Evolving Globally Synchronized Cellular Automata

Rajarshi Das[1,2], James P. Crutchfield[3], Melanie Mitchell[1], and James E. Hanson[1]

To appear in the *Proceedings of the Sixth International Conference on Genetic Algorithms.*
April 5, 1995

Abstract

How does an evolutionary process interact with a decentralized, distributed system in order to produce globally coordinated behavior? Using a genetic algorithm (GA) to evolve cellular automata (CAs), we show that the evolution of spontaneous synchronization, one type of emergent coordination, takes advantage of the underlying medium's potential to form embedded particles. The particles, typically phase defects between synchronous regions, are designed by the evolutionary process to resolve frustrations in the global phase. We describe in detail one typical solution discovered by the GA, delineating the discovered synchronization algorithm in terms of embedded particles and their interactions. We also use the particle-level description to analyze the evolutionary sequence by which this solution was discovered. Our results have implications both for understanding emergent collective behavior in natural systems and for the automatic programming of decentralized spatially extended multiprocessor systems.

1. Introduction

The spontaneous synchronization of independent processes is one of the more widely observed dynamical behaviors in nature. In many such phenomena, synchronization serves a vital role in the collective function of the constituent processes. The spiral waves exhibited during the developmental and reproductive stages of the *Dictyostelium* slime mold [4], the morphogenesis of embryonic structures in early development [11], the synchronized oscillations of neural assemblies which have been thought to play a significant role in encoding information [8], and the marked seasonal variation in the breeding activity of sexually reproducing populations are just a few examples of the temporal emergence of global synchronization.

The importance of global synchronization has been recognized for decades outside of natural science as well. From the earliest days of analog and digital computer design, the functioning of an entire computing device has been critically dependent on achieving global synchronization among the individual processing units. Typically, the design choice has been to use a central controller which coordinates the behavior of all parts of the device. In this way, the interaction of individual units is modulated so that the transfer of information among the units is meaningful.

But what if the option of a central controller is not available? Given the widespread appearance of synchronization in decentralized and spatially extended systems in nature, evidently evolution has successfully overcome this problem. Evolution has effectively taken advantage of

[1]Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, New Mexico, U.S.A. 87501.
Email: {raja,mm,hanson}@santafe.edu

[2]Computer Science Department, Colorado State University, Fort Collins, Colorado, U.S.A. 80523

[3]Physics Department, University of California, Berkeley, CA, U.S.A. 94720-7300.
Email: chaos@gojira.berkeley.edu

the spatially local dynamics in its production of organisms which, on the one hand, consist of potentially independent subsystems, but whose behavior and survival, on the other hand, rely on emergent synchronization. These observations leave us with an unanswered but biologically significant question. By what mechanisms does evolution take advantage of nature's inherent dynamics?

We explore this question in a simple framework by coupling an evolutionary process—a genetic algorithm (GA)—to a population of behaviorally rich dynamical systems—one-dimensional cellular automata (CAs). In this scheme, survival of an individual CA is determined by its ability to perform a synchronization task.

Recent progress in understanding the intrinsic information processing in spatially extended systems such as CAs has provided a new set of tools for the analysis of temporally and evolutionarily emergent behavior [1, 6, 7]. Beyond describing solutions to the computational task, in this paper we use these tools to analyze in some detail the individual CA behavioral mechanisms responsible for increased fitness. We also analyze how these mechanisms interact with selection to drive the CA population to increasingly sophisticated synchronization strategies.

2. Cellular Automata

CAs are arguably the simplest example of decentralized, spatially extended systems. In spite of their simple definition they exhibit rich dynamics which over the last decade have come to be widely appreciated [5, 12, 13]. A CA consists of a collection of time-dependent variables s_t^i, called the *local states*, arrayed on a lattice of N sites (or *cells*), $i = 0, 1, ..., N-1$. We will take each to be a Boolean variable: $s_t^i \in \{0, 1\}$. The collection of all local states is called the *configuration*: $\mathbf{s}_t = s_t^0 s_t^1 \cdots s_t^{N-1}$. $\mathbf{s}_0$ denotes an initial configuration (IC). Typically, the equation of motion for a CA is specified by a look-up table ϕ that maps a site's *neighborhood* η_t^i to a new local state for that site at the next time step : $s_{t+1}^i = \phi(\eta_t^i)$, where $\eta_t^i = s_t^{i-r} \cdots s_t^i \cdots s_t^{i+r}$ and r is called the CA's *radius*. (In contexts in which i and t are not relevant, we will simply use η with no sub- or superscripts to denote a neighborhood.) The *global equation of motion* Φ maps a configuration at one time step to the next: $\mathbf{s}_{t+1} = \Phi(\mathbf{s}_t)$, where it is understood that the local function ϕ is applied simultaneously to all lattice sites. It is also useful to define an operator $\boldsymbol{\Phi}$ that operates on a set of configurations or substrings of configurations—that is, on a formal language—by applying Φ separately to each member of the set.

