Mechanisms of Emergent Computation in Cellular Automata

Wim Hordijk, James P. Crutchfield, Melanie Mitchell

Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, 87501 NM, USA
email: {wim,chaos,mm}@santafe.edu

In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel (eds.), *Parallel Problem Solving from Nature-V*, 613-622, Springer-Verlag, 1998.

Abstract. We introduce a class of embedded-particle models for describing the emergent computational strategies observed in cellular automata (CAs) that were evolved for performing certain computational tasks. The models are evaluated by comparing their estimated performances with the actual performances of the CAs they model. The results show, via a close quantitative agreement, that the embedded-particle framework captures the main information processing mechanisms of the emergent computation that arise in these evolved CAs.

1 Introduction

In previous work we have used genetic algorithms (GAs) to evolve cellular automata (CAs) to perform computational tasks that require global coordination. The evolving cellular automata framework has provided a direct approach to studying how evolution (natural or artificial) can create dynamical systems that perform *emergent computation*; that is, how it can find dynamical systems in which the interaction of simple components with local information storage and communication gives rise to coordinated global information processing [3].

In [5, 6], we analyzed the evolutionary search process by which a genetic algorithm designed CAs to perform various tasks. In this paper we focus on how the behavior of evolved CAs implements the emergent computational strategies for performing these tasks. We develop a class of "embedded-particle" models to describe the computational strategies. To do this, we use the computational mechanics framework of Crutchfield and Hanson [2, 7], in which a CA's information processing is described in terms of regular domains, embedded particles, and particle interactions. We then evaluate this class of models by comparing their computational performance to that of the CAs they model. The results demonstrate, via a close quantitative agreement between the CAs and their respective models, that the embedded particle framework captures the functional features that emerge in the CAs' space-time behavior and that underlie the CAs' computational capability and evolutionary fitness.

The paper is organized as follows. Section 2 gives a brief overview of CAs and the notion of computation in CAs that we use here. Next, section 3 explains the computational mechanics framework for CAs in some detail and defines the notions of regular domains, embedded particles, and particle interactions. Section 4 then introduces the embedded-particle models, while section 5 describes how these models can be evaluated quantitatively. The results of these evaluations are provided in section 6.

2 CAs and Computation

This paper concerns one-dimensional binary-state CAs with spatially periodic boundary conditions. Such a CA consists of a one-dimensional lattice of N two-state machines ("cells"), each of which changes its state as a function, denoted ϕ, of the current states in a local neighborhood. In a one-dimensional CA, a neighborhood consists of a cell and its *radius* r neighbors on either side.

The lattice starts out with an initial configuration (IC) of cell states (0s and 1s) and this configuration changes at discrete time steps during which all cells are updated simultaneously according to the CA's rule ϕ. The rule ϕ can be expressed as a look-up table that lists, for each local neighborhood, the state which is taken by the neighborhood's central cell at the next time step.

One-dimensional binary-state cellular automata are perhaps the simplest examples of decentralized, spatially extended systems in which emergent computation can be observed. In our studies, a CA performing a computation means that the input to the computation is encoded as the IC, the output is decoded from the configuration reached at some later time step, and the intermediate steps that transform the input to the output are taken as the steps in the computation.

To date we have used a genetic algorithm (GA) to evolve one-dimensional, binary-state $r = 3$ CAs to perform a density-classification task [3, 5] and a synchronization task [6].

For the density classification task, the goal is to find a CA that decides whether or not the IC contains a majority of 1s (i.e., has high density). Let ρ_0 denote the density of 1s in the IC. If $\rho_0 > 1/2$, then within M time steps the CA should reach the fixed-point configuration of all 1s (i.e., all cells in state 1 for all subsequent iterations); otherwise, within M time steps it should reach the fixed-point configuration of all 0s. M is a parameter of the task that depends on the lattice size N. As an example, figure 1(a) shows a space-time diagram of a GA-evolved CA ϕ_{dens5} for the density classification task, starting with a randomly generated IC (in this case with $\rho_0 < 1/2$). Cells in state 1 are colored black, cells in state 0 are colored white. Time increases down the page.

