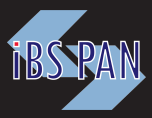

INSTYTUT BADAŃ SYSTEMOWYCH
POLSKIEJ AKADEMII NAUK

TECHNIKI INFORMACYJNE TEORIA I ZASTOSOWANIA

Wybrane problemy
Tom 4 (16)

poprzednio

ANALIZA SYSTEMOWA W FINANSACH
I ZARZĄDZANIU

Pod redakcją
Andrzeja MYŚLIŃSKIEGO

Warszawa 2014

INSTYTUT BADAŃ SYSTEMOWYCH
POLSKIEJ AKADEMII NAUK

TECHNIKI INFORMACYJNE TEORIA I ZASTOSOWANIA

Wybrane problemy
Tom 4 (16)

poprzednio

ANALIZA SYSTEMOWA W FINANSACH I ZARZĄDZANIU

Pod redakcją
Andrzeja Myślińskiego

Warszawa 2014

Wykaz opiniodawców artykułów zamieszczonych
w niniejszym tomie:

Prof. Bernard De BAETS

Dr hab. Ewa BEDNARCZUK, prof. PAN

Dr hab. inż. Wiesław KRAJEWSKI, prof. PAN

Dr hab. inż. Andrzej MYŚLIŃSKI, prof. PAN

Dr inż. Jan W. OWSIŃSKI

Dr hab. Dominik ŚLĘZAK, prof. UW

Prof. dr hab. inż. Andrzej STRASZAK

Prof. dr hab. inż. Stanisław WALUKIEWICZ

ISBN 83-894-7555-3

SOLVING THE DENSITY CLASSIFICATION TASK WITH THE USE OF THE CONTINUOUS CELLULAR AUTOMATA

Witold Bołt[1,2]

[1] Systems Research Institute, Polish Academy of Sciences,
Ph. D. Studies, Warsaw, Poland
[2] Department of Mathematical Modelling, Statistics and Bioinformatics, KERMIT,
Ghent University, Ghent, Belgium
wbolt@ibspan.waw.pl

Abstract. The Density Classification Task (DCT) is one of the most studied problems in the context of Cellular Automata (CAs) computation capabilities. In this paper we propose a novel, relaxed variant of this task, namely the α–DCT, which could be solved with the use of the Continuous Cellular Automata (CCAs). The paper is accompanied with the presentation of the results of simulations utilizing evolutionary algorithm for constructing CCA–based solutions of the α–DCT.
Keywords: cellular automata, density classification task

1 INTRODUCTION

Cellular Automata introduced in [22], form a widely studied and exploited class of discrete dynamical systems. The simplicity of CA–based models originating from local–only interactions, finite state space and the lack of memory, makes them really attractive when it comes to practical applications. Numerous models simulating many natural phenomenons, based on CAs, were introduced through the years [6].

Aside from applications to problems in mathematical modeling of natural systems, CAs are often studied within the field of information and computation theory, as one of the models of parallel computation. One of the most often–studied problems in this field is the DCT. The problem was initially implicitly defined in [12] and then formally presented in [19]. The formulation of DCT is straight forward. Given a final lattice, filled with 0s and 1s, the desired goal of the system is to evolve to a homogenous state (all cell in the lattice being in the same state), that corresponds to one of the states that was initially assigned to the majority of cells. Such a task could be trivially solved by many computation models with global memory, but since CAs exhibit only local interactions, the task becomes non–trivial.

Many attempts were made to construct CAs that could solve the problem. Most notably various techniques utilizing evolutionary algorithms for

discovery of rules solving the DCT were proposed [7], which resulted in the development of relatively effective methods for the automatic construction of the rules of CAs with different properties. Further examination shown that the DCT problem could not be solved using classical CAs [17]. Although currently proven only for 1D and 2D CAs, it is believed [18] that there is no CA solving the DCT in any of the dimensions, no matter how many (finite) states are available.

Knowing that the general problem, for classically defined CAs cannot be solved, the further research activities were divided into two groups. The first one relates to the imperfect solutions of the DCT, *i.e.* rules that could solve the task for the highest possible number of initial configurations. It is not yet known what is the upper–limit for the classification correctness. Best rules known so–far are able to correctly classify up to 90% of initial configurations under consideration, but even the method of comparing the efficiency and correctness of different rules is still a topic for a scientific discussion [18].

