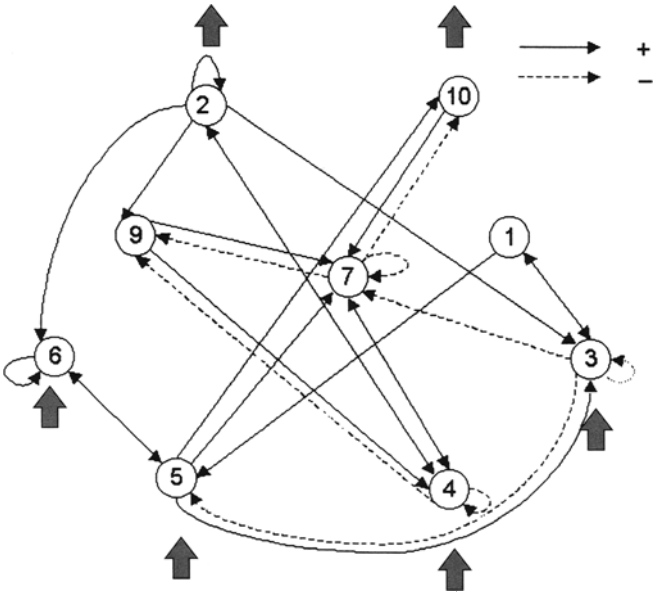

Fig. 3. Schematic diagram of the neural network evolved

3 Application to softbot

The world-wide web (WWW) has a large, widely distributed collection of documents, which can be added to,

deleted, or modified dynamically. Moreover, the document style is varied. It takes considerable time and effort for users to search the web in this environment. For these reasons, several search engines have been investigated and developed.

Conventional search engines for retrieving information on the web are devised for mainly static and nondistributed environments. With these engines, the end user sends queries to the server that maintains the index files to get the relevant document lists. The user's requests are processed through the use of index files, which are made and updated by off-line robot agents that collect and analyze the documents. Because of their fast response time, these search engines are in general use, but they have several limitations. First, they cannot cope with dynamic changes in documents. Second, they can delete important data by incorrect indexing, and by missing the relations between documents. Third, they cannot reflect the user's preferences or habits. To overcome these limitations, a new method is required to replace index-based robot agents.

Our A-life agent is very similar to Infospider, that was originally proposed by Menczer.[8] It has a population of online agents that search documents by deciding their own actions locally. Each agent in a population can reproduce or disappear according to the relevancy of the documents retrieved by the agent. The population of agents converges to optimal states through evolution. However, if we incorporate the user's preference, we can provide accurate information more quickly, and personalize the agents for each user. By updating the user profile at each query, we can reflect the user's preferences. A-life agents maintain their competence by adapting to the user's preference, even though this may change over time.

Several methods have been proposed to retrieve more accurate information by using the web's large, dynamic, distributed environment. Autonomous agents or semiintelligent agents could manage the large amounts of information available online, and estimate the user's preferences and habits.[9] The weighted keyword vector representation is applied to WWW information filtering.[9,10] Several machine-learning techniques have been suggested to produce effective information agents. These would yield, for example, agents that perform look-ahead searches and provide suggestions to the user on the basis of reinforcement learning.[10] NewT is another multiagent system that uses evolution and relevance feedback for information filtering.[11]

3.1 Method

The authors of web documents tend to classify them according to subjects, and connect them in related topics. This tendency results in a semantic topology, which defines the correlation of documents. If some documents are relevant to the user, the links in the current document are also highly relevant to the user. Also, the links close to meaningful keywords are probably more useful than other links. The artificial-life agent can reduce the search space by using this property. It has a population of multiple retrieval agents. The energy of each agent in the population is increased or decreased by the relevance of the document retrieved by the agent itself. This method uses the genetic algorithm based on local selection. The algorithm is given below.

```
Initialize agents;
Obtain queries from user;
while (there is an alive agent) {
  Get document D_a pointed by current agent;
  Pick an agent a randomly;
  Select a link and fetch selected document D_a';
  Compute the relevancy of document D_a';
  Update energy (E_a) according to the document relevancy;
  if (E_a > ε)
    Set parent and offspring's genotype appropriately;
    Mutate offspring's genotype;
  else if (E_a < 0)
    Kill agent a;
}
Update user profile.
```

3.1.1 *Initialization*

Each agent's starting point is initialized by a user profile. The genotype is composed of confidence and energy. Confidence is the degree to which an agent trusts the descriptions that a document contains about its outgoing links, and energy represents the agent's relevancy to the given queries. The energy is initialized to a constant threshold $\varepsilon/2$, and confidence is chosen randomly.