The CAs in the GA experiments reported below had $r = 3$, $N = 149$, and spatially periodic boundary conditions: $s_t^i = s_t^{i+N}$.

3. The Synchronization Task

Our goal is to find a CA that, given any initial configuration $\mathbf{s}_0$, within M time steps reaches a final configuration that oscillates between all 0s and all 1s on successive time steps: $\Phi(1^N) = 0^N$ and $\Phi(0^N) = 1^N$. M, the desired upper bound on the synchronization time, is a parameter of the task that depends on the lattice size N. This is perhaps the simplest non-trivial synchronization task for a CA.

The task is nontrivial since synchronous oscillation is a global property of a configuration, whereas a small-radius (e.g., $r = 3$) CA employs only local interactions mediated by the sites' neighborhoods. Thus, while the locality of interaction can directly lead to regions of local synchrony, it is more difficult to design a CA that will guarantee that spatially distant regions are in phase. Since regions that are not in synchrony can be distributed throughout the lattice, a successful CA must transfer information over large space-time distances ($\approx N$) to remove phase defects separating regions that are locally synchronous, in order to produce a globally synchronous configuration.

For reference, consider a simple benchmark radius 3 CA ϕ_{osc}, which is a naive candidate solution with $\Phi_{osc}(1^N) = 0^N$ and $\Phi_{osc}(0^N) = 1^N$. Its look-up table is defined by: $\phi_{osc}(\eta) = 1$ if $\eta = 0^7$; $\phi_{osc}(\eta) = 0$ otherwise.

We defined the *performance* $\mathcal{P}_K^N(\phi)$ of a given CA ϕ on a lattice of size N to be the fraction of K randomly chosen initial configurations on which ϕ produces correct final behavior. We then measured $\mathcal{P}_{10^4}^N(\phi_{osc})$ to be 0.54, 0.09, and 0.02, for $N = 149$, 599, and 999, respectively. (The behavior of a CA on these three values of N give a good idea of how the behavior scales with lattice size.)

ϕ_{osc} is not a successful solution precisely because it is unable to remove phase defects. A more sophisticated CA must be found to produce the desired collective behavior. It turned out that the successful solutions discovered by our GA were surprisingly interesting and complex.

4. Details and Results of GA Experiments

We used a GA, patterned after that in our previous work on density classification [2, 3, 10], to evolve CAs that perform the synchronization task. The GA begins with a population of P randomly generated "chromosomes"—bit strings encoding CAs by listing each ϕ's output bits in lexicographic order of neighborhood configuration. For binary $r = 3$ CAs, the chromosomes are of length $128 (= 2^{2r+1})$. The size of the space the GA searches is thus 2^{128}—far too large for any kind of exhaustive search.

With the lattice size fixed at $N = 149$, the *fitness* $F_I(\phi)$ of a CA in the population is calculated by randomly choosing I ICs that are uniformly distributed over $\rho_0 \in [0.0, 1.0]$ (where ρ_0 denotes the fraction of 1s in $\mathbf{s}_0$) and iterating ϕ on each IC for a maximum of M time steps. $F_I(\phi)$ is the fraction of the I ICs on which ϕ produces the correct final dynamics: an oscillation between 0^N and 1^N. No partial credit is given for incompletely synchronized final configurations.

In our experiments, we used $F_I(\phi)$ as an estimate of $\mathcal{P}_K^N(\phi)$ with $I \ll K$ and $N = 149$. It should be pointed out that sampling ICs in $F_I(\phi)$ with uniform distribution over $\rho_0 \in [0.0, 1.0]$ is highly skewed with respect to the unbiased distribution of ICs in $\mathcal{P}_K^N(\phi)$, which is binomially distributed over $\rho_0 \in [0.0, 1.0]$ and very strongly peaked at $\rho_0 = 1/2$. Preliminary experiments indicated that while both kinds of distributions allowed the GA to find high performance rules, the uniform distribution helped the GA to make more rapid progress in early generations.

In each generation the GA goes through the following steps. (i) A new set of I ICs is generated from the uniform distribution. (ii) $F_I(\phi)$ is calculated for each ϕ in the population. (iii) The population is ranked in order of fitness; equally fit CAs are ranked randomly relative to one another. (iv) E of the highest fitness ("elite") CAs are copied without modification to the next generation. (v) The remaining $(P - E)$ CAs for the next generation are formed by single-point crossovers between pairs of elite CAs chosen randomly with replacement. The offspring from each crossover are each mutated m times, where a mutation consists of flipping a randomly chosen bit in a chromosome. This defines one generation of the GA; it is repeated G times for one GA run.

$F_I(\phi)$ is a random variable since its value depends on the particular set of I ICs selected to evaluate ϕ. Thus, a CA's fitness varies stochastically from generation to generation. For this reason, we choose a new set of ICs at each generation

For our experiments we set $P = 100$, $E = 20$; $I = 100$, $m = 2$; and $G = 50$. M was chosen from a Poisson distribution with mean 320 (slightly greater than $2N$). Varying M prevents selecting CAs that are adapted to a particular M. A justification of these parameter settings is given in [9].