For the synchronization task, the goal is to find a CA that, from any IC, settles down within M time steps to a periodic oscillation between an all-0s configuration and an all-1s configuration. Again, M is a parameter of the task that depends on N. Figure 1(c) shows a space-time diagram of a GA-evolved CA, denoted ϕ_{sync5}, for the synchronization task, again starting with a randomly generated IC.

Since a CA uses only local interactions, and thus has to propagate information across the lattice to achieve global coordination, both tasks require nontrivial computation. For example, in the synchronization task, the *entire* lattice has to be synchronized, which means the CA must resolve, using only local interactions, separate regions of the lattice that are locally synchronized but are out of phase with respect to one another.

We define the performance $\mathcal{P}_{N,I}(\phi)$ of a CA ϕ on a given task as the fraction of I randomly generated ICs on which ϕ reaches the desired behavior within M time steps on a lattice of length N. Here, we use $N = 149$, $M = 2N$ and $I = 10^4$.

3 Analysis of Evolved CAs

Due to the local nature of a CA's operations, it is typically very hard, if not impossible, to understand the CA's global behavior—in our case, the strategy for performing a computational task—by directly examining either the bits in the look-up table or the temporal sequence of 0-1 spatial configurations of the lattice.

Crutchfield and Hanson developed a method for detecting and analyzing the "intrinsic" computational components in the CA's space-time behavior in terms of regular domains, embedded particles, and particle interactions [2, 7]. This method is part of their *computational mechanics* framework for understanding information processing embedded in physical systems [1].

Briefly, a *regular domain* is a homogeneous region of space-time in which the same "pattern" appears. More formally, the spatial patterns in a regular domain can be described by a regular language that is mapped onto itself by the CA rule ϕ. An *embedded particle* is a spatially localized, temporally recurrent structure found at *domain boundaries*, i.e., where the domain pattern breaks down. When two or more particles "collide" they can produce an interaction result—e.g., another set of particles or a mutual annihilation.

In the space-time diagram of ϕ_{dens5} in figure 1(a), some of the domains, particles, and interactions are labeled. Three regular domains are readily apparent in ϕ_{dens5}'s space-time behavior: the all-white domain, the all-black domain, and a checkerboard domain (alternating white and black cells). The boundaries between these domains form the embedded particles, which can collide and interact with each other, creating other particles or simply annihilating.

Using computational mechanics, we can analyze the space-time behavior of evolved CAs in terms of these domains, particles, and interactions. In particular, the particles and their interactions can be made more explicit by suppressing the domains. Figure 1 shows examples for both ϕ_{dens5} and ϕ_{sync5}. For example, in ϕ_{sync5}'s space-time behavior (figure 1(c), there are two regular domains: the "synchronized" domain (the parts of the lattice which display the desired oscillation) and a second domain which has a zigzag pattern. Having identified these domains, we can build a filter based on the regular languages that represent the domains. Using this filter, the domains can be detected and then suppressed, revealing the domain boundaries. The filtered space-time diagram for ϕ_{sync5} is shown in figure 1(d), where the regular domains are mapped to 0s (white) and