3.1.2 *Link selection*

The relevancy of each link in the current document is estimated by computing the physical distances to keywords matched to the user's queries. This estimation is based on the assumption that any links close to the keywords are generally more relevant to the user's interest than other links. For each link l in a document, the relevancy is calculated as

$$\lambda_l = \sum_{k \in \text{tokens}} \frac{\text{match}(k,Q)}{\text{distance}(k,l)} \tag{1}$$

where k is the number of tokens in document D_a, Q is the number of queries, and distance(k,l) is the number of links separating k and l in the document. Here match(k,Q) is 1 if k is in Q. Otherwise, it becomes 0. To select a link to follow, we use a stochastic selector to pick a link with a probability distribution which is scaled up and normalized by the agent's confidence.

Confidence evolves by selection, reproduction, and mutation. Different confidence values can implement search strategies such as best-first, random walk, or any middle course. With this distribution, an agent selects which link to follow.

3.1.3 Updating energy

After the agent has fetched the document reached by the selected link, it estimates the relevancy of the document. This is proportional to the hit rate of the number of keywords to the whole tokens in the document. The relevancy of the document is represented by

$$r(D_a) = \sqrt[4]{\sum \frac{\text{match}(k,Q)}{\text{distance}(k,D_a)}} \tag{2}$$

where number(k,D_a) is the number of keywords in D_a. An agent's energy is updated according to the relevancy of the document. The use of the network resource means a loss of energy. If the document has already been visited, an increase in energy is not expected.

$$E_A = E_A - \text{expense} + r(D_a) \qquad \text{if } D_a \text{ is new} \tag{3}$$

where $r(D_a)$ is the relevancy of the document, and expense is the loss of energy.

3.1.4 Reproduction

Each agent can reproduce offspring or be killed by comparing the agent's energy with a constant threshold ε. If the agent's energy exceeds the threshold, it reproduces offspring. The offspring's energy is fed by splitting the parent's energy, and the offspring is mutated to provide the evolution with the necessary variation. The confidence boundary is determined by the relevancy of the current document. This mechanism can cause the population of agents to be biased toward regions where the relevant documents exist.

3.1.5 Updating the user profile

The user profile should reflect the user's interests. Since the agents learn about the user's interests by getting the user's queries and feedback, it is important to update the user profile after each search. The updated user profile is composed of relevant document uniform resource locators (URLs) and other interesting subjects. With this property, a user can personalize the agents as queries are repeatedly given.

3.2 Results

In order to provide a fair and consistent evaluation of the system's performance, we restricted the search space to the local machine instead of the real Web. We collected a number of hyper text markup language (HTML) pages on various topics, classified the pages according to subject, and put them in different directories. The initial user profile was composed of the top directories of the local machine. The initial number of agents depends on the number of URLs in the user profile. We compared the A-life agents with breadth first search (BFS) and random search agents. BFS searches all documents exhaustively, while the A-life agents can search documents selectively.

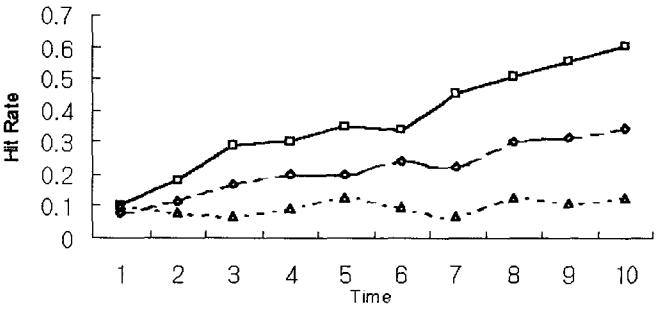

Fig. 4. Hit rate on relevant documents. *Squares*, preposed; *diamonds*, breadth first search; *triangles*, random search

