Front. Electr. Electron. Eng. China 2008, 3(4): 394–398
DOI 10.1007/s11460-008-0078-3

RESEARCH ARTICLE

Haifei YU, Dingwei WANG

Research on food-chain algorithm and its parameters

Abstract Based on the characteristics of colony emergence of artificial organisms, their dynamic interaction with the environment, and the food-chain crucial to the life system, the rules of local activities of artificial organisms at different levels are defined. The article proposes an artificial life-based algorithm, which is referred to as the food-chain algorithm. This algorithm optimizes computation by simulating the evolution of natural ecosystems and the information processing mechanism of natural organisms. The definition, idea and flow of the algorithm are introduced, and relevant rules on metabolic energy and change in the surroundings where artificial-life individuals live are depicted. Furthermore, key parameters of the algorithm are systematically analyzed. Test results show that the algorithm has quasi-life traits that include being autonomous, evolutionary, and self-adaptive. These traits are highly fit for optimization problems of life-like systems such as the location-allocation problem of a distribution network system.

Keywords artificial life, food-chain algorithm, ecosystem, colony emergence, adaptive behavior

1 Introduction

The concept of artificial life [1] (Alife) was first proposed by Langton C. G. in 1987 and defined as 'the study of man-made systems that exhibit behavior characteristic of natural living systems'. Based on artificial life, some types of artificial life algorithms were proposed. Hayashi, Yang and Lee [2] proposed an artificial life algorithm in 1996 and 2000. Their research was mainly based on the ideas that Alife individuals can interact with each other in their environment, and their local activities may lead to emergent colonization. Menczer and Belew [3] introduced a novel artificial life model, i.e., the latent energy environment (LEE). Based on the model, they designed an artificial life algorithm and applied it to information retrieval. Basically, the artificial life algorithm can be used to simulate and optimize complex system behaviors such as that of a power transmission grid, traffic modeling project and molecular structure optimization of pharmaceutical. Other new intelligent algorithms provide an insight into addressing complex optimization problems [4,5]. However, research on artificial life algorithm is still in its infancy [6].

The food-chain [7] is important and pervasive in the living system, which refers to the transfer of food energy from its source in plants through herbivores to carnivores. It is a connected life cycle of eating and being eaten. For example, in a grass-rabbits-wolves food-chain, rabbits that feed on grass are food for wolves. According to the optimal foraging theory and behavioral ecology, natural selection is always apt to drive organisms to transfer their genes in the most effective ways [8]. Assad and Packard [9] studied a general phenomenon (colonization) of distributed agents called artificial organisms (Artorgs). Emergent colonization is also important and pervasive in the living system. A colony is considered a group of spatially clustered organisms, which is relatively stable in size over time and exhibits a high level of interaction amongst its members. Thus, emergent colonization of Artorgs is chosen as the optimization mechanism of the food-chain algorithm we proposed.

Based on the ideas above, we built an artificial food-chain system composed of several kinds of Artorgs to simulate the food-chain, animal predatory behaviors and colony emergence. We then proposed a new artificial-life algorithm called food-chain algorithm (FCA) for the artificial food-chain system, which simulates the evolution of a natural ecosystem and the information expression and processing mechanism of natural organisms to achieve optimization computation.

In the following sections, we will first describe how to build an artificial food-chain system in an artificial world

Translated from *Journal of Northeastern University* (*Natural Science*), 2007, 28(7): 993–997 [译自: 东北大学学报 (自然科学版)]

Haifei YU (✉)
School of Business Administration, Northeastern University, Shenyang 110004, China
E-mail: hfyu@mail.neu.edu.cn

Dingwei WANG
Information Science and Engineering School, Northeastern University, Shenyang 110004, China

(Aworld) [9]. Second, the design of the food-chain algorithm is described, including its definition, ideology and procedure. Third, the key parameters of the algorithm are presented in detail. Finally, conclusions are given.

2 Artificial life-based food-chain algorithm

2.1 Ideology of a food-chain algorithm

The artificial life system proposed by Andrew and Norman is a computational model of organisms in an artificial ecology where colonization emerges through a process of resource gathering and exchanging in an evolving population. Artificial world (Aworld) is a square Cartesian plane containing N-by-N discrete points. In this approach, each point in the Aworld can contain food resources and can also be inhabited by members of different kinds of Artorgs.