The second problem group under active investigation, relates to the DCT in different, relaxed settings, where some of the constrains of the classical CAs are altered. Approaches studied so far include: rule changing CAs [16], programmable CAs [20], asynchronous CAs [14], different types of multi–state rules [11], CAs with memory [21], discretized continuous CAs [5], as well as Stochastic CAs (SCAs) and stochastic mixtures of deterministic CAs [8, 10]. Depending on the setting, the DCT problem might remain unsolvable, but in some cases the situation differs compared to the classical formulation. For example in the stochastic setting, it is know that there exist rules that can solve the problem with any, arbitrary precision [8], with the cost of high accuracy being reflected in long computing times.

In this paper we follow the later of the two presented research paths. A new kind of relaxed DCT formation along with an alternative CA–inspired computation model is presented. The model used here, namely the CCAs, is the natural extension of binary (2–state) CAs. In addition to presenting the initial definition of the novel α–DCT problem, we present a summary of experiments based on evolutionary computing, with the goal of finding a CCA rule solving the α–DCT.

The paper is organized as follows. In Section 2 we formally introduce the concept of the CA, which is further extended in Section 3 to the context of the continuous state space. In Section 4 we present the definition of the α–DCT problem and discuss some of its formal properties. After that, in

Section 5, we describe the Genetic Algorithm (GA) used to find candidate solutions of the α–DCT. The paper is concluded with Section 6, where the results of running the GA are presented. A brief summary is presented in Section 7.

2 PRELIMINARIES

We start by formulating the definition of a CA. In this paper, by a CA we understand a 1D, discrete (both in time and space) dynamical system with finite states and a local, deterministic transition function. Due to discrete nature, the space of such a system is divided into discrete elements referred to as *cells*. We consider only the systems with a finite number of cells. Let $\mathcal{C}$ denote the set of all cells, and $N = |\mathcal{C}|$. Normally, we will presume that cells are ordered and thus we will treat the set $\mathcal{C}$ as a sequence $\mathcal{C} = (c_i)_{i=1}^{N}$.

For each cell we define a neighborhood consisting of "nearby" cells. We consider only symmetric neighborhoods, therefore the cells included in the neighborhood can be expressed with the use of radius $r \geq 0$. Radius indicates how many cells belong to the neighborhood on either side of the given cell (either left or right). For a given r, the neighborhood contains $2\,r + 1$ cells (including the cell for which the neighborhood is defined). Typically, it is desired to satisfy that $r \ll N$, but formally it is only required that $r < N/2$. When the number of cells is finite (as it is in our case), we assume that the cells are placed on a circle, so that for each cell the neighborhood can be easily defined. For convenience, we assume that for any $i \in \mathbb{Z}$ we have: $c_i = c_{i+N} = c_{i-N}$, and thus for any cell c_i the neighborhood is defined as: $\mathcal{N}(c_i) = (c_{i-r}, c_{i-r+1}, \ldots, c_i, \ldots, c_{i+r-1}, c_{i+r})$.

At a given time step t, each cell can be in one of the pre–defined states. Let $\mathcal{S}$ denote the set of possible states. We assume that the set $\mathcal{S}$ is finite. Let $s(c_i, t) \in \mathcal{S}$ denote the state of the i–th cell, at the time step t. For given t, the sequence $s(\cdot, t) = (s(c_i, t))_{i=1}^{N}$ will be called the configuration of the CA at time step t. The configuration $I = s(\cdot, 0)$, will be referred to as the initial configuration or initial condition. For given i and t, the sequence $(s(c_{i-r}, t), \ldots, s(c_{i+r}, t))$ will be called the neighborhood configuration of the cell c_i at the time step t.

The changes of states assigned to cells are described by a function called the local rule, which fulfills the following condition:

$$s(c_i, t+1) = f_A(s(c_{i-r}, t), \ldots, s(c_i, t), \ldots, s(c_{i+r}, t)). \tag{1}$$

The local nature of the CA is expressed in the local rule, by the fact that the future state of a given state, depends only on the states of its neighborhood.