We performed a total of 65 GA runs. Since $F_{100}(\phi)$ is only a rough estimate of performance, we more stringently measured the quality of the GA's solutions by calculating $\mathcal{P}^N_{10^4}(\phi)$ with $N \in \{149, 599, 999\}$ for the best CAs in the final generation of each run. In 20% of the runs the GA discovered successful CAs ($\mathcal{P}^N_{10^4} = 1.0$). More detailed analysis of these successful CAs showed that although they were distinct in detail, they used similar strategies for performing the synchronization task. Interestingly, when the GA was restricted to evolve CAs with $r = 1$ and $r = 2$, all the evolved CAs had $\mathcal{P}^N_{10^4} \approx 0$ for $N \in \{149, 599, 999\}$. (Better performing CAs with $r = 2$ can be designed by hand.) Thus $r = 3$ appears to be the minimal radius for which the GA can successfully solve this problem.

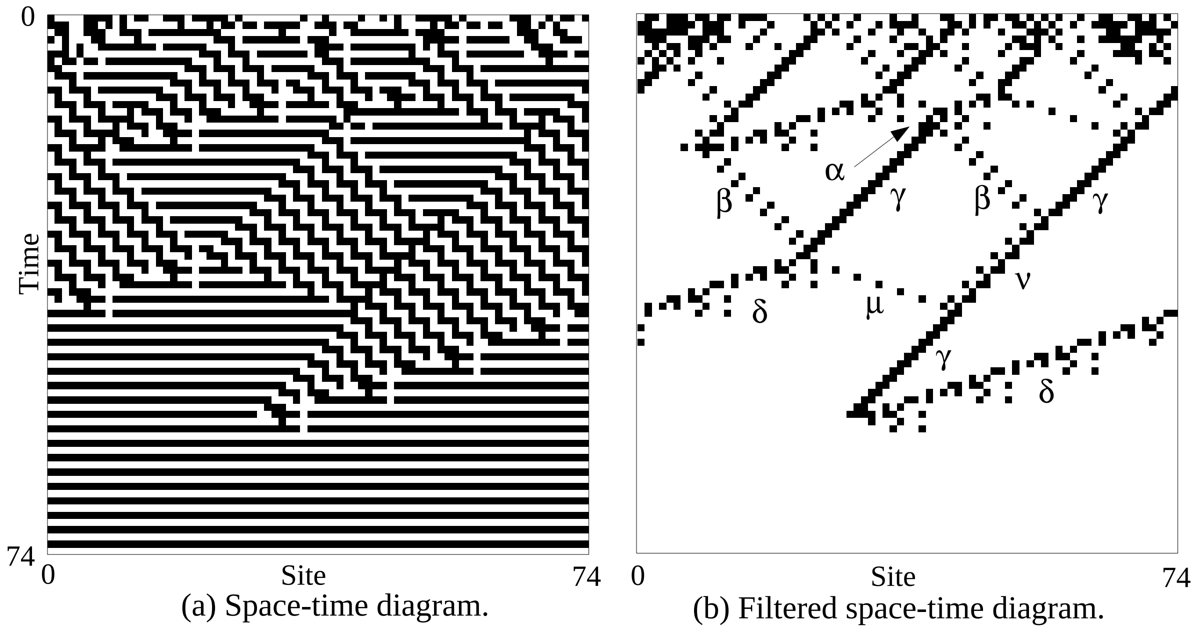

Figure 1: (a) Space-time diagram of ϕ_{sync} starting with a random initial condition. (b) The same space-time diagram after filtering with a spatial transducer that maps all domains to white and all defects to black. Greek letters label particles described in the text.

Figure 1a gives a space-time diagram for one of the GA-discovered CAs with 100% performance, here called ϕ_{sync}. This diagram plots 75 successive configurations on a lattice of size $N = 75$ (with time going down the page) starting from a randomly chosen IC, with 1-sites colored black and 0-sites colored white. In this example, global synchronization occurs at time step 58.

How are we to understand the strategy employed by ϕ_{sync} to reach global synchronization? Notice that, under the GA, while crossover and mutation act on the local mappings comprising a

CA look-up table (the "genotype"), selection is performed according to the dynamical behavior of CAs over a sample of ICs (the "phenotype"). As is typical in real-world evolution, it is very difficult to understand or predict the phenotype from studying the genotype. So we are faced with a problem familiar to biologists and increasingly familiar to evolutionary computationalists: how do we understand the successful complex systems (e.g., ϕ_{sync}) that our GA has constructed?

