Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2014) 22: 1653 – 1663

doi:10.3906/elk-1211-26

Frequency-emulated uniform cellular automata

Hürevren KILIÇ*
Computer Engineering Department, Gediz University, Seyrek, Menemen, İzmir, Turkey

Received: 07.11.2012 • **Accepted:** 26.03.2013 • **Published Online:** 07.11.2014 • **Printed:** 28.11.2014

Abstract: The notion of a frequency-emulated (f-emulated) uniform cellular automata (CA) that enables the behavior emulation of some elementary CA via memory usage is introduced. An algorithm that generates f-emulated uniform CA sets is developed and an upper bound for its output size is given. It is observed that traffic rule 184 together with its 2-emulator version, which generates the behavior of the known majority rule 232, performs the density classification task perfectly. Moreover, it is possible to use a 2-emulated uniform CA for the solution of the parity problem.

Key words: Cellular automata, frequency-emulated cellular automata, core emulator set, f-emulated set, density classification task, parity problem

1. Introduction

The establishment of high-quality computational models for better understanding is a critical part of the study of complex systems, including, for example, the fields of physics, biology, ecology, sociology, and economics. Self-organization of simple components and the resulting aggregate global system behavior are common features of most natural and artificial complex systems. Among different computational models, cellular automata (CA) [1] provide discrete dynamical system models, in which locally interconnected set of cells (i.e. automata) evolve at discrete time steps through mutual interactions. Instead of describing a complex system with complex equations, the CA model enables the complexity to emerge by interaction of its simple components following simple rules.

Starting with specific local behavior in mind and to observe its long-term global impact (forward problem) and identifying local transitions that support some given global behavior (inverse problem) are typical involvements in the CA field. This study is neither about the forward nor the inverse problem, but is an investigation of interautomaton emulation possibilities and their consequences/use in problem solving. The purpose of the study is to attain a basis for CA through examination of their global behavior (or state space) level mimicking capabilities. The goal is to achieve still simpler computational models. By simpler model, we mean a uniform cellular model that does not require designed external automaton combinations/switches for solving hard computational tasks. For our purpose, we introduce a frequency-emulated (f-emulated) uniform CA and, based on it, 2 sets: the core emulator (CES) and f-emulated set (F-ES), are defined. Use of f-emulators in global computation that relies on local coordination is shown via the density classification task (DCT) [2] and parity problem (PP). The notion of f-emulation and the related sets proposed in this paper is original. Different from existing approaches to solve hard computational tasks via CA combinations, we introduce the f-emulated CA model, using which one can solve some hard tasks without automaton switches. Moreover, the proposed CES

*Correspondence: hurevren.kilic@gediz.edu.tr

defines a basis and reduces the search space for search-based approaches that aim to identify cellular models in solving hard computational tasks.

In Sections 2 and 3, we give formal definitions for CA, f-emulated uniform CA, and the emulation set. Section 3 also contains an algorithm that generates f-emulated uniform CA sets, an upper bound for its size, and a list of all 2-emulated automata and their CES. Section 4 contains emulations of known solutions to the DCT and PP. The last section is the concluding remarks.

2. Cellular automata

Deterministic CA is a model that consists of a collection of elementary automata with local interactions evolving in a parallel and synchronous way [1]. A 1-dimensional CA can be built up in the following way:

1) Begin with a cellular space that consists of an infinite line divided into cells indexed by some integer $i \in Z$.

2) There is a neighborhood relation that gives for any cell i, a finite list of cells indexed by $i-r$, $i-r+1, \ldots, i-1$, i, $i+1, \ldots,$ $i+r-1$, $i+r$ called its neighbors, from which it can directly receive information. r is called the neighborhood radius.

3) Specify a finite list of state Q with cardinality $|Q| = k$.

4) For cell i, of the cellular space, define a local transition function f_i from Q^{2r+1} to Q, where $2r+1$ is the number of neighbors of cell i. f_i specifies the state q_i^{t+1} of cell i at time $t+1$, as a function of states of its neighbors at time t, that is:

$$q_i^{t+1} = f_i\left(q_{i-r}^t, q_{i-r+1}^t, \ldots, q_{i-1}^t, q_i^t, q_{i+1}^t, \ldots, q_{i+r-1}^t, q_{i+r}^t\right). \tag{1}$$

5) Any function $c : Z \to Q$ that defines an assignment of states to all cells in the 1-dimensional cellular space is called a configuration and for any cell x_i, $c(x_i)$ is called the state of x_i under configuration c.

