Evolutionary Design of Rule-Changing Cellular Automata guided by Parameter indicating Propagation of Information

Shohei Sato and Hitoshi Kanoh
Department of Computer Science, Graduate School of
Systems and Information Engineering, University of Tsukuba
Tsukuba, Ibaraki, 305-8573, Japan
email: s_shohei@kslab.cs.tsukuba.ac.jp

***Abstract*— A method for designing the transition rules of cellular automata using genetic algorithms is described. Rule-changing cellular automata are expected to perform density classification tasks more effectively than ordinary cellular automata. We propose a method for designing high performance rule-changing cellular automata. This method uses a new parameter that indicates the propagation of information. Experimental results for density classification tasks show that the proposed method performs better than the previous method.**

I. Introduction

Cellular automata (CA) are discrete computational models that consist of grids of cells. At each time step, all cells synchronously update their states in accordance with a transition rule. CA have been used for diverse applications because of an emergent property: simple rules can generate complex behavior. CA with desired properties can be obtained by searching transition rules. In particular, the evolutionary design of CA rules has been studied [2, 8, 9, 14]. A genetic algorithm (GA) was used to evolve CA rules for computational tasks, such as Density Classification Tasks, as benchmark problems.

Wu et al. proposed an evolutionary design of rule-changing CA in which multiple rules are applied sequentially. This reduces the complexity of a given task by dividing the task into sub-tasks and assigning a distinct rule to each one. However, in this expanded model, the size of the search space is larger than for an ordinary CA. Thus, it is difficult to develop high performance solutions for density classification by using this method. To address this, we use a method that was proposed by Oliveira et al. [10-13], which we call here the "Parameter Guided Search". In this method, CA parameters that are calculated directly from the transition rule are incorporated into a fitness function to reduce the search space. At the same time, we define a new parameter that indicates "propagation of information", which we will describe later, and apply it to the Parameter Guided Search to improve the searching performance.

In the following sections, we first give an overview of related works and the problem to be addressed. We then describe the method for designing the rule-changing CA and define the new CA parameter *NC*. After that, we show the experimental results on density classification, comparing the previous and the proposed methods.

II. Overviews

A. Cellular Automata

In this paper, we address one-dimensional two state CA that each consist of a one-dimensional lattice of length L. Each cell can take one of two possible states: state 0 and state 1. The state of the i-th cell at a given time t is denoted by s_i^t, and the configuration for the state of all cells is denoted by S_t. The state of a cell at time t+1 depends only on its own state and the state of its nearby neighbors at time t, in accordance with transition rule Φ (Eq. (1)). A neighborhood consists of a cell and its r neighbors on either side, a total of 2r+1 cells. Boundary conditions are required to apply the transition rule at the edge of the lattice. In this case, the periodic boundary condition $s_i^t = s_{i+L}^t$ is used, the same as in previous studies.

$$s_i^{t+1} = \Phi(s_{i-r}^t, s_{i-r+1}^t, \ldots, s_i^t, \ldots, s_{i+r-1}^t, s_{i+r}^t) \quad (1)$$

The transition rule Φ can be expressed as a rule table that lists each possible neighborhood state (s_{i-1}^t, s_i^t, s_{i+1}^t) with its output bits s_i^{t+1}. Table I shows an example of a rule table when r = 1.

B. Density Classification Task

The density classification task (DCT) is one of the most studied benchmark problems for computations with CA. The goal of the DCT is to find a transition rule that determines whether or not the initial configuration S_0 contains a majority of state 1s. The CA must converge to a final configuration S_m of all cells in state 1, when the initial configuration has more state 1s than state 0s, and vice versa.