According to ecology, food-chain refers to the transfer of food energy from its source in plants through herbivores to carnivores. Elton pointed out that the length of food-chains was almost limited within three to five links [7]. We first use the Aworld as the simulation tool to build an artificial food-chain system that includes three kinds of Artorgs, i.e., top Artorgs, intermediate Artorgs and basal Artorgs. Second, we define the food-chain relationship: the top Artorgs eat waste produced by the intermediate Artorgs and produce white waste; the intermediate Artorgs eat waste produced by the basal Artorgs, which then eat waste produced by the top Artorgs and thus form the food-chain relationship.

In the Aworld, the Artorgs move, seek different types of food, consume energy resources, produce waste, mate, multiply and die. Artorgs can only metabolize the resources they want, with each movement leading to an increase or loss of their energy. For each movement, if an Artorg finds food in its neighborhood region δ, it will eat the food and its internal energy increases by E_e, otherwise, its internal energy will decrease by E_p. If its energy is more than the grown-up energy level e^g, it will have the chance to multiply. If its energy is less than the dead energy level e^d, it will die and disappear from the Aworld. Otherwise, it keeps its current status.

On the other hand, Artorgs have a sensory system that enables them to see resources as well as other Artorgs in the Aworld. They are also able to determine the location of the nearest resources and other Artorgs from their present locations. The nearest resource becomes their goal and drives them to move forward. The neighborhood region of x_s, C, is the space within the Euclidean length. C is written as

$$C = \{x \in \boldsymbol{R}^2 | \, ||x - x_s|| \leqslant \delta\}, \tag{1}$$

where δ is the neighborhood region of Artorgs per generation. Artorgs can only see and find resources and other organisms within their area for each generation. δ can be considered as a constant for the whole generation or a variable according to increasing generations.

In case of the function optimum problems, the Aworld becomes the space of optimizing values. Every location has its own optimizing value. After Artorgs seek the neighborhood region randomly, they produce their offspring at the locations with fitness levels higher than their own. Therefore, Artorgs can produce an emergent colonization at locations that have the optimum of objective functions.

2.2 Design of food-chain algorithm

To make the artificial food-chain system work, the parameters of Artorgs and their local activity rules should be defined according to the "bottom to up model", which is the most important principle of Alife. First, the parameters of Artorgs are defined as follows:

1) e_i^0, the initial energy level of Artorg i.
2) e_i^g, the grown-up energy level of Artorg i.
3) e_i^d, the dead energy level of Artorg i.

Second, we define energy rules of Artorgs as

$$e_i^g = (1+\varepsilon)e_i^0, \tag{2}$$

where ε is a constant parameter, $\varepsilon \in (0, 1)$;

$$e_i^d = (1-\eta)e_i^0, \tag{3}$$

where η is a constant parameter, $\eta \in (0, 1)$.

Finally, we define the neighborhood region of Artorgs as

$$\delta_t = \delta_0 r^{(1-(t/T))^\lambda}, \tag{4}$$

where δ_0 is the initial neighborhood region of the Artorg; δ_t, the tth of generations of Artorgs; r, $r \in (0, 1)$, is a random parameter of the Artorg; λ, $\lambda \in (2, 5)$, is a discordance parameter of the Artorg.

In fact, δ_t is a very important variable. It limits the movement area of Artorgs, which seek and consume food, mate, and multiply.

The procedure of the algorithm is designed as follows:

Step 1 Initialization. Three commensurable kinds of Artorgs are placed at random locations in the Aworld. Set the initial energy parameters of Artorgs, e_i^0, e_i^g, e_i^d, and the neighborhood region δ_i^0. Set the maximal generation number T.

Step 2 Seek food resources. The Artorgs seek food resources around their neighborhood region δ_t. If an Artorg finds one, it eats the food and its energy increases by E_e. The food location is the current local optimum ($\text{Opt}_{\text{local}}$). If $\text{Opt}_{\text{local}}$ is better than the global optimum $\text{Opt}_{\text{global}}$, update $\text{Opt}_{\text{global}}$ with $\text{Opt}_{\text{local}}$, otherwise, if it

fails to find any food resource, its energy will decrease by E_p.