6) Simultaneous application of local transition functions f_i to all the cells of the cellular space defines a global transition function F that acts on the entire array, transforming any configuration c into a new configuration $c^{'}$ such that:

$$c^{'}(x_i) = f_i\left(c(x_{i-r}), c(x_{i-r+1}), \ldots, c(x_{i-1}), c(x_i), c(x_{i+1}), \ldots, c(x_{i+r-1}), c(x_{i+r})\right). \tag{2}$$

Usually, we assume that the cells are connected with a periodic neighborhood boundary condition that leads to simply finite configurations. If the same local transition function f_i is applied to all cells, the cellular automaton is called uniform CA; otherwise, it is called hybrid CA. In the case of 1-dimension, r-neighbor, k-state, the number of all possible uniform CA is: $k^{k^{2r+1}}$. For $r = 1$ and $k = 2$, it is 256. In the Wolfram naming convention [3], the automata (or rules) are enumerated from 0 to 255.

3. Behavior emulation of CA

Definition 1: Given r-neighbor, k-state uniform cellular automaton A defined by local transition function f_i, *f-emulation uniform cellular automaton* A^f is an r-neighbor k-state uniform cellular automaton having local transition function:

$$q_i^{t+1} = f_i^f \left(q_{i-r}^t, \ldots, f_i^j \left(q_{i-r}^t, \ldots f_i^1 \left(q_{i-r}^t, \ldots, q_i^t, \ldots, q_{i+r}^t \right), \ldots, q_{i+r}^t \right), \ldots, q_{i+r}^t \right), \quad (3)$$

where f_i^j indicates the jth application of f_i.

The local transition function of A^f is simply obtained by f times repetitive application of the local transition function of A while keeping the neighbors' state fixed. Different from the original CA transition, the f-emulated uniform CA requires f times consecutive application of the state transition function for cell i in every single CA update step. Note that each cell automaton i is responsible for the update of its current state only and throughout the f-emulation steps not the newly updated intermediate neighbor states but the initial neighbor states of f-emulation are considered. For example, let $r = 1$, $k = 2$, q_{i-1}^t=1, $q_i^t = 0$ and $q_{i+1}^t = 1$. Moreover, assume that $f_i(1, 0, 0) = 1$ and $f_i(1, 1, 0) = 0$ are 2 transition rules of the given CA A. Thus, the 3-emulation uniform CA A^3 of A computes q_i^{t+1} as $f_i^3(1, f_i^2\left(1, f_i^1(1, 0, 0), 0\right), 0) = 1$. Note that $A^1 = A$. The rationale behind the frequency emulation idea is to identify possible behavior mimicking relations among automata descriptions based-on state space dynamics. The frequency emulation idea extends the spatial cellular neighborhood to the temporal neighborhood via simple memorization.

Definition 2 *Emulation set $S(A)$ of a given r-neighbor k-state uniform CA A is $\bigcup_{i=1}^{\infty} A^i$.*

Clearly, $|S(A)| \leq k^{k^{2r+1}}$.

Proposition: *For a given r-neighbor k-state uniform CA A, $|S(A)| \leq k + LCM(1, 2, \ldots, k) - 2$, where $LCM(1, 2, \ldots, k)$ stands for the least common multiples of the first k numbers.* (4)

To see this, let us consider the local instance of the ith cell of A together with its r right and r left neighbors at time t. During emulation, while we keep its neighbors' states fixed, by the end of at most $k-1$ steps, we reach a local instance occurred before (i.e. local cycle of some length). Local cycles imply a global cycle entered by the end of some finite number of emulation steps. The upper bound for the global cycle length can be figured out as the LCM of k^{2r} numbers, where each can take value from 1 to k. The maximum global cycle length also implies the highest number of automaton that can be emulated by a single automaton. Furthermore, since $r \geq 1$ then $k^{2r} > k$ for any $k > 1$. Therefore, according to the pigeonhole principle, only k of k^{2r} such numbers contribute to the LCM. Moreover, the LCM of a smaller (i.e. $< k$) number of items cannot be greater than that of the first k numbers. By considering the global state reached by the end of $k-1$ number of state changes, we can conclude that $S(A) \leq k - 1 + LCM(1, 2, \ldots, k) - 1$.

The proposition gives us an idea about the limitations of the frequency emulation approach based on the number of states k. The size limit for emulation set $S(A)$ is considered in the following *Emulation_Set_Constructor* algorithm and the bound figured out in proposition is useful for time-efficient generation of the emulation set.