5. Computational Mechanics of Cellular Automata

Our approach to understanding the computation performed by the successful CAs is to adopt the "computational mechanics" framework for CAs developed by Crutchfield and Hanson [1, 6, 7]. This framework describes the "intrinsic computation" embedded in the temporal development of the spatial configurations in terms of *domains*, *particles*, and *particle interactions*. A domain is, roughly, a homogeneous region of space-time in which the same "pattern" appears. For example, in Figure 1a, two types of domains can be seen: regions in which the all-1s pattern alternates with the all-0s pattern, and regions of jagged black diagonal lines alternating with jagged white diagonal lines. The notion of a domain can be formalized by describing the domain's pattern using the minimal deterministic finite automaton (DFA) that accepts all and only those spatial configurations that are consistent with the pattern.

Since the domains in Figure 1a are described by simple DFAs, they represent relatively simple patterns. Once the domains have been detected, nonlinear filters can be constructed to filter them out, leaving just the deviations from those regularities (Figure 1b). The resulting filtered space-time diagram reveals the propagation of domain boundaries. If these boundaries remain spatially localized over time, then they are called particles. (For the discussion later, we have labeled some of the particles in Figure 1b with Greek letters.) These "embedded" particles are one of the main mechanisms for carrying information over long space-time distances. This information might indicate, for example, the partial result of some local processing which has occurred elsewhere at an earlier time. Logical operations on the information particles carry are performed when they interact. The collection of domains, domain boundaries, particles, and particle interactions for a CA represents the basic information-processing elements embedded in the CA's behavior—the CA's "intrinsic" computation.

In the example presented in Figure 1a the domains and particles are easy to see by inspection. However, often CAs produce space-time behaviors in which regularities exist but are not so easily discernible. Crutchfield and Hanson have developed automatic induction methods for "reconstructing" domains in space-time data and for building the nonlinear filters that reveal the hidden particles, allowing the intrinsic computation to be analyzed. In Figure 1b, the filtering not only allows us to determine the location of the particles in the space-time diagram, but it also helps in readily identifying the spatial and temporal features of the particles.

To perform the synchronization task, ϕ_{sync} produces local regions of synchronization (alternating 1^* and 0^* patterns, where w^* represents some number of repetitions of string w). In many cases, adjacent synchronized regions are out of phase. Wherever such phase defects occur, ϕ_{sync} resolves them by propagating particles—the boundaries between the synchronized regions and the jagged region—in opposite directions. Encoded in ϕ_{sync}'s look-up table are interactions involving these particles that allow one or the other competing synchronized region to annihilate the other and to itself expand. Similar sets of interactions continue to take place among the remaining synchronized regions until the entire configuration has one coherent phase.