TABLE I.
EXAMPLE OF RULE TABLE ($r = 1$)

Neighborhood:	000	001	010	011	100	101	110	111
Output bit:	0	0	0	1	1	1	0	1

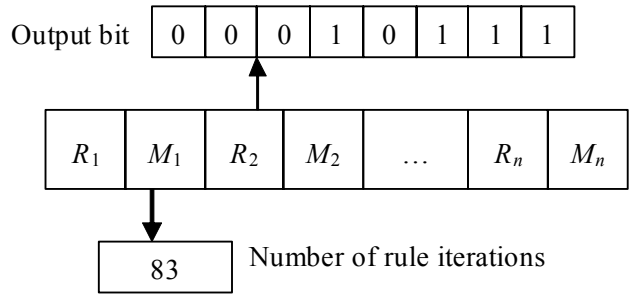

Fig. 1. Example of chromosome of rule-changing CA.

Expression (2) is the definition of DCT, of which ρ_0 denotes the density of state 1s in the initial configuration.

$$\forall i, s_i^t = \begin{cases} 0 & \text{if}(\rho_0 > 0.5) \\ 1 & \text{if}(\rho_0 < 0.5) \end{cases} \quad \text{when} \quad t \geq M \qquad (2)$$

Although solving the DCT is trivial for a von Neumann computer, it is difficult for CA due to the locality of the transition rule and the absence of a memory device in each cell. It requires the emergence of collective behavior to solve, hence is a benchmark for problem solving by CA and an example of emergent computation.

Each rule is evaluated on the percentage of correct answers by testing K randomly generated initial configurations (i.e., the test problems). The efficiency of the designed rules is evaluated for 10^4 initial configurations, which is called the "performance" of the rule here.

C. Rule-changing Cellular Automata

CA in which an applied rule changes with time are called rule-changing CA [6]. The computation based on the rule-changing CA can thus operate as follows:

Step 1: Assign the initial configuration S_0 to all cells.
Step 2: Apply rule R_i to the current configuration for M_i times.
Step 3: Iterate Step 2 for every n rule.

The evolutionary design of the rule-changing CA has been studied by Wu et al. [6], in which an array of the rule R_i and the number of rule iterations M_i were encoded as a chromosome (Fig. 1). Here, M_i denotes the upper limit of the total number of rule iterations. The fitness of an individual in a population f is the fraction of the test problems in which the individual produces the correct final configuration. Test problems are newly created at each generation.

In this method, there have been two issues in the test problems generated for the calculation of f. (1) The initial configurations are uniformly distributed over ρ_0 [0, 1]. Most of these configurations are easy to solve for advanced rules because initial configurations are difficult only in the vicinity of $\rho_0 = 0.5$. As a result, evolution plateaus at relatively low performance. (2) The number of test problems are so few (K = 100) that the fitness fluctuates and degeneration is caused. If the number of test problems is increased, however, so does the computational cost.

D. Parameter Guided Search

To help forecast the dynamic behavior of CA, several parameters have been proposed. The λ parameter defined by Langton is one of the most known examples of CA parameters. Other parameters referred to in this paper are sensitivity (μ) by Binder [1] and the following three parameters by Oliveira et al.: absolute activity (AA), neighborhood dominance (ND), and activity propagation (AP) [10].

The Parameter Guided Search was proposed by Oliveira and is based on reducing search space by forecasting whether or not a transition rule has the desired properties from its parameter value. In this method, evolutionary search is carried out so that both f and Fp increase. A function Fp returns a higher value when the parameters of a transition rule fit more within the predetermined ranges. These parameter ranges are set so that rules with high performance are more likely to occur, on the basis of previous knowledge or a result of the preliminary search. In Ref. [5], the weighted sum of f and Fp was used as a fitness function. Then Oliveira et al. adopted a multiobjective optimization approach, and the searching performance was improved.

III. PROPOSED METHOD

In the proposed method, we adopt the Parameter Guided Search to design the rule-changing CA, in which NSGA-II [3] is used to optimize both f and Fp, the same as Oliveira's method except not using biased genetic operations. We also improve the foregoing issues of the test problems. As a measure against the former, we earlier proposed a method in which the difficulty of the test problems is adjusted [5]. The number of state 1 in the initial configuration D is adjusted so

Step. 1 1 Generate offspring population of size N, and calculate fitness f on a new set of K /2 initial configurations.
Step. 2 Parent and offspring population is combined and sorted by f.
Step. 3 Top N individuals are evaluated again on additional K /2 initial configurations if ($K_{sum} < K_{max}$).
Step. 4 Fitness of all individuals is recalculated as $f = K_{correct} / K_{sum}$
Step. 5 Preserve N elites as a new population based on the NSGA-II algorithm.