The evolution of cell states of the CA, from time step t to time step $t+1$ is synchronous, *i.e.* the local rule is applied to all of the cells simultaneously. With the use of the local rule f_A, we can also define a global rule A satisfying:

$$s(\cdot, t+1) = A(s(\cdot, t)). \tag{2}$$

For the sake of simplicity and convenience, we will encode local rules in the form of tables. Such tables are called lookup tables (LUTs). For a given rule, the LUT lists all possible neighborhood configurations together with the output of the local rule. If we assume that there exists some ordering in the state set, we can write down all possible neighborhood configurations in lexicographical order. Consequently, to encode any local rule, we only need to store its values written in the fixed order of neighborhood configurations.

In subsequent sections, we consider CAs with the state set consisting of only two elements. Such automata will be called binary CAs. Binary CAs with neighborhood radius equal to one are called Elementary CAs (ECAs). ECAs are the most studied and well–understood group of CAs, due to low number of possible local rules (there are only 256 such rules).

In the case of binary CAs, we will always use the state set $\mathcal{S} = \{0, 1\}$. Hence, the LUT of a local rule, for a CA with radius r, will be a bit string of length $2^{2\,r+1}$. Such a bit string can be interpreted as an integer written in the base of 2, such that a unique number can be assigned to every rule [23]. Table 1 shows the general form of the LUT for a local rule of some ECA, as well as the numbering scheme.

Table 1: LUT of ECA local rule $n = (l_7, l_6, l_5, l_4, l_3, l_2, l_1, l_0)_2$.

$(1,1,1)$	$(1,1,0)$	$(1,0,1)$	$(1,0,0)$	$(0,1,1)$	$(0,1,0)$	$(0,0,1)$	$(0,0,0)$
l_7	l_6	l_5	l_4	l_3	l_2	l_1	l_0

Summing up, a 1D, deterministic CA $\mathscr{A}$ with a symmetric neighborhood as discussed above, could be represented by:

$$\mathscr{A} = \langle \mathcal{C}, N, r, \mathcal{N}, \mathcal{S}, A, f_A \rangle, \tag{3}$$

$\mathcal{C}$ is the cell set containing N elements, r is the neighborhood radius defining neighborhood $\mathcal{N}$, $\mathcal{S}$ is the state set, A is the global rule and f_A is the local rule.

3 CONTINUOUS CELLULAR AUTOMATA

As mentioned in Section 2, CAs are traditionally defined with the use of finite state spaces. Various methods of relaxing this constraint were proposed – most notably the use of so–called Fuzzy CAs (FCAs) [1, 2, 9]. Based on the construction of FCAs, we present a richer class of CAs, namely CCAs, which were introduced in [4].

Assume that we consider 1D CAs with a symmetric neighborhood of radius r. The definition of a CCA is analogous to the CA case except for two differences. The first one is the change of the state set S. For CCAs we assume $\mathcal{S} = [0, 1]$. The second difference is that the local rule of a CCA is represented by a multivariate, real–valued polynomial:

$$f(s(c_{i-r}, t), \ldots, s(c_{i+r}, t)) = \sum_{j=0}^{2^{2r+1}-1} l_j \left(\prod_{k=-r}^{r} n_f(s(c_{i+k}, t), j) \right), \quad (4)$$

where $l_i \in [0, 1]$ are parameters characterizing the rule, which are a generalization of the entries from the LUT, and $n_f(s(c_{i+k}, t), j)$ is defined as:

$$n_f(s(c_{i+k}, t), j) = \begin{cases} s(c_{i+k}, t), & \text{if } \mathrm{bin}(j)[k + r + 1] = 1, \\ 1 - s(c_{i+k}, t), & \text{if } \mathrm{bin}(j)[k + r + 1] = 0. \end{cases} \quad (5)$$

The bin function corresponds to the binary representation of integers. It is defined in such a way that $\mathrm{bin}(j)[k]$ is the k-th digit, from left to right, of the binary representation of an integer j, satisfying $j = \sum_m \mathrm{bin}(j)[2\,r + 1 - m]\,2^m$.

It is easy to check that the definition of a CCA is formally correct. Indeed, the values taken by the function f defined in Eq. (4) are guaranteed to belong to $[0, 1]$.

The definition of CCAs discussed here is consistent with binary CAs in the sense that the local rule of any binary CA could be written in the form of Eq. (4), with parameters $l_j \in \{0, 1\}$ taken directly from the LUT (see Table 1). Therefore, CCAs could be considered as an extension of binary CAs.