Note that the prime number theorem implies that the least common multiples of the first k positive integers (i.e. 1, 2, 6, 12, 60, 60, 420 ...) $LCM(1, 2, \ldots, k) = e^{k(1+o(1))}$ as $k \to \infty$. In other words, $\frac{ln(LCM(1,2,\ldots k))}{k} \to 1$ as $k \to \infty$. The importance of this expansion is due to figuring out the exponential number of automata that can be mimicked based on given uniform CA for increased values of k. Furthermore, bigger $|S(A)|$ values imply more flexibility in solving hard computational problems via fewer automata switches and by frequency emulation. Below, we give an algorithm for emulation set construction for a given uniform CA.

Algorithm *Emulation_Set_Constructor*

Input: r - Neighborhood radius; k - # of states; A - r-neighbor, k-state uniform CA

Output: B - Emulation set of A

[1] **{**

[2] $B = \{A\ \}$;

[3] $D = A\ ;$

[4] **for** $i = 2$ **to** $k + LCM\,(1, 2, \ldots, k) - 2$ **{**

[5] **for** $j = 1$ **to** k^{2r+1} **{**

[6] $q = \mathbf{k\_ary}\,(j-1)\ ;$

[7] $s = D.f\,(q)\ ;$

[8] $p = \mathbf{concat}\,(\mathbf{substr}\,(q, 1, r)\,,\ s,\ \mathbf{substr}\,(q, r+2,\ r))$

[9] $D\,[q] = A.f\,(p)\ ;$

[10] **}**

[11] **if** $D \in B$ **then return** B

[12] **else** $= B \cup \{D\ \}$;

[13] **}**

[14] **return** B;

[15] **}**

In the algorithm, the built-in functions **k_ary**, **concat**, and **substr** are used for decimal to base k conversion, string concatenation and substring extraction purposes. Following the initializations, in Line 6, **k_ary** patterns of length $2r+1$ are generated. In Line 7, we apply the local transition function f of D to q. Next, we construct and set the rule table entry for the newly established automaton D, in Lines 8 and 9, respectively. Line 11 checks the existence of the constructed automaton in the set. If not, it is added to the emulation set; otherwise, we guarantee that no further insertion to the set is possible and return the result.

The proposition gives an upper bound for $S\,(A)$ size; however, finding a tighter upper bound (if exists) may lead to construction of more efficient emulation set construction algorithm for the purpose. Time complexity of the algorithm can be figured out as: $\theta(e^{k(1+o(1))} * k^{2r+1}) * Complexity\,of\,function\,\mathbf{k\_ary}$. In the worst case, the function **k_ary** takes $\log_k k^{2r+1}$ time and so it is in $O(r)$. Therefore, the algorithm's complexity is $O(e^{k(1+o(1))} * k^{2r+1} * r)$.

In the proposition, we give an upper bound for the size of the emulation sets. For example, for any $r = 1$ and $k = 2$ uniform CA A, $|S\,(A)| \leq 2$. That means at most 2-emulation is possible. In general, we classify any r-neighbor, k-state uniform CA into 2 disjoint sets based on the f-emulation relation between them: 1) Core emulator set (CES) that contains all A that are not the f-emulation of any other automata except themselves and $|S\,(A)| \geq 1$. 2) The F-ES that holds all A that are some f-emulation of at least one other uniform CA B (i.e. $A \in S\,(B)$). Note that some members of CES can only emulate themselves (i.e. $|S\,(A)| = 1$). Using the emulation set constructor algorithm, we generate Table 1, which lists members of the CES with $|S\,(A)| = 2$ and corresponding members of the F-ES for all $r = 1$ and $k = 2$ uniform CA represented in the Wolfram naming convention. For example, each of rules 43, 46, and 139 automata are able to 2-emulate the rule 142 automaton.

Table 1. CESs and corresponding F-ESs for $r = 1$ and $k = 2$.

Core emulator set, *S(A) = 2*	2-emulated set	Core emulator set, *S(A) = 2* (Cntd.)	2-emul. set (Cntd.)
1	4	171	174
2	8	48, 96, 144	192
3, 6, 9	12	49, 52, 97, 100, 145, 148, 193	196
7	13	53, 101, 149	197
11	14	50, 56, 98, 104, 146, 152, 194	200
16	64	58, 106, 154	202
17, 20, 65	68	51, 54, 57, 60, 99, 102, 105, 108, 147, 150, 153, 156, 195, 198, 201	204
21	69	55, 61, 103, 109, 151, 157, 199	205
18, 24, 66	72	59, 62, 107, 110, 155, 158, 203	206
26	74	63, 111, 159	207
19, 22, 25, 28, 67, 70, 73	76	112	208
23, 29, 71	77	113, 116, 209	212
27, 30, 75	78	117	213
31	79	114, 120, 210	216
81	84	122	218
82	88	115, 118, 121, 124, 211, 214, 217	220
83, 86, 89	92	119, 125, 215	221
87	93	123, 126, 219	222
91	94	127	223
32	128	176	224
33, 36, 129	132	177, 180, 225	228
37	133	181	229
34, 40, 130	136	178, 184, 226	232
42	138	186	234
35, 38, 41, 44, 131, 134, 137	140	179, 182, 185, 188, 227, 230, 233	236
39, 45, 135	141	183, 189, 231	237
43, 46, 139	142	187, 190, 235	238
47	143	191	239
161	164	241	244
162	168	242	248
163, 166, 169	172	243, 246, 249	252
167	173	247	253
		251	254