Fig. 2. Assignment process of fitness values used in proposed method.

TABLE II.
RULE TABLE OF RULE 170 OF ELEMENTARY CA

Neighborhood:	000	001	010	011	100	101	110	111
Output bit:	0	1	0	1	0	1	0	1

as to obtain high-performance individuals. Experimental results showed that the rule obtained by using this method has a higher performance than that obtained using the previous method.

To improve the latter issue, we apply the algorithm shown in Fig. 2 to evaluate the individuals continuously through generations. In this case, N is the size of the population. All individuals should remember the number of test problems K_{max} and problems correctly classified $K_{correct}$ until the current generation. The variance of fitness of superior individuals in the population can be reduced without increasing calculation cost by using this method.

Global information about density is required to solve the DCT correctly. However, the transition process of each cell depends only on local information. Previous research shows that there exist rules that form global interaction known as "embedded particle" computation [2]. Elementary CA rule 170 is a simpler example of interaction between distant cells. This rule depends only on the state of the right cell as shown in Table II. It coincides with the left shift mapping of the space-time diagram example shown in Fig. 3. In this rule, local information of cells is propagated with time hence can be read by a distant cell. Here we call this space-time behavior "propagation of information". While rule 170 conforms to this behavior, it cannot perform the DCT at all. Thus, it requires the appropriate occurrence of propagation of information to be indicated. We define the NC as a measure of the degree of propagation of information. The NC is the correlation of states between the output bit and one of the neighborhood cells. The neighborhood cell is determined so that the absolute value of the correlation coefficient becomes the maximum. The NC is the normalized version of this value between 0 and 1. Actually, the correlation coefficient is calculated for each state s, and then the NC_s are given individually. The NC_s is defined as Eq. (3), where v shows a

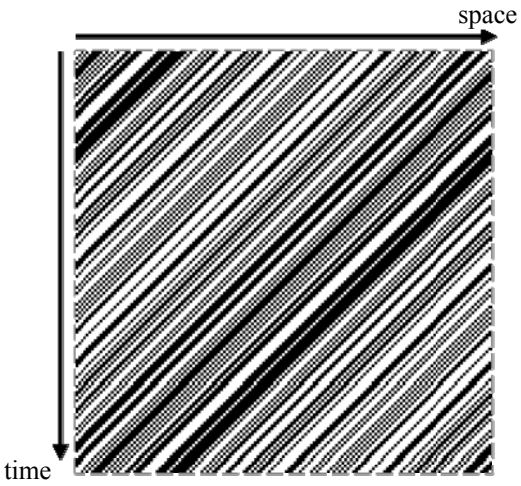

Fig. 3. Example of space-time diagram of rule 170.

neighborhood state and V denotes the set of all neighborhoods. The function $F(p)$ adopted in the equation returns 1 if logical expression p is true, but otherwise returns 0.

$$
\begin{aligned}
NC_s &= nc_s(q_{\max}) \qquad (3)\\
q_{\max} &= \arg\max_q |\, nc_s(q) - 0.5 \,|\\
nc_s(q) &= \frac{1}{2^{2r}} \sum_{v \in V} F(v_q = s \wedge \\
&\qquad \Phi(v_1,...,v_q,...,v_{2r+1}) = v_q)\\
V &= \{v \mid v = (v_1,...,v_{2r+1}), v_i \in \{0,1\}\}\\
F(p) &= \begin{cases} 0 & \text{if}(p \text{ is True}) \\ 1 & \text{if}(p \text{ is False}) \end{cases}
\end{aligned}
$$