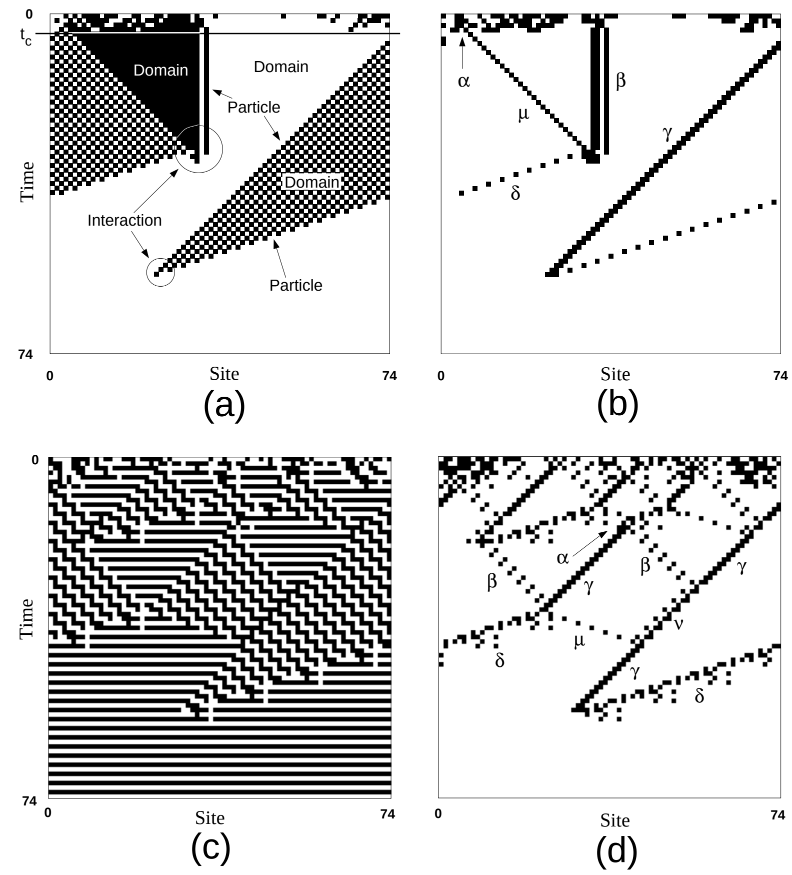

Fig. 1. (a) Space-time diagram of a GA-evolved CA ϕ_{dens5} that classifies high or low density of 1s in the initial configuration. Different regular domains, embedded particles, and particle interactions can be observed. (b) Filtered version of the space-time diagram in (a). Domains are mapped to white, domain boundaries to black. The different particles are labeled with Greek letters. (c) Space-time diagram of GA-evolved CA ϕ_{sync5} that, starting from a random IC, comes to global synchronization. (d) Filtered version of (c). Note that the same Greek letters in figures (b) and (d) do not denote the same particles.

the domain boundaries are mapped to 1s (black). A similar procedure for ϕ_{dens5} leads to the filtered space-time diagram in figure 1(b).

Briefly, the domain filter can be constructed in the following way. First, the (minimal) finite automata representing the regular languages of the domains need to be constructed. A series of transitions between states within each one of these finite automata represents an allowed sequence of site values in the domain configuration. That is, following these transitions is equivalent to remaining within a specific regular domain. Next, a finite-state transducer is built by connecting the separate finite automata of the regular domains by "wall" transitions. The latter represent transitions from one domain to another and so correspond to domain boundaries. Output symbols can be added to the wall transitions to give distinct labelings to the boundaries, if desired. As this transducer scans a CA configuration from left to right (say), it reads the consecutive cell states (0 or 1) and outputs a 0 or a 1 depending on whether it made an intradomain or a wall transition, respectively. In this way, a new, filtered configuration of "domain" and "wall" values is generated based on the presence of regular domains in the original configuration. Doing this for the configuration at each time step results in a filtered space-time diagram as shown in figures 1(b) and (d). (See [2] for details of transducer construction and use.)

The regular domains of ϕ_{dens5} and ϕ_{sync5} are readily apparent in their space-time diagrams, being easily identified by eye. The identification and construction of domain minimal automata are not always so straightforward. Computational mechanics provides an algorithm, called *ϵ-machine reconstruction*, for the recognition and identification of regular domains, including construction of their corresponding finite automaton representations, in spatio-temporal data [2, 4, 7].

Using computational mechanics, we can extract the relevant information about the domains, particles, and their interactions from the space-time diagrams of a given CA. A catalog of ϕ_{sync5}'s observed domains, particles and their temporal periodicities and velocities, and all possible particle interactions, is given in table 3. The *temporal periodicity* p of a particle is the number of time steps after which its spatial configuration repeats. The *velocity* v of a particle is the *displacement* d (the number of sites the particle has shifted in space after exactly one temporal period), divided by the temporal periodicity: $v = d/p$. For example, the particle μ in figure 1(d) has a temporal periodicity of $p = 2$ and after two time steps it has shifted $d = 6$ sites in space, so its velocity is $v = 6/2 = 3$.

Particles transfer information about properties of local regions across the lattice to distant sites. Particle collisions are the loci of information processing and result in either the creation of new information in the form of other particles or in annihilation. The computational mechanics analysis provides us with this particle-level description, which we claim captures the main mechanisms by which the CA transfers and processes local information to accomplish the emergent computation required by the task.