The only self-emulating 16 members of the CES with $|S(A)| = 1$ are {0, 5, 10, 15, 80, 85, 90, 95, 160, 165, 170, 175, 240, 245, 250, 255} . The CES with $|S(A)| = 2$ holds 175 members and the corresponding F-ES holds 65 members. The f-emulation relation is not symmetric and the CES is sufficient to emulate all 256 automata, either via 1-emulation or 2-emulation. In [4], 256 elementary CA rules were divided into 88 equivalence classes based on conjugation, reflection and combined conjugation, and reflection transformations. The introduced 65 F-ES automata span 23 of them. It is interesting to observe that among 64 amphichiral elementary CA, 37 are able to 2-emulate some other CA. However, it is not possible to 2-emulate rule 110 (known to be Turing-complete) and the rules in its equivalence class (i.e. rules 137, 124, and 193). It can be interesting to investigate possible f-emulated automata usage in existing or potential solutions to known hard tasks.

4. Solutions via f-emulation

The density classification task is a computational task that demands CA to show the behavior of converging to a fixed point of all 1's if the initial system configuration is 1-dense (i.e. higher number of 1's), and to a fixed point of all 0's if the configuration is 0-dense (i.e. higher number of 0's). The behavior is required to be shown in some M number of steps in the order of CA with lattice size L and assuming periodic boundary conditions. An early work on DCT reported not perfect but approximate results [5]. By perfect result, it is meant that whatever the initial 0-dense or 1-dense configuration, the proposed solution converges to a correct fixed point of all 0's or all 1's, respectively. In 1995, the authors in [6] proved that no single 1-dimensional 2-state uniform CA exist that solve the original DCT without making some mistakes. Following this, the authors in [7] demonstrated that changing the output specification of the original problem and boundary conditions leads to a solution.

The evolution of asynchronous CA for DCT was studied in [8]. A solution to the problem by a variant of CA called programmable CA is given in [9]. The cellular programming approach based on nonuniform CA that locally coevolve to solve a given task can be found in [10]. A rule changing/switching methodology to find alternative solutions in an evolutionary manner is given in [11]. In [12], the existence of substantial homogeneity between the results for the problem's solution using elementary CA and multistate CA is verified. Like our approach, memory augmented/coupled approaches to the problem were developed in [13][14]. Use of stochastic CA with arbitrary precision for DCT was developed in [15]. In this study, a blend of stochastic and deterministic rule usage is shown to result in adjustable classification quality at the cost of longer convergence times.

In [16], it was proven that the use of rule 184 followed by rule 232 performs the task perfectly. In the paper, it is shown that the given configuration, say s of length L with density ρ, and consecutive application of 2 rules $F_{232}^{m}\left(F_{184}^{n}\left(s\right)\right)$, where $m=\lfloor\left(L-1\right)/2\rfloor$ and $n=\lfloor\left(L-2\right)/2\rfloor$, produces the result perfectly. Here, F_{i}^{m} indicates m times application of the global function associated with elementary CA rule i. The production $F_{232}^{m}\left(F_{184}^{n}\left(s\right)\right)$ consists of only 0's if $\rho<1/2$ and only 1's if $\rho>1/2$; otherwise, (i.e. $\rho=1/2$), it is an alternating sequence of 0 and 1, i.e. $\ldots 010101\ldots$ In the solution, the role of traffic rule 184 is to eliminate 11 patterns if $\rho<1/2$ (or pattern 00 if $\rho=1/2$) in n steps while preserving density ρ. The subsequent application of majority rule 232 for m steps provides replacement of the minority and leads to the homogeneous configuration of all 0's (or all 1's), respectively.

