International Symposium on Nonlinear Theory and its Applications
Xi'an, PRC, October 7–11, 2002

A GA-based synthesis method of self-replication cellular automata

Daisuke Kobayashi and Toshimichi Saito

EECE Dept., Hosei University, Tokyo, 184-8584 Japan
Email: kobayashi@nonlinear.k.hosei.ac.jp, tsaito@k.hosei.ac.jp

Abstract: This paper presents a GA-based synthesis algorithm for ring-type CAs with self-replicating structure. The algorithmble has simpler fitness and shorter genes than previous algorithms. Efficiency of the algorithm is confirmed by basic experiments.

1. Introduction

Cellular Automata (ab.CAs) are discrete dynamical systems which can exhibit various interest phenomenon [1]-[3]. The dynamics of the CAs is governed by a rule-table. In order to synthesize desired CA, an efficient search algorithm of the desired rule table is necessary. Since the number of possible rule tables is incredibly large, direct search of the desired rule is almost impossible.

In this paper, we consider a GA-based synthesis algorithm for ring-type CAs with self-replicating structure. As is well known, the GA is an efficient strategy for searching such extremely large search spaces. In our algorithm, Our algorithm has the following properties:

(1) The self-replicating structure is evaluated with a simple fitness function.

(2) The genes at time t correspond to rules used really until time t. The gene length is time-variant.

That is, our algorithmble has simpler fitness and shorter genes than previous algorithms [4] [5]. Efficiency of the algorithm is confirmed by basic experiments.

2. Cellular Automata

We consider ring-type CAs with N cells. Let the cellular space at time t be presented by

$$a^t = (a_0^t, a_1^t, \cdots, a_{N-1}^t), \tag{1}$$

where a_i^t is a discrete state at time t. For 2-state 3-neighbor CAs, the dynamics is described by

$$a_k^{t+1} = F(a_{k-1}^t, a_k^t, a_{k+1}^t), \quad k \in \{1, 2, \cdots, N\} \pmod N, \tag{2}$$

where F represents the rule table. A k-th state at time $t+1$ is determined by the states of k-th and its both sides cells. The both sides cells are called neighbors. Fig. 1 shows an example of time evolution of CA.

For n-state and m-neighbor CAs, the number of possible rule tables are n^{n^m}. (The number is 256 for 2-state and 3-neighbor CAs.) As size of CAs increases, the size of the search spaces (the number of all possible rule tables) is to be incredibly large.

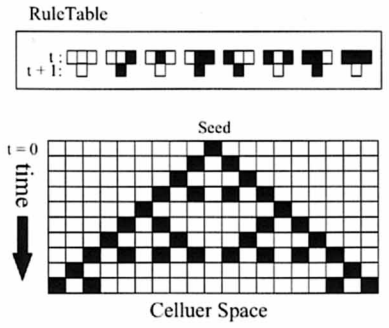

Figure 1: A 2-state 3-neighbor CA

3. Searching Algorithm

First, we present a fitness function for evaluating the self-replicating structure. Let a seed structure is represented by

$$s = (s_0, s_1, \cdots, s_{k-1}) \tag{3}$$

The fitness function is formulated by the following.

$$F(t) = \frac{1}{k} \sum_{l=0}^{k-1} \frac{f_l(t)}{M_l^t \times k} \tag{4}$$

$$f_l(t) = \begin{cases} 0 & \text{if } M_l^t \leq 1 \\ \sum_{i=0}^{N-1} C_1(a_i, s_l) & \text{otherwise} \end{cases}$$

$$C_1(a_i, s_l) = \begin{cases} \sum_{j=i-l}^{i-l+k-1} C_2(a_j^t, s_{j-(i-l)}) & \text{if } a_i^t = s_l \\ 0 & \text{otherwise} \end{cases}$$

$$C_2(a_j^t, s_{j-(i-l)}) = \begin{cases} 1 & \text{if } a_j^t = s_{j-(i-l)} \\ 0 & \text{otherwise} \end{cases}$$

where M_l^t denotes the number of state l in the cellular space at time t. Fig.2 shows an example for $N = 12$. At $t = 10$, we can calculate value of $F(10)$ by $f_0(10) = 5$ and $f_1(10) = 4$.

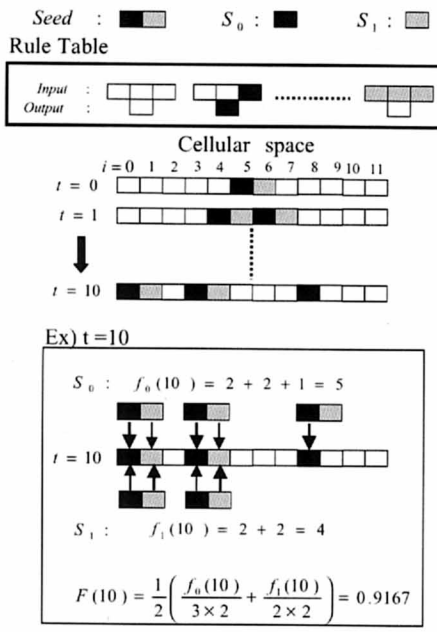

Figure 2: An example of fitness calculation

Next, we explain the correspondence between CA and GA. As a seed is given the cellular space is updated by a rule table. At time t, the rules used until time t determines the gene length. Fig.3 shows a gene at $t = 0$. Then the algorithm is defined following.

