Improved Evolutionary Design for Rule-Changing Cellular Automata Based on the Difficulty of Problems

Hitohsi Kanoh, Member, IEEE, and Shohei Sato

Abstract—This paper describes a method to promote the evolution of the transition rules of cellular automata using a genetic algorithm. We previously proposed the evolutionary design of a cellular automaton in which an applied rule changes with time. This method encodes a rule and the number of times the rule is applied as a chromosome. In this paper, we describe the improvement of the method and analyze rules obtained using the Lambda parameter defined by Langton. The difficulty of test problems in an evolutionary process is adjusted so as to obtain a rule which performs the density classification task with high probability. Experiments using ten-thousand randomly generated tasks have shown that the proposed method performs better than the previous method.

I. INTRODUCTION

Evolutionary computation by parallel computers has gained attention as a method for designing complex systems [1]. In particular, parallel computers based on cellular automata (CA) [2,3] offer the advantages of vastly parallel, highly local connections, and have attracted increased research interest [4,5]. However, the difficulty of designing CA transition rules to perform a particular task has severely limited CA applications [6].

The evolutionary design of CA rules has been studied in detail by the EVCA group [3,6-8]. A genetic algorithm (GA) was used to evolve CA for two computational tasks: density classification and synchronization. Those GA were shown to have discovered rules that gave rise to sophisticated emergent computational strategies. Land and Belew [9] proved that no perfect two-state rule for performing a density classification task exists. Fuks [10], however, showed that a pair of human written rules enabled perfect performance of the task.

We earlier proposed the evolutionary design of a CA in which an applied rule changes with time for the two tasks [11]. This method encodes a rule and the number of times the rule is applied as a chromosome. This is to reduce the complexity of a given task by dividing the task into smaller ones and assigning a distinct rule to each sub-task.

In this paper, we explain how the above method and the analysis of obtained rules can be improved by using the Lambda parameter $\lambda$ [12] defined by Langton. The difficulty of test problems in an evolutionary process is adjusted so as to obtain a rule which performs the density classification task with high probability. The role of each of two rules is clarified by dividing the rules into classes.

In the following sections, we first describe the problem to be addressed and provide an outline of our previous method. We then describe how the difficulty of problems can be adjusted and the algorithm of the proposed method. After that, we show the results of experiments comparing the previous and the proposed method and the results of analysis using the Lambda parameter.

II. OVERVIEW

A. Cellular Automata

In this paper we address one-dimensional CA that each consist of a one-dimensional lattice of $N$ cells. Each cell can take one of $k$ possible states. The state of each cell at a given time depends only on its own state at the previous time step and the state of its nearby neighbors at the previous time step according to a transition rule $R$. A neighborhood consists of a cell and its $r$ neighbors on either side. A major factor for CA is how far one cell is from another. The CA rules can be expressed as a rule table that lists each possible neighborhood with its output bit; that is, the update value of the central cell of the neighborhood. Table I shows an example of a rule table when $k = 2$ and $r = 1$. The output bit “11101000” is a binary number that converts to a decimal value of 232, so we denote the rule in Table I as rule 232. Here, we describe CA with a periodic boundary condition.

The behavior of one-dimensional CA is usually displayed as space-time diagrams in which the horizontal axis depicts the configuration at time $t$ and the vertical axis depicts successive time steps. The term “configuration” refers to the collection of local states over the entire lattice, and $S(t)$ denotes a configuration at time $t$.

Langton defined the Lambda parameter to analyze the structure of the CA rule space [12]. This parameter is defined as follows. We pick an arbitrary state, and call it the quiescent state $S_q$. Let there be $N_q$ transitions to this special quiescent state in the possible transitions. Then

$$\lambda = \frac{k^{2r+1} - N_q}{k^{2r+1}}.$$

When $k = 2$ and $S_q = 0$, the Lambda parameter is the fraction of state-1 bits in the output bits for a given rule table.
B. Computational Task for CA

We used the density classification task (DCT) as a benchmark problem [3,6]. The task goal is to find a transition rule that decides whether the initial configuration contains a majority of 1s. Here, \( \rho(t) \) denotes the density of 1s in the configuration at time \( t \). If \( \rho(t) > 0.5 \), then within \( M \) time steps the CA should go to the fixed-point configuration of all 0s; otherwise, within \( M \) time steps it should produce the fixed-point configuration of all 0s (\( \rho(t; M) = 0 \)). The value of constant \( M \) depends on the task.