From Table 1, we know that rule 232 is the 2-emulation of rule 184 (i.e. *rule 232 automaton* $\in$ *S(rule 184 automaton)*). To see this, let us apply the emulation set constructor algorithm using the inputs $r=1;k=2$ and *rule* 184 *automaton*. Line 3 adds the rule 184 automaton to the output set. Based on the input values, the outer loop is executed only once and the inner one is executed 8 times to compute all of the possible alternative local neighborhood configurations. Table 2 shows variable value traces and 2-emulated automaton construction due to Lines 7, 8, and 9. For example, when $q=100$, the cell produces the output 1 and the configuration becomes 110. The second application of rule 184 to the obtained configuration produces output ; therefore, the transition rule for the emulated automaton should contain mapping $100\rightarrow 0$. Application of the 2-emulation transition rule for all other possible values of q results in constructed rule 232. Finally, Line 12 adds the constructed rule 232 automaton to the output set. As a consequence, for any arbitrarily given initial system configuration, n step execution of rule 184 and $2m$ step execution of its 2-emulated version solve the density task perfectly.

From Table 2, we can see that the same results can also be obtained using independent rule 226 and/or rule 178 two-emulations. Note that in the rule 232 execution part, allowing changes among the rule 232 two-

Table 2. Two-emulated rule 232 automaton constructions using rules 184, 178, and 226 as input instances to the emulation set constructor algorithm.

	2-emulator rule 184		2-emulator rule 178		2-emulator rule 226		2-emulator rule 232
q	s	p	s	p	s	p	$D(q) \leftarrow A.f(p)$
000	0	000	0	000	0	000	0
001	0	001	1	011	1	011	0
010	0	000	0	000	0	000	0
011	1	011	0	001	0	001	1
100	1	110	1	110	0	100	0
101	1	111	1	111	1	111	1
110	0	100	0	100	1	110	1
111	1	111	1	111	1	111	1

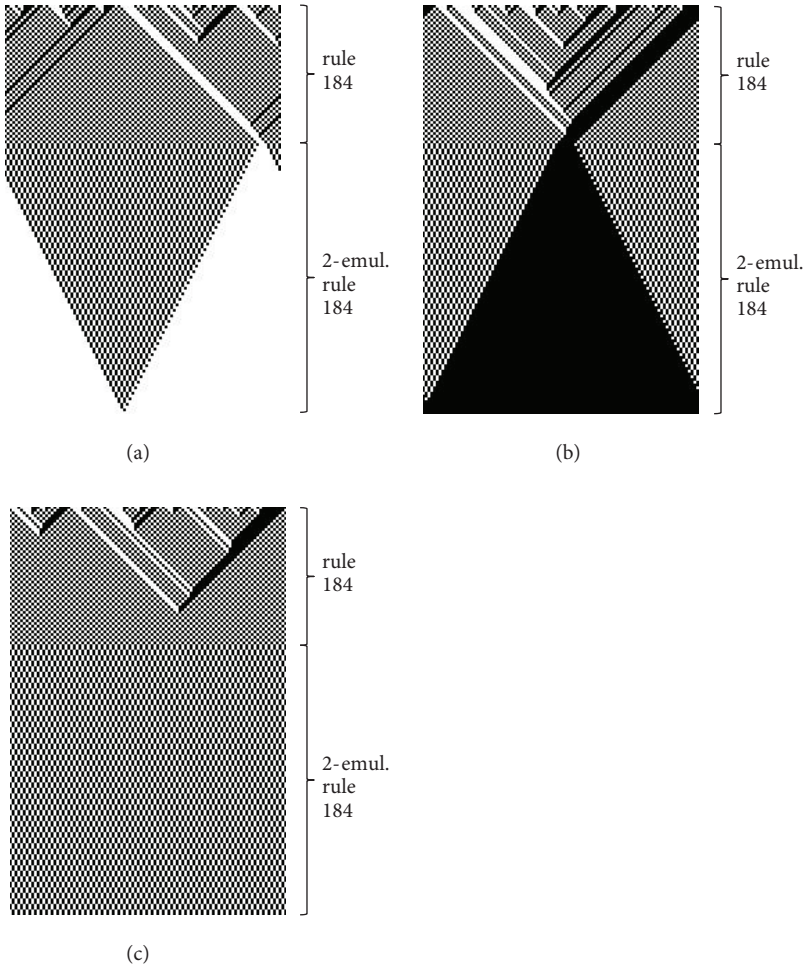

Figure 1. Density task: spatiotemporal diagrams for rule 184 followed by its 2-emulator variety for lattice size L = 100, where the initial densities of 1 s are: a) $\rho = 0.49$, b) $\rho = 0.53$, and c) $\rho = 0.50$.

emulators, one can talk about the existence of $3^{\lfloor (L-1)/2 \rfloor}$ alternative solutions to the problem. This is simply because consecutive application of different 2-emulators emulating the same CA does not affect the resulting global system dynamics. In Figure 1, example DCT spatiotemporal diagrams for 3 different initial densities for rule 184 followed by its majority rule emulator variety are given. The execution length of the 2-emulation part is depicted as around double its base rule execution part. However, it should be clear that 2-emulator implementation demands not only 2 times the computation of the local transition function in a 1-clock cycle, realized with memory usage, but also 1-step suppressed sensory capability that prevents information flow to other cells. Otherwise, one can only talk about simulation (but not emulation) of automaton behavior. Note that the basic distinction between emulation and simulation is that the former is focused on exact reproduction of the system behavior while the latter uses some abstract model of the system for behavior generation purposes.