The algorithm at $t = 0$ (see fig.4 t=0)

STEP0-1: A seed is given in the cellular space. Let generation be 0.

STEP0-2: It extracts the rules used at $t = 0$. The initial gene is determined by the rules. The number of the extracted rules is 4 in fig.4. This number corresponds to the gene length.

STEP0-3: Find the gene with best fitness.

STEP0-4: The cellular space is updated with the best gene.

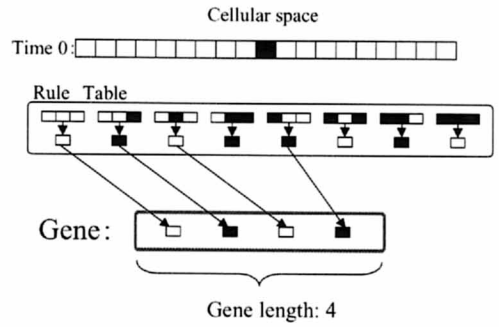

Figure 3: coding

The algorithm at $timet \geq 1$ (see fig.4 t=1).

STEPt-1: It extracts new rules used at time t. The new rules are added to the gene and go to STEPt-3. The number of extracted rules is 6 in Fig.4. In Fig.4, the number of new rules is 2 and the gene length is to be 6.

STEPt-2: Applying GA to the additional part of the genes.

STEPt-3: Cellular space is updated with the best gene. Let $t = t + 1$. If $t = T_{max}$, the program is terminated. Otherwise, go to STEPt-1.

4. Experiment

We have executed the experiments with the following condition.

CA condition:
State : 5
Neighbor : 7
Cellular space : 100
Max time : 30

GA condition: Individuals : 30
Selection : roulette wheel selection
Crossover : uniform crossover
Mutation rate : 0.01
Max generation : 300

Figs. 5 and 6 shows typical experimental results. The comparison of calculation time shows Table.1.

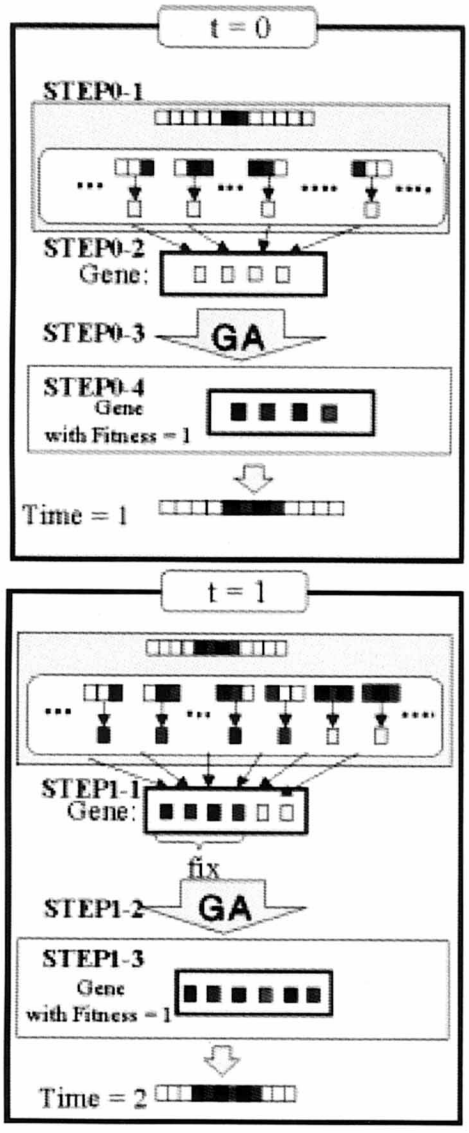

Figure 4: Algorithm flow chart

state - neighbor	New algorithm	Old algorithm [5]
4 - 5	0.098[s]	30.4[s]
5 - 7	1.60[s]	impossible

Table 1: The comparison of calculation time (CPU = AMD K6-2 500MHZ)

5. Conclusion

We have presented a GA-based synthesis algorithm of self-replicating CAs and demonstrated its efficiency. Future problems includes optimal GA parameters settings, simplification of fitness functions and practical applications.