IV. EXPERIMENTS

A. Experimental Method

Two experiments were conducted where CA Rules were evolved to perform the DCT. Each experiment was repeated for 50 trials under the same conditions but using a different random number seed. In the first experiment, we tested the efficacy of using the Parameter Guide and of improving the test problems by comparing the four search methods as follows. The specifications of the GA and CA environment

TABLE III.
SPECIFICATIONS FOR FIRST EXPERIMENT

CA	Number of rules	2
	Radius of neighborhood: r	3
	Number of rule iterations: M	300
	Lattice size: L	149
Test problems	Size of test problems: K	200
	Upper limit of K: K_{max}	1000
	Number of state 1s in initial configuration: D	70
GA	Number of individuals: N	100
	Number of generations	200
	Mutation rate	2% per bit

are presented in Table III.

Previous: The method proposed by Wu et al. in [6].

Test problems: The method improving the test problems upon Previous.

Parameters: (μ, AA, ND, and AP) were used in this method. The method introducing the Parameter Guided Search upon Previous. Four parameters

Proposed1: This method combines Test problems and Parameters.

The second experiment was conducted to demonstrate the effectiveness of using the NC by comparing the three search methods as follows. In the Proposed2, the NC was used instead of the AA and other conditions were the same.

Previous: Same as the first experiment.

Proposed1: Same as the first experiment.

Proposed2: The method introducing the NC upon Proposed 1. Five parameters (μ, ND, AP, NC_0, and NC_1) were used in this method.

In this case, the same conditions were used as in the first experiment, except $N_{pop} = 200$ and $N_{gen} = 1000$.

Oliveira defined the parameter ranges by the values of the best rules, found in previous research [10, 12, 13]. However, no such solution is known in the case of the rule-changing CA. Therefore, we adopted the method presented in Ref. [11]. We first ran the GA without the Parameter Guided Search, then the parameter ranges were calculated for the best 50 rules found. This search was conducted with the same conditions as the experiment in Ref. [5]. The parameter ranges of the NC_0 and the NC_1 were defined empirically. The value ranges defined are presented in Table IV.

TABLE IV.
PARAMETER RANGES USED IN EXPERIMENT

CA Parameter	Range
μ	0.27 to 0.36
AA	0.72 to 0.86
ND	0.13 to 0.18
AP	0.44 to 0.55
NC_0	0.10 to 0.40
NC_1	0.10 to 0.40

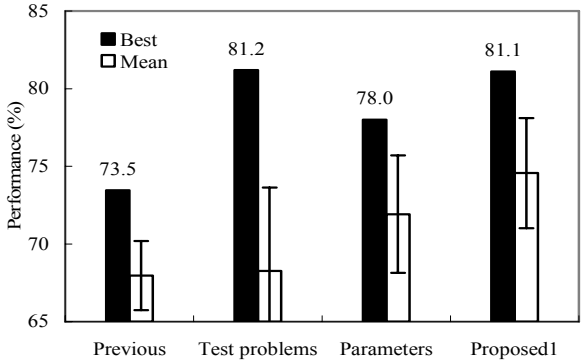

Fig. 4. Best and average performances of obtained rules in first experiment.

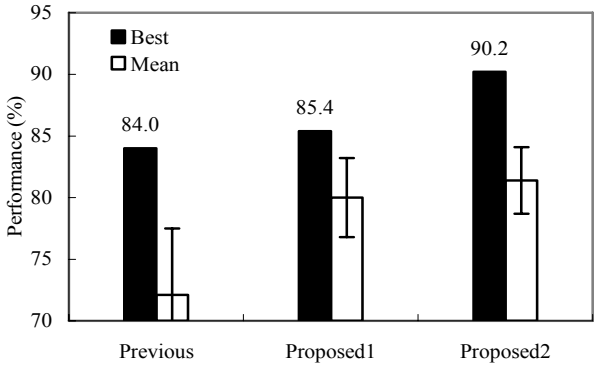

Fig. 5. Best and average performances of obtained rules in second experiment.