As an example we consider the Elementary CA (ECA) rule 150. Its local rule can be represented in the form of Eq. (4) as:

$$f_{150}(p, q, r) = pqr + p(1-q)(1-r) + (1-p)q(1-r) + (1-p)(1-q)r, \quad (6)$$

where variables p, q, r correspond to the states of right–most, central, and left–most cell in the neighborhood.

Note that some authors opt for a different definition of CCAs, and relate it to the concept of Coupled Map Lattices (CMLs) [15]. The exact relation between CCAs defined here, and CMLs is not yet known, and thus we consider only systems following the above CCA definition.

4 PROBLEM FORMULATION

In this paper we consider the DCT problem in the continuous setting. It is required to redefine (extend) the problem statement, to account for the real–valued states. Although we are considering states from the set $\mathcal{S} = [0, 1]$, we assume that the initial conditions, contain only cells in states 0 or 1. This technical simplification, which in fact results in finiteness of the number of possible initial conditions, enables us to relate the variant of the DCT presented here, to the classical formulation. Additionally, we will only consider CCAs with an odd number of cells, so that the majority could be always defined.

For simplicity we will denote the configurations of a CCA with capital letters, and treat the states of cells in the configuration as vector positions. So if J is a configuration of a CCA, for some time step, we will write $J = (J_1, \ldots, J_N)$, where N denotes the number of cells, and $J_i \in [0, 1]$. Letter I will be reserved for initial configurations.

Informally speaking, the desired outcome of a CCA "solving the DCT" is to evolve, in the finite time, to a configuration which is "close" to "all–0s" or "all–1s" configuration, depending on the number of 0s and 1s in the initial configuration.

We will now formalize this concept. For any configuration J in $[0, 1]^N$ we define the density of the configuration $\rho(J)$ as:

$$\rho(J) = N^{-1} \sum_{i=1}^{N} J_i. \tag{7}$$

For the initial configuration I, we know that $\rho(I) > 0.5$ or $\rho(I) < 0.5$, due to the fact that N is odd. Obviously, for any configuration J we know that $\rho(J) \in [0, 1]$.

We introduce a new class of configurations named α–homogenous configuration. Recall that a homogenous configuration is a configuration in which all cells are in the exact same state. The α–homogenous configuration is the generalization of this concept. We say that J is an α–homogenous configuration, if and only if, for any i, k we know that $|J_i - J_k| \leq \alpha$. We

will only consider $\alpha \in [0, 1]$. In other words, cells in α–homogenous configurations are in states which are relatively close to each other.

Obviously all possible configurations in our setting, are 1–homogenous. Additionally the 0–homogenous configurations are equivalent to the homogenous configurations. Directly from the definition we can check that α–homogenous configurations fulfill the property described by the following proposition.

Proposition 1. *Assume that J is a α–homogenous configuration, for some $\alpha \in [0, 1]$. Then J is also α'–homogenous for any $\alpha' \geq \alpha$.*

Using the concept of α–homogenous configurations, we will define the α–DCT problem, which will generalize the standard DCT. Let $\mathscr{A}$ be a CCA. We say that $\mathscr{A}$ solves the α–DCT if and only if, for any initial configuration $I \in \{0, 1\}^N$, there exist a time step τ such that, for any $t > \tau$, after t time steps, the automaton $\mathscr{A}$ evolves to an α–homogenous configuration J^t, such that if $\rho(I) < 0.5$ then $\rho(J^t) \leq \alpha$, and if $\rho(I) > 0.5$ then $\rho(J^t) \geq 1 - \alpha$,

Note that if $\alpha = 0$ then the α–DCT is equivalent to the standard DCT formulation. It is relatively easy to check that 1–DCT is trivial, *i.e.* every CCA $\mathscr{A}$ solves the 1–DCT.

The choice of a reasonable range of values for the parameter α should be based on the two following properties of the α–DCT problem. The first one is a direct consequence of Proposition 1.

Proposition 2. *Let $\mathscr{A}$ be a CCA, which solves the α–DCT for some $\alpha \in [0, 1]$. Then $\mathscr{A}$ solves the α'–DCT for any $\alpha' \geq \alpha$.*

The second of the properties is more important, as it ties the α–DCT with the direct meaning of the standard DCT. Before stating the proposition we need to introduce an auxiliary definition.