Table 1. A catalog of ϕ_{sync5}'s domains, particles and their properties, and interactions, some of which can be observed in the filtered space-time diagram of figure 1(c). An interaction result denoted by $\emptyset$ means that the two particles annihilate. The probabilities associated with the interaction results are also provided. If no explicit probability is given for an interaction result, it occurs with probability 1. These interaction result probabilities are explained in section 4.

ϕ_{sync5} **Particle Catalog**				
Domains Λ				
Label	Regular language			
Λ^s	$\Lambda_0^s = 0^4 0^*$, $\Lambda_1^s = 1^4 1^*$			
Λ^z	$\Lambda_0^z = (0001)^*$, $\Lambda_1^z = (1110)^*$			
Particles P				
Label	Wall	p	d	v
α	$\Lambda_1^s \Lambda_0^s$	-	-	-
β	$\Lambda_0^z \Lambda_0^s$, $\Lambda_1^z \Lambda_1^s$	2	2	1
γ	$\Lambda_0^s \Lambda_1^z$, $\Lambda_1^s \Lambda_0^z$	2	-2	-1
δ	$\Lambda_0^z \Lambda_1^s$, $\Lambda_1^z \Lambda_0^s$	4	-12	-3
μ	$\Lambda_0^s \Lambda_0^z$, $\Lambda_1^s \Lambda_1^z$	2	6	3
ν	$\Lambda_0^z \Lambda_1^z$, $\Lambda_1^z \Lambda_0^z$	2	-2	-1
Interactions I				
Type	Interaction	Interaction		
decay	$\alpha \to \gamma + \beta$			
react	$\beta + \gamma \overset{0.84}{\to} \delta + \mu$	$\beta + \gamma \overset{0.16}{\to} \nu$		
	$\mu + \delta \to \gamma + \beta$	$\nu + \delta \to \beta$		
	$\mu + \nu \to \gamma$			
annihilate	$\mu + \beta \to \emptyset$	$\gamma + \delta \to \emptyset$		

4 A Formal Model of Computational Strategies

To formalize the notion of computational strategy in a CA, and the resulting emergent computation, we model the CA's behavior using only the notions of domains, particles, and interactions, moving away from the underlying individual cells in a configuration to a higher level description. The resulting embedded-particle model employs a number of simplifying assumptions.

Before the assumptions are stated, we first define the *condensation time* t_c as the *first* time step at which the lattice can be completely described in terms of domains and particles. To identify this condensation time in a CA's space-time diagram, we can extend the transducer that is used to filter out domains to also recognize particles. In terms of the transitions in the transducer, particles are specific sequences of wall transitions that lead from states in one domain to those in another. Including these transition paths in the set of allowed transitions, the transducer can recognize both regular domains *and* the particles. (A more detailed example of how to extend the transducer to incorporate the particles can be found in [8].) Using this extended transducer, the condensation time t_c is then defined as the first time step at which filtering the lattice does not generate any disallowed transitions.

The occurrence of the condensation time is illustrated in figure 1(a) for ϕ_{dens5}. The condensation time ($t_c = 4$ in this particular case) is marked by the solid line. It is the time step at which the nondomain/particle structures at the first few time steps have died out and there remain only domains and particles. The particular value of t_c for a given CA depends on the IC, but we can estimate the average condensation time $\overline{t_c}$ for a given rule by sampling t_c over a large set of random ICs. The measured value of $\overline{t_c}$ for various rules will be used later when we evaluate the particle models. For ϕ_{dens5} on a lattice size of 149, $\overline{t_c} \approx 12$.

As a first simplifying assumption of the model, we ignore the details of the space-time dynamics up to t_c and assume that the net effect of this behavior is to generate some distribution of particles of various types, probabilistically located in the configuration at time t_c. In other words, we assume that beyond generating this distribution at t_c, the initial "pre-condensation" dynamics are not relevant to predicting the performance of the CA.