Step 3 Update location. If the Artorgs find food resource, they move towards their $\mathrm{Opt}_{\mathrm{local}}$, otherwise they move at random in their neighborhood region.

Step 4 Update the neighborhood region δ_t according to Eq. (4).

Step 5 Produce food resources. All Artorgs produce new food resources and place it at random locations in their neighborhood region.

Step 6 Evolution. Check the internal energy of each Artorg e_i. If it reaches the grown-up energy level e_i^g, the Artorgs multiply and place their offspring at a random location in the neighborhood region. Reset the internal energy of the father e_i with e_i^0, and reset δ_t with δ_i^0. Otherwise, if its energy is less than the dead energy level e_i^d, it dies and is removed from the Aworld.

Step 7 Stop criterion. A generation is increased by 1. If it reaches the maximal generation number, stop the algorithm; otherwise return to Step 2.

3 Analysis of rules and parameters of food-chain algorithm

The parameters and rules of the food-chain algorithm that influence its performance include energy rules, neighborhood region δ_0, population size, food-chain length and the maximal generation number T. We will choose two standard test functions [10] (see Table 1) to test the algorithm. The food-chain algorithm is programmed with Java and run in the following environment: Windows XP, 128 MB memory, Intel Pentium III-733 MHz CPU.

Table 1 Standard test functions

name	formula	note
Rosenbrock	$f_{\mathrm{R}}(\vec{x}) = 100\left(x_2 - x_1^2\right)^2 + (x_1 - 1)^2$	$x_1, x_2 \in [-2.048, 2.048]$; the global minimum f_{R} is 0, at (1,1).
Schaffer	$f_{\mathrm{S}}(\vec{x}) = 0.5 + \dfrac{\left(\sin\sqrt{x_1^2 + x_2^2}\right)^2 - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}$	$x_1,\ x_2 \in [-100,\ 100]$; the global minimum f_{S} is 0, at(0,0).

3.1 Analysis of energy rules

The energy rules of Artorgs are introduced in Sect. 2.2. $(\varepsilon,\ \eta)$ is key parameter of the energy rules which influences the robustness of the algorithm and determines whether the food-chain will be broken when faced with some species of food-chain extinction. Hence, we define a new terminology–the brokenness ratio of the food-chain–which means the ratio of the food-chain being broken. We can study the brokenness ratio to analyze how $(\varepsilon,\ \eta)$ affects algorithm robustness.

The parameters of the algorithm are initialized as follows: the length of the food-chain is 3; the initial population size is 20; the maximal population size is 40; the maximal generation number T is 2000 and the algorithm repeats 50 times. Taking the Rosenbrock function for an example, we recorded the simulation results as follows (see Table 2):

Table 2 Analysis between $(\varepsilon,\ \eta)$ and brokenness ratio of food-chain

(ε,η)	number of brokenness	brokenness ratio	δ_0
(0.9, 0.1)	50	1	0.2
(0.8, 0.2)	50	1	0.2
(0.7, 0.3)	49	0.98	0.2
(0.6, 0.4)	46	0.92	0.2
(0.5, 0.5)	43	0.86	0.2
(0.4, 0.6)	34	0.68	0.2
(0.3, 0.7)	29	0.58	0.2
(0.2, 0.8)	19	0.38	0.2
(0.1, 0.9)	11	0.22	0.2
(0.9, 0.1)	49	0.98	0.3
(0.8, 0.2)	43	0.86	0.3
(0.7, 0.3)	33	0.66	0.3
(0.6, 0.4)	20	0.4	0.3
(0.5, 0.5)	11	0.22	0.3
(0.4, 0.6)	2	0.04	0.3
(0.3, 0.7)	0	0	0.3
(0.2, 0.8)	0	0	0.3
(0.1, 0.9)	0	0	0.3

It is shown in Table 2 that the brokenness ratio is very sensitive to $(\varepsilon,\ \eta)$. Under the condition $\delta_0 = 0.2$, the brokenness ratio is very high, and the food-chain system easily collapses. Under the condition $\delta_0 = 0.3$, the food-chain system becomes more stable. For example, when $(\varepsilon,\ \eta)$ is (0.3, 0.7), the brokenness ratio is 0.