We say that a CCA $\mathscr{A}$, which solves the α–DCT for some α, is an unambiguous density classifier if the determination of the initial density is possible by examining the state of a single cell, after a finite number of time steps. Additionally, the minimum number of time steps needed, needs to be depended only on the initial configuration in question. If the state of the cell is lower than 0.5 then the initial density was lower than 0.5. Similarly if the state of the cell is higher than 0.5, the initial density was higher than 0.5. The word unambiguous refers to the requirement, that the choice of the cell could not impact the classification.

Note that not all of the solutions to the α–DCT are unambiguous density classifiers. Obviously for $\alpha = 1$, we can take any rule as a solution,

for example a rule that transfer any initial configuration to a homogenous configuration with all 0s. In such case, configurations with density higher then 0.5, will always be incorrectly classified. Therefore such a rule is not a valid unambiguous density classifier. The following propositions shows a simple criteria, describing some of the unambiguous density classifiers.

Proposition 3. *Assume that $\mathscr{A}$ is the solution of the α–DCT problem for some $\alpha \leq 0.25$. Then $\mathscr{A}$ is an unambiguous density classifier.*

As a consequence, we will be interested in solving the problem for $\alpha \leq 0.25$.

As noted earlier, it is known that there are no binary CAs that can fully solve DCT. The situation for CCAs and α–DCT is not yet understood. We know that 1–DCT is trivially solved by any CA and CCA rule. Is is expected that the situation is similar for α close to 1. Unfortunately we are not able to answer the general question regarding existence of solutions for any α within the scope of this paper. Instead we will concentrate of finding best possible CCAs that "almost" solve the DCT, *i.e.* CCAs that exhibit correct behavior for the highest possible number of initial conditions.

5 FINDING THE SOLUTION OF AN α–DCT WITH THE USE OF A GA

Starting from this section we will concentrate on CCAs with neighborhood radius $r = 1$, which could be considered as a generalization of ECAs. This choice is motivated by the fact, that there are known SCAs with neighborhood of radius 1, that solve the classical DCT with an arbitrary precision [8].

The proposed approach of finding the solutions of the α–DCT is to use a heuristic search technique, namely a GA [13], which is often used to find CA rules based on various conditions [3].

We start with the outline of the GA used for our problem. The algorithm starts with a random selection of a set of CCA local rules. Such a set will be called the initial population. Each of the elements in the population, refereed to as an individual, is assigned a fitness value. Fitness value, which is a number from the unit interval, measures how well a given rule is performing in solving the α–DCT. The fitness is calculated by examining the rule over a test set of initial conditions. After that, a new population is being built by transforming existing individuals. Individuals from the current population are selected for "reproduction" in such a way, that the fittest individuals have the highest probability of selection. Individuals are selected

in pairs, called the "parent" pairs. Each pair of parents is then transformed to one "offspring" individual with the use of the genetic operators. Firstly the cross–over operator is applied on the pair of parents, which results in creating the one offspring rule. The goal is to transfer some of the features of both of the parents to the offspring. Then the mutation operator is applied on the offspring individual, with the goal of brining innovation into the population. After building a new population, the fittest individuals from the previous one, forming so–called elite, are transferred to the new population without any changes, which increases the chances of gaining a stable growth between populations. The process is then repeated many times. The test set, used for fitness calculations, is being updated after evolving each of the populations. This gives the ability to evolve more generalized solutions. Below each of the aforementioned steps is defined more precisely.

The population consist of the local rules of CCAs, encoded by the vectors of parameters $l_i \in [0, 1]$. Since the radius of the neighborhood is known to be 1, such vectors contain $L = 8$ elements. So if A is an element of the population, then $A = (l_0, l_1, \ldots, l_{L-1})$ and $l_i \in [0, 1]$.

By means of $\mathcal{P}_i$ we will denote the population at the i–th iteration of the GA. Initial population $\mathcal{P}_0$ is created by selecting vectors randomly, while populations $\mathcal{P}_i$, for $i > 0$, are composed using the genetic operators defined below. We will use fixed–size populations containg P rules.

The goal of a GA is to maximize a predefined fitness function. In our case, the fitness function will be defined as a ratio of successful classifications of the test initial configurations to the total number of test cases in a test set $\mathcal{I}$.