To estimate this particle probability distribution at t_c, we again employ the extended transducer that is used for determining t_c. Using a large set of random ICs (in our case 10^4), the CA is run on each one up to the actual condensation time, i.e., up to the first time step at which the extended transducer only goes through domain and wall transitions while scanning the configuration. Next, the number of times each of these transitions are taken is counted and this is averaged over the set of 10^4 condensation-time configurations. From these counts, a *transition probability* for each allowed transition can be calculated. The extended transducer, together with the estimated transition probabilities, provides an approximation of the actual particle probability distribution at $\overline{t_c}$.

As a second simplifying step, we assume that all particles have zero width, even though, as can be seen in figure 1, particles actually have varying widths.

As a third simplification, we allow interactions only between pairs of particles. No interactions involving more than two particles are included in the model.

A fourth simplifying assumption we make is that particle interactions are instantaneous. As can be seen in figure 1, when two particles collide and interact with each other, typically the interaction takes time to occur—for some number of time steps the configuration cannot be completely decomposed into domains and particles. In the embedded-particle model when two particles collide they are immediately replaced by the interaction result.

The interaction result of a particle collision is determined by the phases that both particles are in at the time of their collision. As a fifth simplifying assumption, we approximate this relative phase dependence by a stochastic choice of interaction result. To determine an interaction result, the model uses a particle catalog (e.g., as shown in table 3 for ϕ_{sync5}) that contains interaction-result probabilities. For each possible pair of particle types, this catalog lists all the interaction results that these two particles can produce, together with the probability that each particular result occurs. These probabilities can be estimated empirically by simply counting, over a set of 10^4 random ICs, how often each interaction result occurs in the space-time diagram. In the model, when two particles collide, the catalog is consulted and an interaction result is determined at random according to these probabilities. For example, in table 3, the $\beta + \gamma$ interaction has two possible outcomes. Each is given with its estimated probability of occurrence.

In summary, the embedded-particle model of a CA's computational strategy consists of:

1. A catalog of possible domains, particle types, and a set of pairwise particle interactions and results, along with the interaction-result probabilities for each.
2. An approximate particle probability distribution at $\overline{t_c}$, represented by the domain-particle transducer with estimated transition probabilities.

The first of these two components is given in table 3 for ϕ_{sync5}.

5 Evaluating the Embedded-Particle Model

One way to evaluate an embedded-particle model is to compare its task performance to that of the CA it models. A close agreement would support our claim that the embedded-particle model is a good description of the CA's computational strategy. This is a quantitative complement to the computational mechanics analysis which establishes the structural roles of domains, particles, and interactions.

To run the model, we start by generating an initial particle configuration at $\overline{t_c}$, according to the particle probability distribution in the model (i.e., we ignore the "pre-condensation" phase). Since this probability distribution is represented by the transducer with estimated transition probabilities, we now use this transducer as a domain-particle *generator* instead of a recognizer. Starting in a randomly chosen state (according to an empirically determined probability distribution), the transducer traverses N lattice sites, and at each site it chooses a transition according to the estimated probabilities and then outputs either a domain or a particle symbol, according to which transition was chosen.

This process creates an initial particle configuration by placing a number of particles of various types in the configuration at random locations according to the approximated particle probability distribution. The result is that we know for each particle in the initial particle configuration ($t = \overline{t_c}$) its type and also its velocity and spatial location. It is then straightforward to calculate geometrically at what time step t_i the first interaction between two particles will occur. The particle catalog is then consulted and the result of this particular interaction is determined. The two interacting particles are then replaced by the interaction result, yielding a new particle configuration at time step t_i.

This process of calculating the next interaction time and replacing the interacting particles with their interaction result is iterated either until there are no particles left in the lattice (i.e., they have all annihilated one another) or until a given maximum number ($M - \overline{t_c}$) of time steps is reached, whichever occurs first. We refer to this iteration process as the model's ballistic particle dynamics.

Since the embedded-particle model contains information about which domains are associated with which particles, we can keep track of the domains between the particles at each time step while the model is run. Thus, if all particles eventually annihilate one another, we know which domain is left occupying the entire lattice. This way, we can check whether running the model resulted in the correct final behavior for the given task.

6 Results

We can now evaluate the performance of an embedded-particle model by running it on a large number I of initial particle configurations at $\overline{t_c}$ and calculating the fraction over which it displays the correct behavior; i.e., that it settles down to the correct domain within the maximum number of allowed time steps. This performance estimate can then be compared to the actual performance of the CA from which the model was built. The parameters used here are $N = 149$, $M = 2N$, and $I = 10^4$.