According to ecology, the less ε is, the higher η is. Thus, the higher efficiency of energy transfer in the food-chain system means that the species of the food-chain do not easily become extinct. Hence, $(\varepsilon,\ \eta)$ is an important parameter and needs a careful/an appropriate trade-off between system robustness and optimization performance.

3.2 Analysis of neighborhood region δ_0

The parameters of the algorithm are initialized as follows: the length of food-chain is 3; the initial population size is 20; the maximal population size is 40; $(\varepsilon,\ \eta)$ is set to be (0.2,0.8); the maximal generation number T is 2000 and the algorithm repeats 50 times. Taking the Rosenbrock function as an example, we recorded the simulation results in Table 3.

It is shown in Table 3 that the neighborhood region of the Artorgs δ_0 mainly affects the precision and standard deviation of solutions. For example, under the condition $\delta_0 = 0.3$, the best, worst, average, and standard deviations of the solutions are 5.12E–09, 2.34E–05, 1.92E–06, and

Table 3 Analysis of neighborhood region δ_0

δ_0	best	worst	average	standard deviation	average execution time/s
0.3	5.12E–09	2.34E–05	1.92E–06	1.04E–08	3.6091
0.4	2.74E–08	1.45E–05	2.91E–06	5.40E–07	3.7895
0.5	8.76E–08	2.74E–05	5.60E–06	8.06E–07	3.51105
0.6	1.98E–07	1.45E–05	2.41E–06	4.62E–07	3.4304
0.8	1.42E–07	3.38E–05	4.85E–06	4.46E–09	3.0799
1.0	4.81E–07	7.55E–05	1.19E–05	5.03E–07	3.1996

1.04E–08 respectively. However, under the condition $\delta_0 = 1.0$, the best, worst, average, and standard deviations of the solutions are 4.81E–07, 7.55E–05, 1.19E–05, and 5.03E–07 respectively. The neighborhood region of the Artorgs δ_0 does not have much influence on the execution time.

According to ecology, the neighborhood region of Artorgs is very important, where the Artorg moves, seeks food, and produces waste. It determines whether the Artorg has a chance of survival. It also means a collision domain where the Artorgs must avoid colliding with each other. Furthermore, it controls the number of Artorgs in the area, adjusts the pressure of natural selection and evolvement/evolution. Besides, according to the intelligent algorithm, the neighborhood region of Artorgs also influences the capability to search for the optimum solution.

3.3 Analysis of maximal generation number T

Taking the Rosenbrock function as an example, we recorded the simulation results in Table 4.

It is shown in Table 4 that the algorithm has good capability for optimization calculation. For example, under the condition $T = 300$, its optimum solution is 2.04E–04 with an average execution time 0.4526 s. Whereas, under the condition $T = 5000$, its optimum solution is 6.77E–10, with an average execution time 8.992 s.

3.4 Analysis of population size

Taking the Rosenbrock function as an example, we recorded the simulation results in Table 5.

It is shown in Table 5 that the population size of Artorgs is not very helpful for improving the capability of optimization calculation, while its average execution time expends rapidly.

3.5 Analysis of length of food-chain

Taking the Rosenbrock function and the Schaffer function as examples, we recorded the simulation results in Table 6.

It is shown in Table 6 that the length of the food-chain is not very helpful for improving the capability of optimization calculation. However, its average execution time expends rapidly with the increase of food-chain length. According to ecology, given this increase, the number of species also increases, which means that it is more difficult for the artificial food-chain system to harmonize its members. It can be concluded from the results that when the