To improve the quality of solutions obtained by the algorithm, the test set $\mathcal{I}$ used for fitness calculations is being updated constantly. The algorithm starts with predefined test set $\mathcal{I}$ containing randomly selected initial conditions. We assume that elements of $\mathcal{I}$ are numbered, *i.e.* $\mathcal{I} = (I^{(0)}, \ldots, I^{(M-1)})$. After i–th population of the GA is created, the initial condition $I^{(i \bmod M)}$ is being replaced with a new, randomly selected initial condition. Such approach makes the problem harder, since individuals in the population need to adapt constantly to the new environment. On the other hand, for high M, the impact of replacing one of the test cases is small enough, to allow the population to progress.

Note that in the definition of the α–DCT, the requirement is that the specified conditions are fulfilled for all of the time steps, starting from some time step τ, and there are no restrictions on the value of τ. Due to practical reasons, we impose a limit $\tau < 10\,N$, and additionally assume

that if the condition is fulfilled for the time step $t = 10\,N$, then it is also fulfilled for all of further time steps. This simplification enable us to calculate fitness in practice. Having defined the fitness function, we now turn to the definition of the genetic operators.

Selection and reproduction – those operators are responsible for selecting individuals from population $\mathcal{P}_{i-1}$ to build population $\mathcal{P}_i$. Individuals are selected at random with replacement, with a selection probability proportional to the fitness value. Selections are being made in pairs and there are P pairs of parents selected from the population $\mathcal{P}_{i-1}$. From each pair, one offspring rule is built with the use of the cross–over operation. After that, the mutation operator is applied and the individual is placed in the population $\mathcal{P}_i$.

Cross–over – assume that A and B are two individuals selected from $\mathcal{P}_{i-1}$ for which holds that $A = (l_j)$ and $B = (m_j)$. For such a pair, a random number $r \in [0, 1]$ is selected (with uniform distribution). The resulting rule $C = (n_j)$ is built by following formula:

$$n_j = r\, m_j + (1 - r)\, l_j. \tag{8}$$

Mutation is done by introducing small perturbations to some of the components of vectors defining individuals. The mutation procedure randomly selects one entry l_j in the vector, and then picks a random number $m \in [-\mu_r, \mu_r]$ for small $\mu_r > 0$ with uniform distribution. The component l_j is then updated with the value l'_j calculated according to the formula:

$$l'_j = \max(0, \min(1, l_j + m)). \tag{9}$$

This procedure is repeated $\mu_c > 0$ times for each individual, so at most μ_c vector positions are mutated.

Normally the GA stops when a perfect solution is found. In our case the maximum possible value of the fitness function is 1, therefore one could consider stopping the algorithm after finding such a rule. Taking into account that the test set $\mathcal{I}$ is constantly being updated, also the fitness function changes. Therefore obtaining the maximum possible value of fitness at some point in the evolution of populations is not necessarily a desired stopping condition, since the value could decrease at any of the next steps. Due to this we opt for a simple stoping condition relating to a pre–define number of allowed GA iterations, denoted by Λ. The algorithm runs for Λ iterations and returns the rule that was fittest after the last iteration. Such an approach can be successful only coupled with the application of the elite survival scheme. After evolving population $\mathcal{P}_i$, we pick the "elite" consisting of $P_E \ll P$ of the fittest individuals from the population $\mathcal{P}_{i-1}$ and place

them at random positions in $\mathcal{P}_i$ replacing the formerly evolved individuals. This gives that algorithm a chance to evolve generalized solutions, which are likely to behave correctly on a large number of initial conditions.

6 RESULTS OF THE EXPERIMENT

In this section we present a summary of the results obtained by implementing the GA method presented in Section 5. We considered $N = 69$ cells, and $\alpha = 0.1$. The GA was using populations of $P = 60$ individuals, with the elite containing $P_E = 6$ fittest individuals. The test set contained $|\mathcal{I}| = 500$ initial conditions. Mutation parameters were set to $\mu_c = 6$, $\mu_r = 0.015$, and the GA was executed for $\Lambda = 2000$ iterations.

In this experiment we selected the initial conditions for $\mathcal{I}$, using a selection procedure that guarantees equal selection probability for each of the possible densities. Our initial experiments shown, that such a selection procedure increases the performance of the GA.