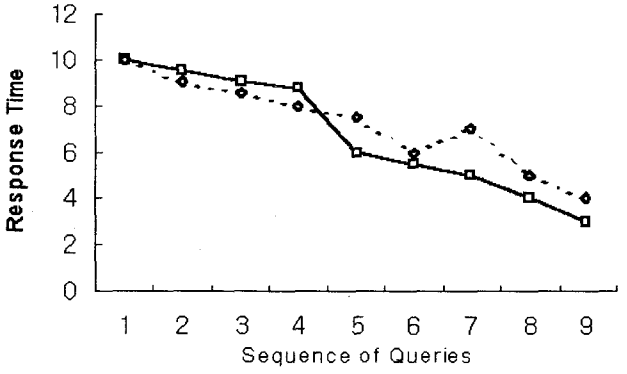

Fig. 5. Response times for queries in the same category. *Squares*, task 1; *diamonds*, task 2

The initial population was ten agents. The population size has no limitation in run time. The constant threshold ε was set to 0.4. An agent whose energy is greater than ε can reproduce offspring. The initial agent's energy is set at ε/2. This agent uses the network resources, which means a loss of energy. This loss of energy is called "expense," and set at 0.1. The process is influenced by the expense value. If we increase the expense value, the agents have less chance to search further. Irrelevant agents may disappear quickly, and there is some possibility that even some relevant agents can disappear without searching the regions sufficiently. We selected the expense value by trial and error.

The most important property of the A-life agents is that they can discard useless agents which irrelevant to the user's preference. Figure 4 shows the hit rate on relevant documents. In the beginning, the performance was no better than other search methods, but it improved rapidly. This result implies that each agent can effectively cut out irrelevant document paths. The action of each agent gradually goes toward relevant document paths. By using this property, A-life agents can reduce access to irrelevant documents.

We tested the performance improvement in cases where the user gives all queries in the same category. For each query, the user profile is updated according to document relevancy, and the agents adapt to the user's preference. If the user gives queries in the same category, our agents improve their response time according to these queries. Figure 5 shows the results of two tasks. In task 1, the

sequence of queries is computer, artificial intelligence, neural network, agent, evolution, user feedback, retrieval, and search. In task 2, the sequence is computer, document style semantics and specification language (DSSSL), standard generalized markup language (SGML), grove, property set, repository, and database. Initial response time is not good, but as the queries are given repeatedly, we can see an improvement in the response time for the two tasks.

4 Concluding remarks

This article has introduced the key concepts of A-life technology, and shown its potential in applications such as controlling a mobile robot and developing adaptive agents on the WWW. While artificial intelligence uses the technology of computation as a model of intelligence, A-life is attempting to develop a new computational paradigm based on biological processes.

Acknowledgment This work was supported by A Korea Research Foundation Grant (KRF-2002-005-H20002) and Super Intelligent Chip Project.

References

1. Langton CG (1989) Artificial life. Proceedings of Artificial Life. Addison-Wesley, Reading, p 1–48
2. De Garis H (1996) CAM-Brain: ATR's billion neuron artificial brain project. A three-year progress report. Proceedings of an International Conference on Evolutionary Computation, IEEE, Piscataway, p 886–891
3. Green DG (1993) Cellular automata. http://www.csu.edu.au/complex_systems/tutorial1.html
4. Gers F, de Garis H, Korkin M (1997) CoDi-1Bit: a simplified cellular automata-based neuron model. Proceedings of an Artificial Evolution Conference
5. Floreano D, Mondada F (1996) Evolution of homing navigation in a real mobile robot. IEEE Trans Syst Man Cybern 26:396–407
6. Michel O (1995) Khepera simulator version 1.0. User Manual
7. Cho SB, Song GB, Lee JH, et al. (1998) Evolving CAM-Brain to control a mobile robot. Proceedings of an International Conference on Artificial Life and Robotics, AROB, Oita, p 271–274
8. Menczer F (1997) ARACHNID: adaptive retrieval agents choosing heuristic neighborhoods for information discovery. Proceedings of the 14th International Conference on Machine Learning
9. Balabanovicand M, Shoham Y (1995) Learning information retrieval agents: experimental with automated web browsing. Proceedings of AAAI SSS Information Gathering from Heterogeneous Distributions Envst
10. Armstrong R, Freitag D, Joachims T, et al. (1995) Web watcher: a learning apprentice for the world wide web. Proceedings of AAAI SSS Information Gathering from Heterogeneous Distributions. Envst
11. Sheth B, Maes P (1993) Evolving agents for personalized information filtering. Proceedings of the 9th Conference on Artificial Intelligence for Application