Table 4 Analysis of food-chain algorithm and maximal generation T

generation number T	best	worst	average	standard deviation	average execution time/s
100	0.003765	2.749888	0.435788	0.094297	0.11715
300	2.04E–04	1.05906	0.128923	0.018282	0.4526
500	6.34E–06	0.05116	0.01024	0.002288	0.79515
800	3.43E–06	0.002777	6.88E–04	2.07E–04	1.3094
1000	6.67E–07	0.217988	0.011003	0.002459	1.47865
1500	7.29E–08	5.20E–04	2.84E–05	5.55E–06	2.3504
3000	5.18E–09	4.49E–06	6.70E–07	1.38E–07	5.0648
5000	6.77E–10	2.08E–07	4.74E–08	7.15E–09	8.992

Table 5 Analysis of food-chain algorithm and population sizes

population size	best	worst	average	standard deviation	average execution time/s
10	8.35E–07	9.07E–05	1.19E–05	1.59E–06	1.39705
20	5.12E–09	2.34E–05	1.92E–06	1.04E–08	3.6091
30	7.42E–09	5.55E–06	5.47E–07	1.15E–07	8.54025
40	3.86E–09	4.83E–06	7.61E–07	1.26E–07	14.8604
50	4.12E–09	7.97E–05	4.36E–06	9.74E–07	22.27055

Table 6 Analysis of food-chain algorithm and its length

function name	length of food-chain	best	worst	average	standard deviation	average execution time/s
Rosenbrock	3	5.12E–09	2.34E–05	1.92E–06	1.04E–08	3.6091
	4	1.86E–08	3.39E–05	3.84E–06	7.98E–07	5.7377
	5	6.26E–08	7.32E–05	6.05E–06	9.43E–07	8.3289
Schaffer	3	−0.999999985	−0.999559803	−0.999945262	7.37E–06	4.6787
	4	−0.999999919	−0.999964601	−0.999989565	2.20E–06	8.48765
	5	−0.999999453	−0.981546115	−0.99906924	2.06E–04	11.14005

length of the food-chain is 3, the proposed algorithm becomes most powerful.

4 Conclusions

1) Considering artificial life characteristics, artificial life emergent colonization and food-chain phenomena in ecology, a new artificial life algorithm called the food-chain algorithm is proposed. It achieves optimization computation by mimicking the evolution of a natural ecosystem and the information processing mechanism of natural organisms. Its energy rules, neighborhood regions and parameters are systemically discussed in detail.

2) The algorithm is designed according to the down-up principle, focusing particularly on its bottom basic rules. The algorithm does not require objective functions to be either continuous or gradient.

3) According to the simulation results, the algorithm performs well in function optimization. It has quasi-life traits such as autonomy, evolution, adaptiveness and robustness, and can solve optimization problems of life-like systems such as supply chain planning and distribution network designing.

4) Although the proposed algorithm has improved the performance of function optimization in most situations, more research is needed to better design the parameters and rules of the food-chain algorithm.

Acknowledgements The work was supported by the National Natural Science Foundation of China (Grant Nos. 70431003, 70571077, 75103012).

References

1. Langton C G. Artificial Life. Redwood: Addison-Wesley, 1989, 1–47
2. Song J D, Yang B S, Choi B G, et al. Optimum design of short journal bearings by enhanced artificial life optimization algorithm. Tribology International, 2005, 38(4): 403–412
3. Menczer F, Belew R K. Adaptive retrieval agents: Internalizing local context and scaling up to the web. Machine Learning, 2000, 39(2–3): 203–242
4. Howard K R. Unjamming traffic with computers. Scientific American, 1997, 277(4): 158–161
5. Rosin C D, Halliday R S, Hart W E, et al. A comparison of global and local search methods in drug docking. In: Proceedings of the 7th International Conference on Genetic Algorithms. Orlando: Morgan Kaufmann, 1997, 221–228
6. Yu H F, Wang D W. Food-chain algorithm and its application to supply-chain operation management problems. Journal of Northeastern University (Natural Science), 2005, 26(1): 25–28 (in Chinese)
7. Zhang E D, Kang A L. Pursue and Escape: Behavioral Ecology. Shanghai: Shanghai scientific and Technical Publishers, 2002 (in Chinese)
8. Dawkins R. The Selfish Gene. Oxford: Oxford University Press, 1996
9. Assad A M, Packard N H. Emergent colonization in an artificial ecology. In: Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life. MIT Press, 1992, 143–152
10. Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs. Berlin: Springer-Verlag. 1992