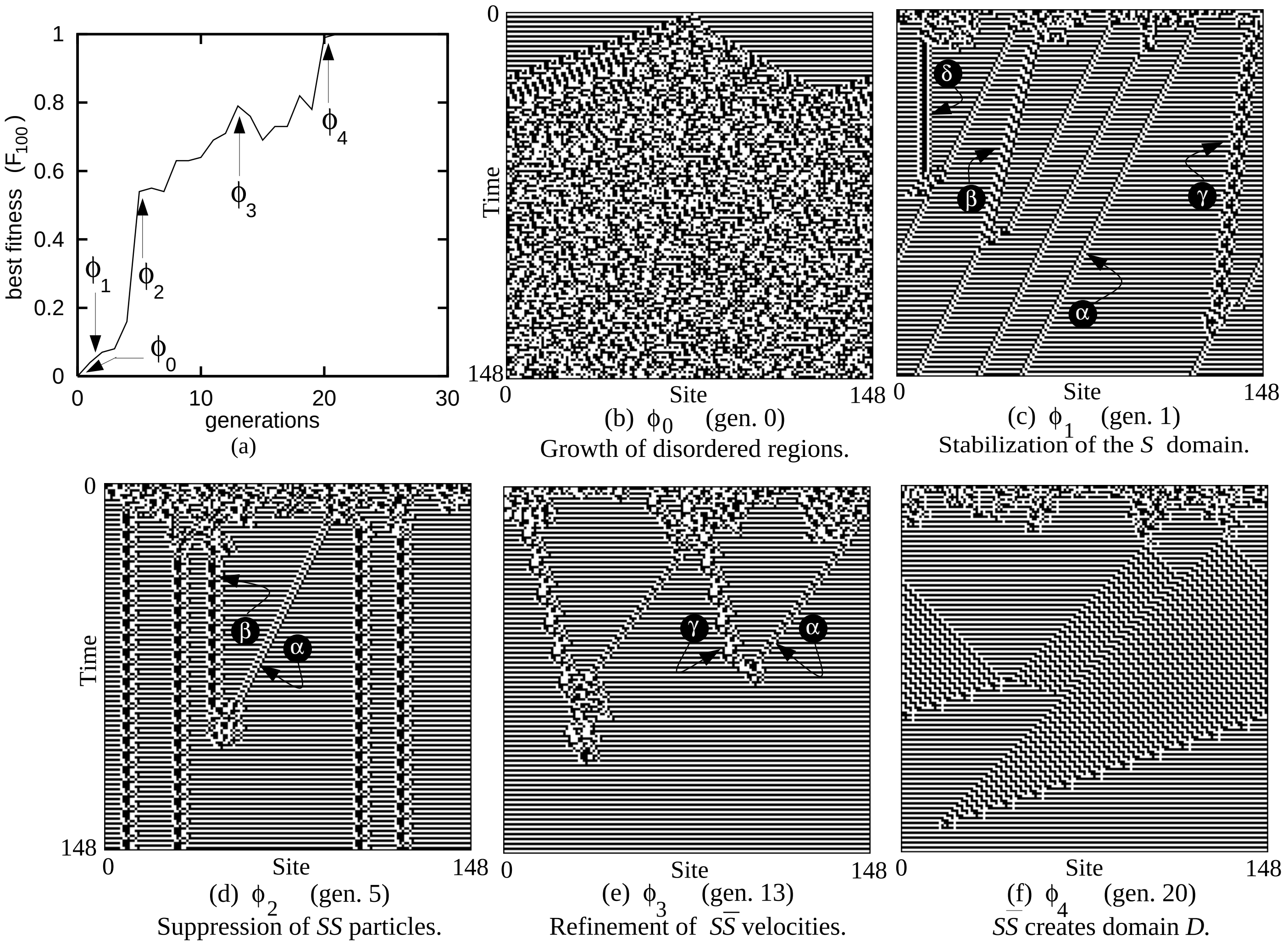

Figure 2: Evolutionary history of ϕ_{sync}: (a) F_{100} versus generation for the most fit CA in each population. The arrows indicate the generations in which the GA discovered each new significantly improved strategy. (b)–(f) Space-time diagrams illustrating the behavior of the best ϕ at each of the five generations marked in (a). The ICs are disordered except for (b), which consists of a single 1 in the center of a field of 0s. The same Greek letters in different figures represent different types of particles.

In the next section we will make this intuitive description more rigorous. In particular, we will describe the evolutionary path by which our GA discovered ϕ_{sync}, using the computational mechanics framework to analyze the mechanisms embedded in the increasingly fit CAs created by the GA as a run progresses.

6. The Evolution to Synchronization

Figure 2a plots the best fitness in the population versus generation for the first 30 generations of the run in which ϕ_{sync} was evolved. The figure shows that, over time, the best fitness in the population is marked by periods of sharp increases. Qualitatively, the overall increase in fitness can be divided into five "epochs". The first epoch starts at generation 0 and each of the following epochs corresponds to the discovery of a new, significantly improved strategy for performing the synchronization task. Similar epochs were seen in most of the runs resulting in CAs with 100% performance. In Figure 2a, the beginning of each epoch is labeled with the best CA in the population at that generation.

Epoch 0: Growth of Disordered Regions. To perform the synchronization task, a CA ϕ must have $\phi(0^7) = 1$ and $\phi(1^7) = 0$. These mappings insure that local regions will have the desired oscillation. Such a synchronized region is a domain—denote it S—with a temporal periodicity of two: $0^* = \mathbf{\Phi}(1^*)$, and $1^* = \mathbf{\Phi}(0^*)$. Since the existence of the S domain is guaranteed by fixing just two bits in the chromosome, approximately 1/4 of the CAs in a random initial population have S.

However, S's stability under small perturbations depends on other output bits. For example, ϕ_0 is a generation 0 CA with these two bits set correctly, but under ϕ_0 a small perturbation in S leads to the creation of a disordered region. This is shown in Figure 2b, where the IC contains a single 1 at the center site. In the figure, the disordered region grows until it occupies the whole lattice. This behavior is typical of CAs in generation 0 that have the two end bits set correctly. Increasing the number of perturbation sites in S leads to a simultaneous creation of disordered regions all over the lattice, which subsequently merge to eliminate synchronous regions. Thus, CAs like ϕ_0 have zero fitness unless one of the test ICs has $\rho_0 = 0.0$ or $\rho_0 = 1.0$.

Epoch 1: Stabilization of the Synchronous Domain. The best CA at generation 1, ϕ_1, has $F_{100} \approx 0.04$, indicating that it successfully synchronizes on only a small fraction of the ICs. Although this is only a small increase in fitness, the space-time behavior of ϕ_1 (Figure 2c) is very different from that of ϕ_0. Unlike ϕ_0, ϕ_1 eliminates disordered regions by expanding (and thereby stabilizing) local synchronous domains. The stability of the synchronous regions comes about because ϕ_1 maps all the eight neighborhoods with six or more 0s to 1, and seven out of eight neighborhoods with six or more 1s to 0. Under our lexicographic ordering, most of these bits are clustered at the left and right ends of the chromosome. This means it is easy for the crossover operator to bring them together from two separate CAs to create CAs like ϕ_1.