C. Framework of rule-changing CA

CA in which an applied rule changes with time are called rule-changing CA [11], and a pair of rules and the number of rule iterations can be considered as a step in the computer program. The computation based on the rule-changing CA can thus operate as follows:

Step 1: The input to the computation is encoded as the initial configuration \( S(0) \).

Step 2: Apply rule \( R_i \) to \( S(0); M_i \) times; \( \ldots \); apply rule \( R_n \) to \( S(M_1; \ldots; M_n) \) \( M_n \) times.

Step 3: The output is decoded from the final configuration \( S(M_1; \ldots; M_n) \).

In this case, \( n \) is a parameter that depends on the task, and rule \( R_i \) and the number of rule iterations \( M_i (i = 1, \ldots, n) \) can be obtained by the evolutionary algorithm shown in Fig. 1. Each chromosome in the population represents a candidate set of \( R_i \) and \( M_i \) as shown in Fig. 2. The present method is meant to reduce the complexity of a given task by dividing the task into \( n \) smaller tasks and assigning \( (R_i, M_i) \) to the \( i \)-th task.

D. Related Work

Much research has been done on CA to perform the DCT. Land et al. proved that no perfect two-state rule for performing the density classification task exists [9]. Oleveira et al. reported the best currently known rules when \( r = 3 \) [13]. On the other hand, Packard pointed out that transition rules with a Lambda parameter close to a critical point are frequently obtained [14,15]. Mitchell et al. investigated the distribution of Lambda parameters of the rules that evolved by GA when \( r = 3 \), and they showed that the majority of the rules were concentrated on \( \lambda = 0.43 \) and 0.57 [16].

Much research on a rule-changing CA has also been reported. Fuks proved that the DCT when \( r = 1 \) can be solved perfectly by applying two rules sequentially and he gave two examples [10]. Martins et al. obtained 26 complete solutions consisting of two or three rules using a GA [17].

Recently the evolutionary design of CA has become notable because of its application possibilities: the online partitioning problem [18], the spatial pattern formation [19], and the evolving memory [20] for the usual CA; the modeling of a virtual city [21] and the design of an artificial society [22] for rule-changing CA.

III. PROPOSED METHOD

A. Fitness Function

The fitness of an individual in a population is the fraction of \( N_{test} \) initial configurations (i.e., the test problems) in which the individual produces the correct final configurations. In the previous method [11], the initial configurations are uniformly distributed over \( \rho(0) \in [0,1] \). When these configurations are used, some of the rules randomly generated as an initial population can simply solve the tasks of \( \rho(0) \approx 0 \) or 1, resulting in evolutionary retardation. To improve the method
regarding this point, we let the configurations \( \rho(0) = D/N \) or \( (N-D)/N \) be the test problems in this study, where \( D \) is an experimentally determined parameter and \( N \) is the lattice size. In this method, the difficulty of test problems can be adjusted through the value of \( D \). Figure 3 shows the distribution of test problems generated by the proposed method. The horizontal and the vertical axes respectively represent density \( \rho(0) \) and the number of problems, where \( N_{test} = 100, N = 149, \) and \( D = 70 \).

**B. Algorithm**

First, generate \( N_{pop} \) individuals as an initial population. The following operations are then repeated for \( N_{gen} \) generations.

Step 1: Generate a new set of \( N_{test} \) initial configurations described in III-A, and calculate the fitness on this set for each individual in the population.

Step 2: Rank the individuals in order of fitness, and the top \( N_{elit} \) elite individuals are copied to the next generation without modification.

Step 3: \( N_{mutation} \) individuals are selected from the population by roulette wheel selection. They are mutated and added to the next generation. The rules \( R_1, \ldots, R_n \) are each mutated at exactly two randomly chosen positions. The numbers of iterations \( M_1, \ldots, M_n \) are mutated with probability 0.5 by substituting random numbers subject to \( M_1 + \ldots + M_n = M \).

Step 4: The remaining \( (N_{pop} - N_{elit} - N_{mutation}) \) individuals for the next generation are formed by \( n \)-point crossovers between the individuals randomly selected from \( N_{elit} \) elites and the individuals selected from the whole population by roulette wheel selection, so that the rules \( R_1, \ldots, R_n \) on the parents each undergo crossover just one time.