Similarly, in Figure 2, spatiotemporal diagrams for the same task with the same initial densities and lattice size but using a rule 178-based 2-emulated version of rule 232 are given. The basic difference between the diagrams given in Figures 1 and 2 is in the rule 184- and 178-based 2-emulation regions, due to the existence of different ways to emulate rule 232. Among 3 alternative 2-emulators of rule 232 (namely rules 184, 178, and 226), the one based on rule 184 does not require any rule switch. However, all 3 emulations still require intermediate transition steps to execute the task. To the best of the author's knowledge, the DCT solution: rule 184 followed by the rule 178-based 2-emulation of rule 232, points to the first time use of rule 178 in the solution. It can be interesting to examine alternative f-emulated solutions based on known alternative chained 2-rule and 3-rule solutions [11] using Table 1.

Another hard task for elementary CA is the PP. Given a finite binary string, say s, the parity of s is equal to 1 if the number of 1's in s is odd; otherwise, it is 0. The PP in a CA context is to find elementary CA, which will converge to all 1's if the parity of s is 1; otherwise, it will converge to all 0's, where s is the initial global state of the CA with periodic boundary conditions. In [17], it is concluded that no single $r = 1$ CA rule exists to solve the PP with fixed boundary conditions.

In [18], a solution involving more than 1 rule is developed. In their solution:

1. Given lattice size L being odd, application of the $\left(F_{132}^{\lfloor L/2 \rfloor} F_{222}^{\lfloor L/2 \rfloor}\right)^{\lfloor L/2 \rfloor}$ operator/program to s generates the desired output. Note that both rule 132 and rule 222 $\in$ F-ES. Based on the results given in Table 1, the existence of 3 alternative 2-emulators for rule 132 and 3 alternative 2-emulators for rule 222 implies $\left(3^{\lfloor L/2 \rfloor} 3^{\lfloor L/2 \rfloor}\right)^{\lfloor L/2 \rfloor}$ alternative only 2-emulated CA solutions to the problem each executes in the order of $\theta\left(L^2\right)$ time steps.
2. Similarly, if $L = 2q$, where q is odd, application of the $F_{254}^{\lceil L/2 \rceil} F_{76} \left(F_{132}^{\lfloor L/2 \rfloor} F_{222}^{\lfloor L/2 \rfloor}\right)^{\lfloor L/2 \rfloor}$ operator/program to s generates the desired output. Note that rules 254, 76, 132, and 222 $\in$ F-ES. In this case, the existence of one 2-emulator for rule 254, 7 alternative 2-emulators for rule 76, 3 alternative 2-emulators for rule 132, and 3 alternative 2-emulators for rule 222 implies $7\left(3^{\lfloor L/2 \rfloor} 3^{\lfloor L/2 \rfloor}\right)^{\lfloor L/2 \rfloor}$ alternative only 2-emulated CA solutions to the problem each still executes in the order of $\theta\left(L^2\right)$ time steps.
3. The last case of their solution is when $L = 2^m q$, where $m \geq 2$ and q is odd, application of the $F_{254}^{\lceil L/2 \rceil} F_{76} \left(F_{132}^{\lfloor L/2 \rfloor} F_{222}^{\lfloor L/2 \rfloor} F_{184} F_{252}\right)^{m-1} \left(F_{132}^{\lfloor L/2 \rfloor} F_{222}^{\lfloor L/2 \rfloor}\right)^{\lfloor L/2 \rfloor}$

operator/program to s generates the desired output. Note that rules 254, 76, 132, 222, and 252 $\in$ F-ES. However, rule 184 $\in$ CES. Thus, it is not possible to set up a rule sequence constituted from only 2-emulated

CA. We have one 2-emulator for rule 254, 7 alternative 2-emulators for rule 76, 3 alternative 2-emulators for rule 132, 3 alternative 2-emulators for rule 222, and 3 alternative 2-emulators for rule 252. This implies $7\left(3^{\lfloor L/2\rfloor}3^{\lfloor L/2\rfloor}3\right)^{m-1}\left(3^{\lfloor L/2\rfloor}3^{\lfloor L/2\rfloor}\right)^{\lfloor L/2\rfloor}$ alternative solutions to the problem each executes in the order of $\theta\left(L^2+L\log L\right)$ time steps.