Figure 2c shows that under ϕ_1, the synchronous regions fail to occupy the entire lattice. A significant number of constant-velocity particles (here, boundaries between adjacent S domains) persist indefinitely and prevent global synchronization from being reached. Due to the temporal periodicity of the S domains, the two adjacent S domains at any boundary can be either in-phase or out-of-phase with each other. We will represent the in-phase and the out-of-phase defects between two S domains as SS and $S\overline{S}$ respectively. A more detailed analysis of ϕ_1's space-time behavior shows that it supports one type of stable $S\overline{S}$ particle, α, and three different types of stable SS particles: β, γ, and δ, each with different velocities. Examples of these particles are labeled in Figure 2c, and their properties and interactions are summarized in Table 1. (We should note that we have used the same set of Greek letters to represent different types of particles in different rules.)

For most ICs, application of ϕ_1 quickly results in the appearance of these particles, which then go on to interact, assuming they have distinct velocities. A survey of their interactions indicates that the α particle dominates: it persists after collision with any of the SS particles. Interactions among the three SS particles do take place, resulting in either a single β or a pair of α's. Thus, none of the interactions are annihilative: particles are produced in all interactions. As a result, once a set of particles comes into existence in the space-time diagram, one can guarantee that at least one particle persists in the final configuration. For almost all values of ρ_0, ϕ_1's formation of persistent particles ultimately prevents it from attaining global synchrony. Only when ρ_0 is very close to 0.0 or 1.0 does ϕ_1 reach the correct final configurations. This accounts for its very low fitness.

Cellular Automata		Particles and Interactions			
Chromosome	Generation $(\mathcal{P}^{149}_{10^4}, \mathcal{P}^{599}_{10^4}, \mathcal{P}^{999}_{10^4})$	Label	Domain Boundary	Temporal Periodicity	Velocity
$\phi_1 =$	1	α	$S\overline{S}$	2	-1/2
F8A19CE6		β	SS	4	-1/4
B65848EA	(0.00,	γ	SS	8	-1/8
D26CB24A	0.00,	δ	SS	2	0
EB51C4A0	0.00)	$\beta + \alpha \rightarrow \alpha$, $\gamma + \alpha \rightarrow \alpha$, $\delta + \alpha \rightarrow \alpha$			
$\phi_2 =$	5	α	$S\overline{S}$	2	-1/2
F8A1AE2F		β	$S\overline{S}$	6	0
CF6BC1E2 D26CB24C 3C266E20	(0.33, 0.07, 0.03)	$\beta + \alpha \rightarrow \emptyset$			
$\phi_3 =$	13	α	$S\overline{S}$	4	-3 /4
F8A1AE2F		β	$S\overline{S}$	6	0
CE6BC1E2	(0.57,	γ	$S\overline{S}$	12	1/ 4
C26CB24E	0.33,	δ	$S\overline{S}$	2	1/2
3C226CA0	0.27)	$\beta + \alpha \rightarrow \emptyset$, $\gamma + \alpha \rightarrow \emptyset$, $\delta + \alpha \rightarrow \emptyset$			
$\phi_{sync} =$	100	α	$S\overline{S}$	-	0
FEB1C6EA		β	DS	2	1
B8E0C4DA	(1.00,	γ	SD	2	-1
6484A5AA	1.00,	δ	DS	4	-3
F410C8A0	1.00)	μ	SD	2	3
		ν	$D\overline{D}$	2	-1
	Decay:$\alpha \rightarrow \gamma + \beta$ Annihilative: $\gamma + \delta \rightarrow \emptyset$, $\mu + \beta \rightarrow \emptyset$				
	Reactive: $\beta + \gamma \rightarrow \nu$ (d mod $4 = 1$), $\nu + \delta \rightarrow \beta$, $\mu + \nu \rightarrow \gamma$				
	Reversible: $\beta + \gamma \rightarrow \delta + \mu$ (d mod $4 \neq 1$), $\mu + \delta \rightarrow \gamma + \beta$				

Table 1: ϕ_{sync} and its ancestors: Particles and their dynamics for the best CAs in early generations of the run that found ϕ_{sync}. The table shows only the common particles and common two-particle interactions that play a significant role in determining fitness. $\emptyset$ indicates a domain with no particles. Each CA ϕ is given as a hexadecimal string which, when translated to a binary string, gives the output bits of ϕ in lexicographic order ($\eta = 0^7$ on the left).

Epoch 2: Suppression of In-Phase Defects. Following the discovery of ϕ_1, the next sharp increase in fitness is observed in generation 5, when the best CA in the population, ϕ_2, has $F_{100} \approx 0.54$. The rise in fitness can be attributed to ϕ_2's ability to suppress in-phase (SS) defects for ICs with very low or very high ρ_0.