To verify how performant were the rules obtained in the experiment, we selected a validation set of initial conditions $\mathcal{I}^\star$, containing $|\mathcal{I}^\star| = 10^5$ elements. This set was not used by the GA directly. Instead, at every 20–th iteration of the GA, we calculated the fitness for the elite individuals using $\mathcal{I}^\star$, for diagnostic purposes. The results are shown on Fig .1. Each line corresponds to one of 20 GA executions. At each examined iteration, we plotted the maximum fitness of the elite in given GA execution.

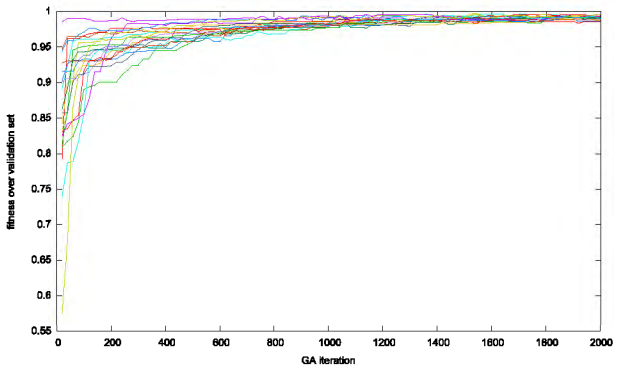

Fig. 1: Plot showing the change of the maximum fitness over the validation set, for the elite during GA iterations

As you can see, in all of the cases, the elite progresses toward the optimal value, which means that the elite individuals are indeed generalizing

their abilities to correctly classify diverse cases. On the other hand, none of the rules evolved by the GA was able to correctly classify all of the test cases in $\mathcal{I}^\star$. As mentioned earlier, it is not yet known if such rule exists for small α. Further experiments are planned in order to better understand the properties of the proposed GA, as well as properties of the rules discovered by the GA.

7 SUMMARY

In this paper the α–DCT problems was proposed, as a new variant of the classical DCT. Such a problem, coupled with CCAs was not yet discussed in the literature. Heuristic strategy of solving this problem, based on a GA was proposed. Both the problem statement and the solution strategy presented in the paper open few new, interesting research topics. Most importantly it is not yet known what are the conditions for the existence of the solution of the α–DCT problem. The α–homogenous configurations were introduced in this paper, as an auxiliary definition for the α–DCT. Understanding the behavior of CCAs on such configurations is an interesting, and difficult problem on its own and it will be further investigated.

References

1. Betel, H., Flocchini, P., (2009) On the Asymptotic Behavior of Fuzzy Cellular Automata. Electronic Notes in Theoretical Computer Science 252, 23–40
2. Betel, H., Flocchini, P., (2009) On the Relationship Between Boolean and Fuzzy Cellular Automata. Electronic Notes in Theoretical Computer Science 252, 5 – 21
3. Bołt, W., Baetens, J.M., De Baets, B., (2013) Identifying CAs with evolutionary algorithms. In: Proceedings 19th International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA 2013). Exploratory Papers, 11–20
4. Bołt, W., Baetens, J.M., De Baets, B., (2014) Analysis of stochastic CAs with the use of deterministic rules. In: Proceedings 20th International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA 2014). Exploratory Papers
5. Briceño, R., de Espanés, P.M., Osses, A., Rapaport, I., (2013) Solving the density classification problem with a large diffusion and small amplification cellular automaton. Physica D: Nonlinear Phenomena 261, 70 – 80
6. Das, D., A (2012) Survey on cellular automata and its applications. In: Krishna, P., Babu, M., Ariwa, E. (eds.) Global Trends in Computing and Communication Systems, Communications in Computer and Information Science, vol. 269, pp. 753–762. Springer Berlin Heidelberg
7. Das, R., Crutchfield, J.P., Mitchell, M., Hanson, J.E., (1995) Evolving globally synchronized cellular automata. In: Proceedings of the Sixth International Conference on Genetic Algorithms. pp. 336–343. Morgan Kaufmann
8. Fatès, N., (2013) Stochastic Cellular Automata Solutions to the Density Classification Problem - When Randomness Helps Computing. Theory of Computing Systems 53(2), 223–242
9. Flocchini, P., Geurts, F., Mingarelli, A., Santoro, N., (2000) Convergence and Aperiodicity in Fuzzy Cellular Automata: Revisiting Rule 90. Physica D: Nonlinear Phenomena 142(1), 20–28