References

[1] S. Wolfram, Universality and Complexity in Cellular Automata, Physica D, 10 pp.1-35 (1984)

[2] H.Gutowitz and C.Langton, Methods for Designing 'Interesting' Cellular Automata, CNLS News Letter (1988)

[3] J.Y.Perrier, M.Sipper and J.Zahnd, Toward a viable, self-reproducing universal computer, Physica D, 87, pp.335-352 (1996)

[4] J.D.Lohn and J.A.Reggia, Automatic Discovery of Self-Replicating Structures in Cellular Automata, IEEE Trans. Evolutionary Computation, 1, 3, pp.165-178 (1997)

[5] H. Kajisha and T. Saito, Synthesis of self-replication cellular automata using genetic algorithms, Proc. of IJCNN, V, pp.173-177 (2000)

[6] Jusitn Werfel, Melanie Mitchell and James P.Crutchfield, Resource Sharing and Coevolution in Evolving Cellular Automata IEEE Trans. Evolutionary Computation, 4, 4, pp.388-393 (2000)

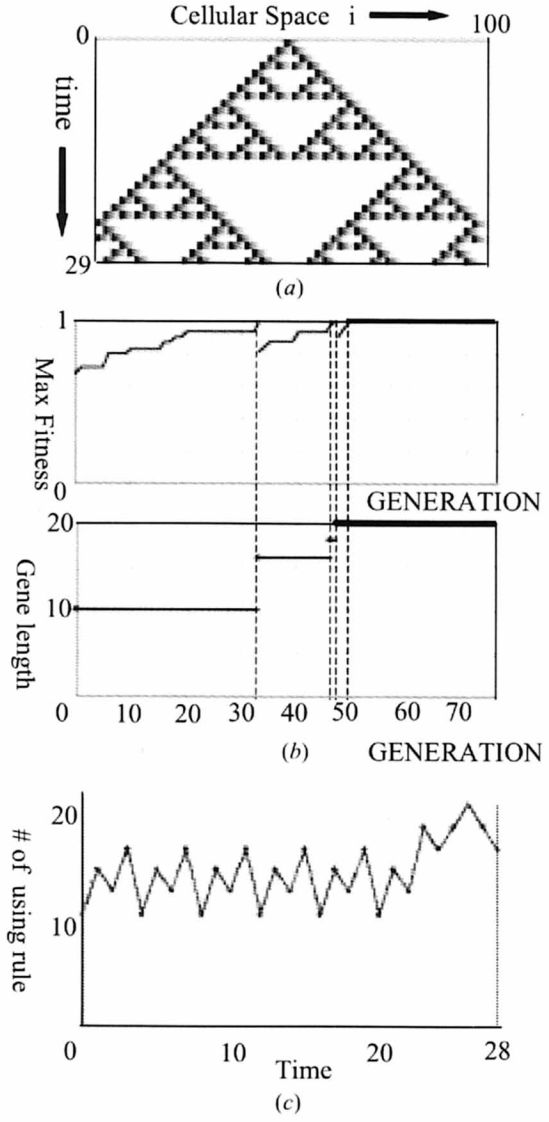

Figure 5: Sierpinski-gasket pattern. (a) Time evolution, (b) The maximum fitness, (c) The number of rules used actually.

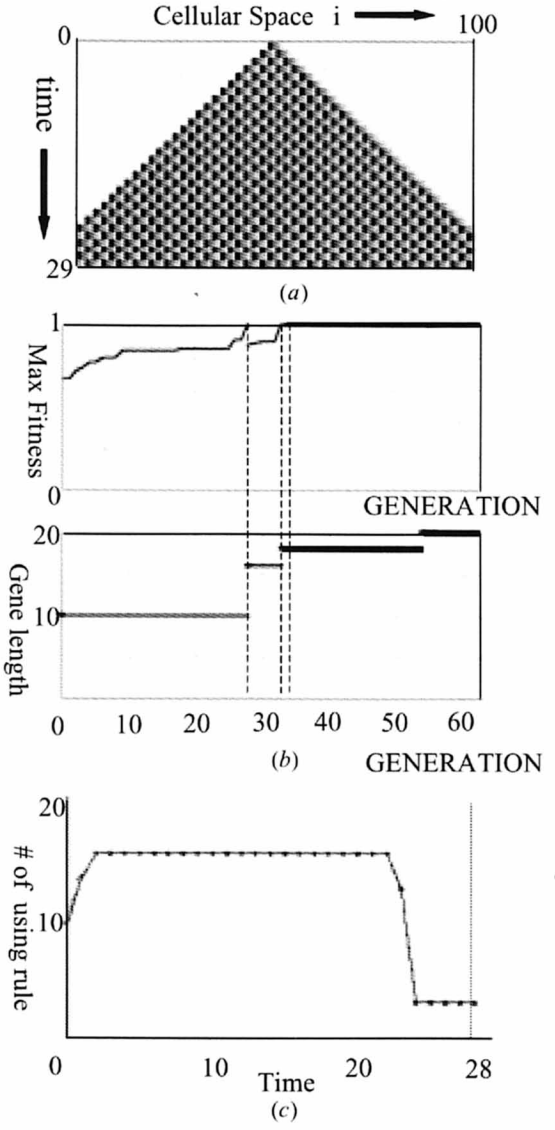

Figure 6: Seed duplicates dubling pattern. (a) Time evolution, (b) The maximum fitness, (c) The number of rules used actually.