The space-time behavior of ϕ_2 is dominated by two new and different $S\overline{S}$ particles, labeled α and β (see Table 1; examples are labeled in Figure 2d). In addition to the suppression of SS boundaries, α and β annihilate each other; even on some ICs with intermediate ρ_0, ϕ_2 is able to reach synchronous configurations due to these annihilations. However, since the velocity difference between α and β is only 1/2, the two particles might fail to annihilate each other before the maximum of M time steps have elapsed.

In spite of these improvements, ϕ_2 still fails on a large fraction of its fitness tests. Often the same type of particle occurs more than once in the configuration. Since they travel at the same velocity, these identical particles cannot interact, so they persist in the absence of particles of a different type. Global synchrony is achieved (possibly in more than M time steps) only when the number of α particles and β particles in any configuration are equal. Our studies of ϕ_2 show that the probability of occurrence of β is about twice that of α, so their numbers are often unequal.

From the standpoint of the genetic operators acting on the CA rules, a small change in the relevant entries in ϕ is sufficient to significantly modify the properties of the domain boundaries. As a result, it is the mutation operator that seems to play the primary role in this and subsequent epochs in discovering high-performance CAs.

Epoch 3: Refinement of Particle Velocities. A much improved CA, ϕ_3, is found in generation 13. Its typical behavior is illustrated in Figure 2e. ϕ_3 differs from ϕ_2 in two respects, both of which result in improved performance. First, as noted in Table 1, the velocity difference between α and γ, the two most commonly occurring particles produced by ϕ_3, is larger (1 as compared to 1/2 in ϕ_2), so their annihilative interaction typically occurs more quickly. This means ϕ_3 has a better chance of reaching a synchronized state within M time steps. Second, the probabilities of occurrence of α and γ are almost equal, meaning that there is a greater likelihood they will pairwise annihilate, leaving only a single synchronized domain.

In spite of these improvements, it is easy to determine that ϕ_3's strategy will ultimately fail to synchronize on a significant fraction of ICs. As long as $S\overline{S}$ particles exist in the space-time diagram, there is a non-zero probability that a pair of $S\overline{S}$ defect sites would be occupied by a pair of identical particles moving in parallel. In the absence of other particles in the lattice such a particle pair could exist indefinitely, preventing global synchrony. Thus a completely new strategy is required to overcome persistent parallel-traveling particles.

Epoch 4: The Final Innovation. In the 20th generation a final dramatic increase in fitness is observed when ϕ_4 is discovered. ϕ_4 has $F_{100} \approx 0.99$ and displays quite different space-time behavior (Figure 2f). Following the discovery of ϕ_4 and until the end of the run in generation 100, the best CAs in each generation have $F_{100} = 1.00$. Also, no significant variation in the space-time behavior is noticeable among the best CAs in this run. In particular, ϕ_4's strategy is very similar to that of ϕ_{sync}, a perfected version of ϕ_4 that appeared in the last generation. Here we will make our earlier intuitive description of ϕ_{sync}'s strategy more rigorous.

As can be seen in Figure 1a, after the first few time steps the space-time behavior of ϕ_{sync} is dominated by two distinct types of domains and their associated particles. While one of the domains is the familiar S, the other domain—denoted D in Table 1— consists of temporally alternating and spatially shifted repetitions of 1000 and 1110. The result is a pattern with temporal and spatial period 4. In terms of the domain's regular language, though, D has temporal period 2: $(1000)^* = \mathbf{\Phi}((1110)^*)$ and $(1110)^* = \mathbf{\Phi}((1000)^*)$.

Using a transducer that recognizes the S and D regular languages, Figure 1a can be filtered to display the propagation of the particles embedded in the space-time behavior of ϕ_{sync} (Figure 1b). As pointed out earlier, such filtered space-time diagrams allow us to readily analyze the complex dynamics of ϕ_{sync}'s particles and their interactions. As shown in Table 1, ϕ_{sync} supports five stable particles, and one unstable "particle", α, which occurs at $S\overline{S}$ boundaries. α "lives" for only one time step, after which it decays into two other particles, γ and β, respectively occurring at SD and $D\overline{S}$ boundaries. β moves to the right with velocity 1, while γ moves to the left at the same speed.

The following simple scenario illustrates the role of the unstable particle α in ϕ_{sync}'s synchronization strategy. Let ϕ_{sync} start from a simple IC consisting of a pair of $S\overline{S}$ domain boundaries which are a small distance from one another. Each $S\overline{S}$ domain boundary forms the particle α, which exists for only one time step and then decays into a β-γ pair, with β and γ traveling at equal and opposite velocities. In this example, two such pairs are formed, and the first interaction is between the two interior particles: the β from the left pair and the γ from the right pair. As a result of this interaction, the two interior particles are replaced by δ and μ, which have velocities of -3 and 3, respectively. Due to their greater speed, the new interior particles can intercept the remaining β and γ particles. Since the pair of interactions $\gamma + \delta \rightarrow \emptyset$ and $\mu + \beta \rightarrow \emptyset$ are annihilative, and because the resulting domain is S, the configuration is now globally synchronized[4]. The basic innovation of ϕ_{sync} over ϕ_3 is the formation of the D domain, which allows two globally out-of-phase S domains to compete according to their relative size and so allows for the resolution of global phase frustration. D achieves this by replacing S domains with itself—a nonsynchronizable region.