Step 5: The rules on the offspring from crossover are mutated by the same method as in step 3.

**IV. EXPERIMENTS**

**A. Experimental method**

The experiments were done under the following conditions: number of states \( k = 2 \), lattice size \( N = 149 \), number of neighborhoods \( r = 3 \), and the upper limit of time steps \( M = N \) for CA; \( N_{pop} = 100, N_{elit} = 20, \) and \( N_{mutation} = 20 \) for GA. The other conditions will be shown in the following sections. In addition, each experiment was repeated for 50 trials under the same conditions but using a different random number seed.

**B. Evaluation of the proposed method**

The performance of the best rules obtained was evaluated on a \( 10^4 \) randomly generated initial configuration. They conformed to a binomial distribution with a peak at \( \rho(0) = 0.5 \) since each bit is independently drawn with probability 0.5 of being 1. This problem is obviously more difficult for CA than the test problem used during evolution. Figure 4 shows the relationship between the parameter of difficulty \( D \) and the performance of the proposed and the previous [11] methods, where \( n = 2, N_{gen} = 200, \) and \( N_{test} = 300 \). We can see that the proposed method performs better the previous method in the range of \( D = 69 \) to 72 and has a maximum value of 82.7 at \( D = 70 \).

**C. Role of rules**

To clarify the effect of each rule on an individual, we classified obtained rules into five groups using the Lambda parameter [23]. In this section, the experiments were done under the conditions \( N_{gen} = 100 \) and \( N_{test} = 100 \). The first experiment, where \( n = 1 \), gave Lambda values of the top 20 elite individuals in the final generation. Figure 5 shows the frequency distribution of 1000 Lambda values obtained in 50 trials. In this figure, ranges \( A \) to \( E \) can be defined as shown in Table II.

Most of the rules in Fig. 5 are distributed in ranges \( B \) and \( D \), and there are peaks at \( \lambda = 0.42 \) and 0.57. These results agree with those of Mitchell et al. [16]. In the following, we assume...
that the rule contained in the range of A or E is a simple rule, one in B or D is a complex rule, and one in C is a random rule (Table II).

The same experiment was done for the rule-changing CA of \( n = 2 \). Figures 6 and 7 show the frequency distribution of the Lambda values of \( R_1 \) and \( R_2 \), respectively. We can see that \( R_1 \) peaks at \( \lambda = 0.42 \) and 0.60. These values are about the same as for the case of \( n = 1 \), but the other rules are more widely distributed than in that case. On the other hand, \( R_2 \) peaks at \( \lambda = 0.32 \) and 0.67, but most of the rules are simple rules.

Figure 8 shows the correlation between the Lambda values of \( R_1 \) and \( R_2 \) for the same individuals in Figs. 6 and 7. Each point in this figure corresponds to one individual. Ranges A to E are also indicated in this figure. We can see that at least one of \( R_1 \) and \( R_2 \) is a complex rule for most individuals, but there are some individuals for which both are simple rules. This means that complex tasks can be solved by combining simple rules only. This is why the rule-changing CA is more efficient than the usual CA, since simple rules can be easily obtained.

CA in which \( R_1 \) is included in A and \( R_2 \) is included in E are denoted as \( A+E \) CA. Figure 9 shows examples of space-time diagrams of \( A+A, A+E, E+A, \) and \( E+E \) CA. The chromosomes and the Lambda values of the CA are also shown.

V. CONCLUSION

In this paper, we proposed a method to promote the evolution of CA rules by adjusting the difficulty of the test problem. The proposed method can be widely used in evolving CA: it has a simple algorithm and can be combined with various evolutionary algorithms. The evolutionary design of rule-changing CA may become difficult if the
The number of rules increases, though it has been proven that the calculation capability of rule-changing CA is high. We think that this method is effective especially in such a case. Rule-changing CA have been applied to the modeling of virtual cities and the design of artificial societies. In our future work, we will apply the proposed method to such problems.

**ACKNOWLEDGMENT**

The authors would like to thank Dr. Yun Wu for letting us use her source code. The authors would also like to thank Daisuke Ichiba for his helpful comment and suggestions.

**REFERENCES**