10. Fukś, H., (1997) Solution of the density classification problem with two cellular automata rules. Phys. Rev. E 55, R2081–R2084
11. Gabriele, A., (2005) The density classification problem for multi-states cellular automata. In: Capcarrére, M., Freitas, A., Bentley, P., Johnson, C., Timmis, J. (eds.) Advances in Artificial Life, Lecture Notes in Computer Science, vol. 3630, pp. 443–452. Springer Berlin Heidelberg
12. Gács, P., Kurdyumov, G.L., Levin, L.A., (1978) One-dimensional uniform arrays that wash out finite islands. Problemy Peredachi Informatsii 14(3), 92–96
13. Holland, J., (1975) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press
14. Jeanson, F., (2008) Evolving asynchronous cellular automata for density classification. In: Bullock, S., Noble, J., Watson, R., Bedau, M.A. (eds.) Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems. pp. 282–288. MIT Press, Cambridge, MA
15. Kaneko, K., (1993) Theory and Applications of Coupled Map Lattices. Wiley
16. Kanoh, H., Wu, Y., (2003) Evolutionary design of rule changing cellular automata. In: Palade, V., Howlett, R., Jain, L. (eds.) Knowledge-Based Intelligent Information and Engineering Systems, Lecture Notes in Computer Science, vol. 2773, 258–264. Springer Berlin Heidelberg
17. Land, M., Belew, R.K., (1995) No Perfect Two-State Cellular Automata for Density Classification Exists. Phys. Rev. Lett. 74, 5148–5150
18. de Oliveira, P., (2013) Conceptual connections around density determination in cellular automata. In: Kari, J., Kutrib, M., Malcher, A. (eds.) Cellular Automata and Discrete Complex Systems, Lecture Notes in Computer Science, vol. 8155, pp. 1–14. Springer Berlin Heidelberg
19. Packard, N.H., (1988) Adaptation toward the edge of chaos. University of Illinois at Urbana-Champaign, Center for Complex Systems Research
20. Sahoo, S., Pal Choudhury, P., Pal, A., Nayak, B.K., (2009) Solutions on 1D and 2D Density Classification Problem Using Programmable Cellular Automata. ArXiv e-prints
21. Stone, C., Bull, L., (2009) Solving the density classification task using cellular automaton 184 with memory. Complex Systems 18(3), 329
22. Von Neumann, J., (1966) Theory of Self-Reproducing Automata. University of Illinois Press, Champaign, IL, USA
23. Wolfram, S., (1983) Statistical mechanics of cellular automata. Rev. Mod. Phys. 55, 601–644

KLASYFIKACJA GĘSTOŚĆI KOMÓREK PRZY UŻYCIU CIĄGŁYCH AUTOMATÓW KOMÓRKOWYCH

Streszczenie. Jednym z częściej studiowanych problemów z obszaru analizy możliwości obliczeniowych automatów komórkowych jest problem klasyfikacji gęstości, zwany też problemem głosowania większościowego (ang. *density classification problem* lub *majority classification problem*). Sformułowanie problemu jest następujące. Startując z dowolnej konfiguracji początkowej mamy dojść do konfiguracji homogenicznej, w której wszystkie komórki są w tym ze stanów, w którym była większość komórek w konfiguracji początkowej. Tak postawiony problem nie jest żadnym wyzwaniem, jeśli dysponujemy globalną pamięcią lub możliwością inspekcji globalnego stanu, czego brakuje automatom komórkowym z definicji. Dla skończonej liczby komórek problem klasyfikacji gęstości nie ma rozwiązania. W literaturze rozwiązuje się problemy zbliżone m. in. przez modyfikację definicji samych automatów komórkowych. W niniejszej pracy podążamy tą drogą i definiujemy nowy problem, który nazywamy α–klasyfikacją gęstości dla $\alpha \in (0, 1)$. Ma on być rozwiązywany przez tzw. ciągłe automaty komórkowe. W pracy zaproponowano algorytm genetyczny, który poszukuje rozwiązania przedmiotowego problemu. Wstępne wyniki eksperymentów pokazują, że dla $\alpha = 0.1$, jesteśmy wstanie znaleźć rozwiązania poprawne dla ponad 95% ze 100 000 losowo wybranych konfiguracji początkowych.
Słowa kluczowe: automaty komórkowe, problem głosowania większościowego

ISBN 83-894-7555-3