Figure 2(a) shows the results of comparing the average performance of five evolved CAs for density classification with the performance predicted by their respective models. In all cases the average performance is calculated over 10 sets of 10^4 random ICs (in case of the actual CAs) or initial particle configurations (in case of the models). The five CAs appeared at different generations during one particular run of the GA on the density classification task. Each successive CA implemented an improved computational strategy for performing the task, reflected in the successive increases in performance. The evolutionary process that gave rise to these CAs is discussed in more detail in [5].

Figure 2(b) shows similar results for five CAs that appeared during a GA run on the synchronization task (note that the first one, ϕ_{sync1}, had performance 0). These CAs were described in [6]. Table 2 shows the performance data for all ten CAs, including the standard deviations of the measured average performances. Recall that typical space-time behavior of ϕ_{dens5} and ϕ_{sync5} was shown in figures 1(a) and (c), respectively.

Generally, there is very good agreement between the CA and embedded-particle model performances. Furthermore, most of the discrepancies can be traced back directly to the simplifying assumptions underlying the particle models. For example, the discrepancies for ϕ_{sync4} and ϕ_{sync5} are partly caused by the "zigzag" domain having a spatial periodicity of four (see figure 1(c)). Due to the periodic boundary conditions on a lattice of size 149, a CA can never settle down to a configuration containing only the zigzag domain, since 149 is not divisible by 4. However, this can happen in the embedded-particle model, since it ignores the spatial periodicity of domains. Such configurations are counted as incorrect behavior in the models and this results in slightly lower predicted versus actual performances. Furthermore, for the synchronization CAs some possible particle collisions have multiple interaction results, depending on the relative

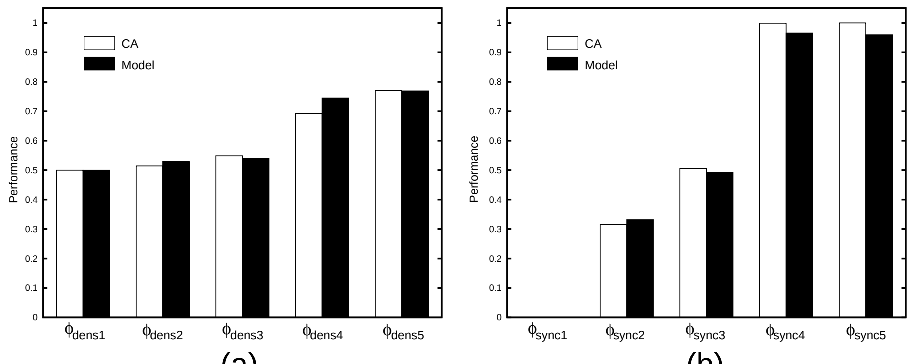

Fig. 2. Comparison of the average CA performances (labeled "CA") and the model-estimated average performances (labeled "Model") for (a) five evolved density classification CAs and (b) five evolved synchronization CAs.

particle phases at the time of collision. Since this is modeled by a probabilistic choice of interaction results, the particle models for the synchronization CAs are slightly less accurate than the models for the density classification CAs, where all collisions have just one possible interaction result.

The generally good agreement between the performance of a model and that of the corresponding CA demonstrates the promise of the embedded-particle framework for explaining the spatially extended computation performed by CAs. The observed discrepancies, which will be explained in more detail elsewhere, demonstrate where our simplifying assumptions fail. We can improve on the model's agreement with these and other CAs by incorporating additional features, such as taking particle phases into account.

7 Conclusions

Emergent computation in decentralized spatially extended systems, such as in CAs, is still not well understood. In previous work we have used an evolutionary approach to search for CAs that are capable of performing computations that require global coordination. We have also qualitatively analyzed the emergent "computational strategies" of the evolved CAs in terms of domains, particles, and particle interactions. The embedded-particle models described here provide a means to more rigorously formalize the notion of "emergent computational strategy" in spatially extended systems and to make quantitative predictions about the computational behavior and evolutionary fitness of the evolved CAs. This is an essential, quantitative part of our overall research program—to understand how natural spatially extended systems can perform globally coordinated computations and how evolutionary processes can give rise to systems with sophisticated emergent computational abilities.