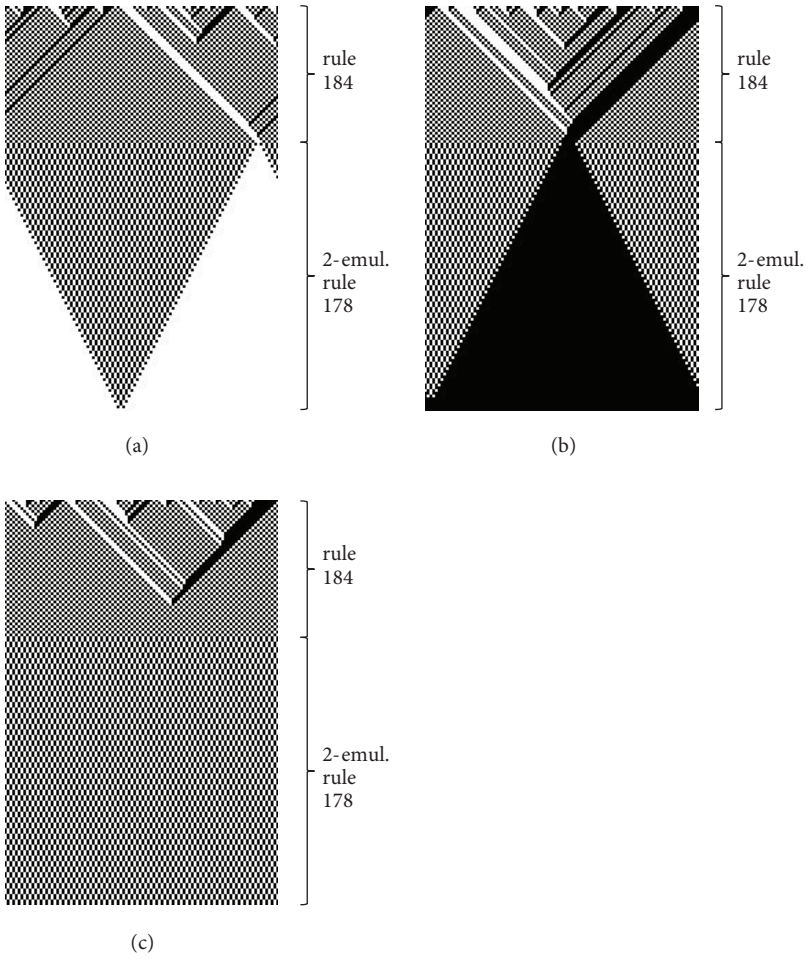

Figure 2. Density task: spatiotemporal diagrams for rule 184 followed by 2-emulator rule 178 for lattice size L = 100, where the initial densities of 1 s are: a) $\rho = 0.49$, b) $\rho = 0.53$, and c) $\rho = 0.50$.

In [19], alternative chains of elementary CA solutions based on rule behavior analysis are reported. Each reported alternative solution may bring the potential application of their 2-emulated versions. Recently, in [20], it was proved that there exists no radius 2-rule that can solve the PP from arbitrary initial configurations. Furthermore, their designed radius 4-rule that provides quick convergence for any initial condition can be examined for higher frequency emulations using the emulation set constructor algorithm.

5. Conclusions

F-emulated uniform CA are introduced. It is shown that the behavior emulation of some elementary CA via increased update frequencies is possible. The proposed model is not a simple memory addition that causes a direct size increase in the transition function. While keeping the transition function the same, it updates the

current cell state as frequently as it is required for emulation purposes, which results in global computation that relies entirely on the local coordination of actions at the cost of an extended execution time in the order of f. The generated f-emulated automaton equivalences given in Table 1 provide building blocks for uniform and simpler solutions to the known hard computational tasks. An algorithm that generates f-emulated uniform CA sets is developed and an upper bound for its output size is given. Based on the f-emulation capabilities of elementary CA, we classify them into 2 sets: CES and F-ES.