B. *Experimental results*

The efficiency of each method was measured by the performance of the best individual found. The results of the first experiment are shown in Fig. 4, where the vertical axis corresponds to the best and average performances and the vertical bars indicate the standard deviation of the average. We can see that the improvement of the test problems provides a rule with higher performance (81.2%) although the

TABLE V.
DISTRIBUTION OF PERFORMANCE ACHIEVED BY PREVIOUS METHODS AND PROPOSED METHOD

Approach	CA (Parameter Guided Search [12])	CA (Co-evolutionary GA [15])	Rule-changing CA (Proposed2)
over 80%	0	27	84
70% to 80%	7	20	16
60% to 70%	89	53	0
under 60%	4	0	0

TABLE VI.
PERFORMANCE OF BEST RULE OBTAINED BY PREVIOUS METHODS AND PROPOSED METHOD

Approach	CA (Parameter Guided Search [13])	CA (Co-evolutionary GA [4])	Rule-changing CA (Proposed2)
Performance	86.2%	86.1%	90.2%

average performance was not much different from the previous method. On the other hand, the average performance improved when the Parameter Guided Search was adopted. We consider that Proposed1, in which both of the improvement of test problems and the Parameter Guided Search are used, is the most efficient in terms of both the best and average performance.

The results of the second experiment are shown in Fig. 5. Proposed1 shows a higher performance than the previous method, as in the first experiment. Proposed2 provides the highest searching performance among these methods. The effectiveness of the *NC* was confirmed by this result.

Last, we compared the calculation capability of the rule-changing CA to ordinary CA. The performance distribution of obtained rules is presented in Table V. The first and second columns show the results of ordinary CA in previous works [12, 15], and the third column shows the results of the rule-changing CA obtained by Proposed2. While different methods were used, the computational effort of each method was equal in all the experiments. We observed that the rules with high performance (80%) appeared more frequently in rule-changing CA. The performance of the best rules known is shown in Table VI. In this case, the computational effort of Proposed2 is less than that of the other methods. The best individual obtained in the proposed method has a performance of 90.2%, although the best CA rules have performances of about 86%. The best individual obtained by the proposed method is shown in Table VII, and

TABLE VII.
BEST RULE OBTAINED BY PROPOSED2 FOR DCT

R_i	M_i
BF80BFF0BFFCFEA0BF8080008000FA00	156
05343551004500775E054622500221ED	144

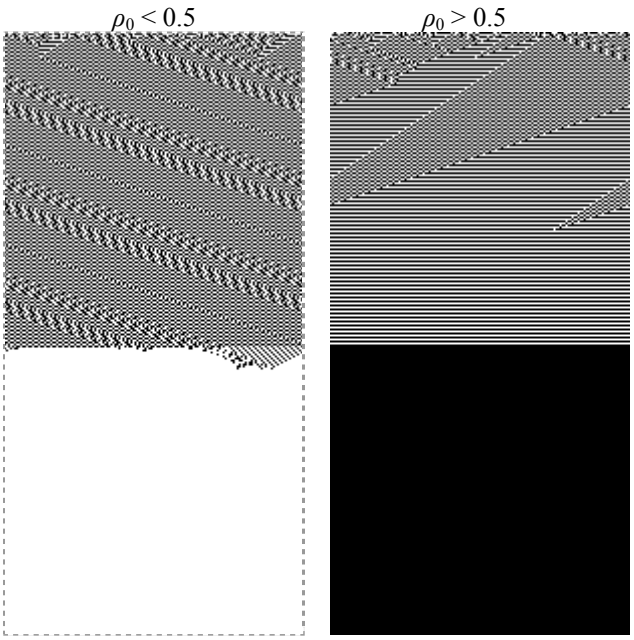

Fig. 6. Examples of space-time diagrams of best (90.2%) rule found.

examples of the space-time diagrams for this are shown in Fig. 6. The rules in Table VII have been converted to hexadecimal, the same as in previous studies [2, 6, 10].