Table 2. Comparison of the average CA performances and the model-predicted average performances for five evolved density-classification CAs and five evolved synchronization CAs. The averages are calculated over 10 sets of 10^4 ICs each. The standard deviations are given in parentheses. The $\overline{t_c}$ used for each model is given, as is the hexadecimal code for each CA's ϕ, with the most significant bit being associated with the neighborhood `0000000`.

		$\mathcal{P}_{149,10^4}(\phi)$			
Rule	Hex	CA	Model	Difference	$\overline{t_c}$
ϕ_{dens1}	`04004489 020107FF`	0.5000	0.5000	0%	10
	`6B9F7793 F9FFBF7F`	(0)	(0)		
ϕ_{dens2}	`04004581 00000FFF`	0.5145	0.5291	2.8%	15
	`6B9F7793 7DFFFF7F`	(0.0026)	(0.0034)		
ϕ_{dens3}	`05004581 00000FFF`	0.5487	0.5405	1.5%	34
	`6B9F7793 7FBFFF5F`	(0.0027)	(0.0055)		
ϕ_{dens4}	`05004581 00000FBF`	0.6923	0.7447	7.6%	29
	`6B9F7593 7FBDF77F`	(0.0055)	(0.0070)		
ϕ_{dens5}	`05040587 05000F77`	0.7702	0.7689	0.2%	12
	`03775583 7BFFB77F`	(0.0036)	(0.0052)		
ϕ_{sync1}	`F8A19CE6 B65848EA`	0.0000	0.0000	0%	28
	`D26CB24A EB51C4A0`	(0)	(0)		
ϕ_{sync2}	`F8A1AE2F CF6BC1E2`	0.3161	0.3316	4.9%	40
	`D26CB24C 3C266E20`	(0.0033)	(0.0047)		
ϕ_{sync3}	`F8A1AE2F CE6BC1E2`	0.5063	0.4923	1.4%	29
	`C26CB24E 3C226CA0`	(0.0059)	(0.0037)		
ϕ_{sync4}	`F8A1CDAA B6D84C98`	0.9991	0.9655	3.4%	33
	`5668B64A EF10C4A0`	(0.0002)	(0.0019)		
ϕ_{sync5}	`FEB1C6EA B8E0C4DA`	1.0000	0.9596	4.0%	21
	`6484A5AA F410C8A0`	(0)	(0.0021)		

Acknowledgments

This research was supported by the Santa Fe Institute under ONR grant N00014-95-1-0975, NSF grants IRI-9320200 and IRI-9705853, and DOE grant DE-FG03-94ER25231.

References

1. Crutchfield, J. P.: The calculi of emergence: Computation, dynamics, and induction. Physica D **75** (1994) 11–54.
2. Crutchfield, J. P., Hanson, J. E.: Turbulent pattern bases for cellular automata. Physica D **69** (1993) 279–301.
3. Crutchfield, J. P., Mitchell, M.: The evolution of emergent computation. Proceedings of the National Academy of Sciences, USA 92 **23** (1995) 10742–10746.
4. Crutchfield, J. P., Young, K.: Inferring statistical complexity. Physical Review Letters **63** (1989) 105–108.

5. Das, R., Mitchell, M., Crutchfield, J. P.: A genetic algorithm discovers particle-based computation in cellular automata. Parallel Problem Solving from Nature—PPSN III, Davidor, Y., Schwefel, H.-P., Männer, R., eds. (1994) 244–353.
6. Das, R., Crutchfield, J. P., Mitchell, M., Hanson, J. E.: Evolving globally synchronized cellular automata. Proceedings of the Sixth International Conference on Genetic Algorithms, Eshelman, L. ed., (1995) 336–343.
7. Hanson, J. E., Crutchfield, J. P.: The attractor-basin portrait of a cellular automaton. Journal of Statistical Physics **66** (5/6) (1992) 1415–1462.
8. Hanson, J. E., Crutchfield, J. P.: Computational Mechanics of Cellular Automata: An Example. Physica D **103** (1997) 169–189.
9. Wolfram, S.: Cellular Automata and Complexity. Addison-Wesley, 1994.