We show that combined use of some elementary CA and their f-emulated versions enables us to find alternative solutions to known problems, as long as we allow minimal memory usage coupled with frequency emulation. For example, we observe that traffic rule 184 together with its 2-emulated version, which generates the behavior of majority rule 232 (i.e. already known rule 184 and rule 232 coupled solution [16]), performs the DCT perfectly. The f-emulation–based solution to the DCT uses the separation and suppression properties of rule 184 and its 2-emulation version, respectively. *Thinking twice* in 1 clock cycle together with a 1-step output suppression (i.e. *silence*), and/or memory usage, lead to the frequency-driven alternative execution of original core rule 184 automaton without any dramatic change in functionality (i.e. switching to a totally new rule 232). We can speculate that *self-concentration* on processing via a frequency increase to survive is a typical individual behavior observed in most natural systems before any possible functional level cooperation formation among individuals. Similar to using already known chained solutions to the DCT, we figure out alternative solutions for the PP. However, this time we are not able to identify any single rule coupled with its augmented 2-emulator variety that solves the PP.

As a consequence of our classification, the CES can either be used as background knowledge in CA-based analysis and design efforts and/or be part of rule space reduction for performance demanding evolutionary CA search methodologies, in which finding sequential combinations of CA rules that solve given hard tasks is the ultimate goal. In future, one can investigate alternative frequency emulation possibilities for other hard computational tasks.

References

[1] J. von Neumann, Theory of self-reproducing cellular automata, University of Illinois, Urbana, 1966.

[2] N.H. Packard, "Adaptation toward the edge of chaos", In: J.A.S. Kelso, A.J. Mandell, and M.F. Shlesinger (eds.), Dynamic Patterns in Complex Systems, World Scientific Singapore, pp. 293–301, 1988.

[3] S. Wolfram, "Statistical mechanics of cellular automata", Reviews of Modern Physics, Vol. 55, pp. 601–644, 1983.

[4] S. Wolfram, "A new kind of science", Champaign, IL: Wolfram Media, Inc. 2002.

[5] 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.

[6] M. Land, R.K. Belew, "No perfect two-state cellular automata for density classification task exists", Physical Review Letters, Vol. 74, pp. 5148–5150, 1995.

[7] M.S. Capcarrère, M. Sipper, M. Tomassini, "Two-state, r=1 cellular automaton that classifies density", Physical Review Letters, Vol. 77, pp. 4969–4971, 1996.

[8] M. Tomassini, M. Venzi, "Evolving robust asynchronous cellular automata for the density task", Complex Systems, Vol. 13, pp. 185–204, 2002.

[9] S. Sahoo, P.P. Choudry, A. Pal, "Solutions on 1D and 2D density classification problem using programmable cellular automata", arXiv: 0902.2671 [nlin.CG], 2009.

[10] M. Sipper, Evolution of parallel cellular machines: The cellular programming approach (Lecture Notes in Computer Science), Springer, 1997.

[11] C.L.M. Martins, P.P.B. de Oliveira, "Evolving sequential combinations of elementary cellular automata rules", Advances in Artificial Life, Lecture Notes in Artificial Intelligence, Vol. 3630, pp. 461–470, 2005.

[12] A.R. Gabrielle, "The density classification problem for multi-states cellular automata", Advances in Artificial Life, Lecture Notes in Artificial Intelligence, Vol. 3630, pp. 443–452, 2005.

[13] C. Stone, L. Bull, "Solving the density classification task using cellular automaton 184 with memory", Complex Systems, Vol. 18, pp. 329–344, 2009.

[14] R. Alonso-Sanz, L. Bull, "One-dimensional coupled cellular automata with memory: initial investigations", Journal of Cellular Automata, Vol. 5, pp. 29–49, 2010.

[15] N. Fatès, "Stochastic cellular automata solutions to the density classification problem", Theory of Computing Systems, 2012, Vol. 53, pp. 223–242, 2013.

[16] H. Fukś, "Solution of the density classification problem with two cellular automaton rules", Physical Review E, Vol. 55, pp. 2081–2084, 1997.

[17] M. Sipper, "Computing with cellular automata: Three cases for nonuniformity", Physical Review E, Vol. 57, pp. 3589–3592, 1998.

[18] K.M. Lee, H. Xu, H.F. Chau, "Parity problem with a cellular automaton solution", Physical Review E, Vol. 64, pp. 267021–267024, 2001.

[19] C.L.M. Martins, P.P.B. de Oliveira, "Improvement of a result on sequencing elementary cellular automata rules for solving the parity problem", Electronic Notes in Theoretical Computer Science, Vol. 252, pp. 103–119, 2009.

[20] H. Betel, P.P.B. de Oliveira, P. Flocchini, "On the parity problem in one-dimensional cellular automata", Proceedings of the 18th International Workshop on Cellular Automata and Discrete Complex Systems and 3rd international symposium Journées Automates Cellulaires , EPTCS 90, pp. 110–126, 2012.