V. CONCLUSION

We proposed a method to promote the evolution of rule-changing CA by using a new parameter *NC*. The experimental results showed that the proposed method has higher searching performance than previous methods and has the advantage of using rule-changing CA for density classification. In our future work, we will extend the proposed method to two-dimensional CA to apply to practical problems.

ACKNOWLEDGEMENT

The authors wish to thank Dr. Wu Yun and Mr. Daisuke Ichiba for their considerable advice with rule-changing CA. The authors are also deeply indebted to Mr. Souichi Tsukahara for valuable assistance in implementation of

computer programs. This research was partly supported by Grant-in-Aid for Scientific Research by the Ministry of Education, Culture, Sports, Science and Technology (18500105).

REFERENCES

[1] P. M. Binder: A Phase Diagram for Elementary Cellular Automata, *Complex Systems*, Vol. 7, pp. 241-247 (1993)

[2] J. P. Crutchfield, M. Mitchell, R. Das: Evolutionary Design of Collective Computation in Cellular Automata, In "*Evolutionary Dynamics: Exploring the Interplay of Selection, Accident, Neutrality, and Function*", Oxford University Press, pp. 361-411 (2003)

[3] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp. 182-197 (2002)

[4] H. Juille, J. B. Pollack: Coevolving the "Ideal" Trainer: Application to the Discovery of Cellular Automata Rules. *Proc. of 5th Annual Conference on Evolutionary Programming*, pp. 461-468, MIT press (1998)

[5] H. Kanoh, S. Sato: Improved Evolutionary Design for Rule-Changing Cellular Automata Based on the Difficulty of Problems, *IEEE International Conference on Systems, Man and Cybernetics*, pp. 1243-1248 (2007)

[6] H. Kanoh, Y. Wu: Evolutionary Design of Rule Changing Cellular Automata, *Proc. of International Conference on Knowledge-Based Intelligent Information and Engineering Systems*, LNAI 2773, pp. 258-264 (2003)

[7] C. G. Langton, "Computation at the Edge of Chaos: Phase Transitions and Emergent Computation," *Physica* D, 42, pp. 12-37, 1990.

[8] M. Mitchell, P. T. Hraber, J. P. Crutchfield: Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations, *Complex Systems*, Vol. 7, pp. 89-130 (1993)

[9] M. Mitchell, J. P. Crutchfield, P. T. Hraber: Evolving Cellular Automata to Perform Computations: Mechanisms and Impediments, *Physica* D, Vol. 75, pp. 361-391 (1994)

[10] G. M. B. De Oliveira, P. P. B. De Oliveira, N. Omar: Definition and Application of a Five-parameter Characterization of One-dimensional Cellular Automata Rule Space, *Artificial Life*, Vol. 7, pp. 277-301 (2001)

[11] G. M. B. De Oliveira, P. P. B. De Oliveira, N. Omar: Searching for One-Dimensional Cellular Automata in the Absence of a priori Information, *Proc. of ECAL 2001*, LNCS 2159, pp. 262-271 (2001)

[12] P. P. B. De Oliveira, J. C. Bortot, G. M. B. De Oliveira: Multiobjective Evolutionary Search for One-Dimensional Cellular Automata in the Density Classification Task, *Artificial Life*, Vol. 8, pp. 202-206 (2002)

[13] P. P. B. De Oliveira, J. C. Bortot, G. M. B. De Oliveira: The Best Currently Known Class of Dynamically Equivalent Cellular Automata Rules for Density Classification, *Neurocomputing*, Vol. 70, pp. 35-43 (2006)

[14] N. H. Packard: Adaptation Toward the Edge of Chaos, In "*Dynamic Patterns in Complex Systems*", World Scientific, Singapore, pp. 293-301 (1988)

[15] J. Werfel, M. Mitchell, J. P. Crutchfield: Resource Sharing and Coevolution in Evolving Cellular Automata, *IEEE Transaction on Evolutionary Computation*, Vol. 4. No. 4, pp. 388-393 (2000)