The particle interactions in the filtered space-time diagram in Figure 1b (starting from a random IC) are somewhat more complicated than in this simple example, but it is still possible to identify essentially the same set of particle interactions ($\beta + \gamma \rightarrow \delta + \mu$, $\beta + \gamma \rightarrow \nu$, and $\gamma + \delta \rightarrow \emptyset$) that effect the global synchronization in the CA.

7. Concluding Remarks

In summary, the GA found embedded-particle CA solutions to the synchronization task. Although such perfectly performing CAs were distinct in detail and produced different domains and particles, they all used similar strategies for performing the task. It is impressive that the GA was able to discover complex orchestrations of particle interactions resulting in 100% correct solutions such as that described for ϕ_{sync}. The computational mechanics framework allowed us to "deconstruct" the GA's solutions and understand them in terms of particle interactions. In general, particle-level descriptions amount to a rigorous language for describing computation in spatially extended systems.

Several issues, important for putting the preceding results in a more general context, should be mentioned in closing. First, implicit in the definition of a CA is a globally synchronous update clock. That is, a CA's local states are updated at the same time across the lattice. (And this is a fundamental architectural difference with many of the natural processes mentioned in the introduction.) But since each site has a processor ϕ which determines local behavior and site-to-site interactions, the effect of the underlying global update need not be manifest directly in globally synchronous configurations[5]. In this light, our choice of the synchronization task means that we have considered one particular aspect of how this dynamical behavior might emerge: i.e., can local information processing and communication be designed by a GA to take advantage of the globally synchronous update signal?

Second, this observation suggests an alternative and potentially more important study to undertake: the evolution of a decentralized, distributed system whose components are fully

[4]One necessary refinement to this explanation comes from noticing that the β-γ interaction depends on the inter-particle distance d, where $0 \leq d \leq 2r$. If $d \bmod 4 \neq 1$, then we have the interaction $\beta + \gamma \rightarrow \delta + \mu$. But if $d \bmod 4 = 1$, then we have $\beta + \gamma \rightarrow \nu$. The particle ν is essentially a defect in the D domain.

[5]Indeed, one of the earliest mathematical articulations of a similar synchronization problem in a distributed system—the firing-squad synchronization problem (FSSP)—uses a globally synchronous update clock. In spite of the global update mechanism, it is the site-to-site interactions among the individual processors in the FSSP that makes the problem interesting and difficult. Although FSSP was first proposed by Myhill in 1957, it is still being actively studied [14].

asynchronous. We hope to return to this more difficult GA study in the future.

Third and finally, biological evolution is a vastly more complex process than the restricted framework we've adopted here. Its very complexity argues for new methods of simplifying its analysis—methods that are sensitive to the interaction between the nonlinear dynamics of individual behavior, on the one hand, and population dynamics guided by natural selection, on the other. Our goal is to delineate the evolutionary mechanisms that drive the emergence of useful structure. Given this, we believe that detailed analysis of simplified models, such as the one presented above, is a prerequisite to understanding the emergence and diversity of life.

Acknowledgments

We thank Dan Upper for suggesting the synchronization task and for helpful discussions. This research was supported by the Santa Fe Institute, under the Adaptive Computation and External Faculty Programs and under NSF grant IRI-9320200 and DOE grant DE-FG03-94ER25231. It was supported by the University of California, Berkeley, under the ONR Dynamical Neural Systems Program and AFOSR grant 91-0293.

References

[1] J. P. Crutchfield and J. E. Hanson. Turbulent pattern bases for cellular automata. *Physica D*, 69:279–301, 1993.

[2] J. P. Crutchfield and M. Mitchell. The evolution of emergent computation. Technical Report 94-03-012, Santa Fe Institute, Santa Fe, New Mexico, 1994.

[3] R. Das, M. Mitchell, and J. P. Crutchfield. A genetic algorithm discovers particle-based computation in cellular automata. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature—PPSN III*, volume 866, pages 344–353, Berlin, 1994. Springer-Verlag (Lecture Notes in Computer Science).

[4] P. Devreotes. *Dictyostelium discoideum*: A model system for cell-cell interactions in development. *Science*, 245:1054, 1989.

[5] H. A. Gutowitz, editor. *Cellular Automata*. MIT Press, Cambridge, MA, 1990.

[6] J. E. Hanson and J. P. Crutchfield. The attractor-basin portrait of a cellular automaton. *Journal of Statistical Physics*, 66:1415, 1992.

[7] J. E. Hanson. *The Computational Mechanics of Cellular Automata*. PhD thesis, University of California, Berkeley, 1993. published by University Microfilms Intl., Ann Arbor, MI.

[8] G. Laurent and H. Davidowitz. Encoding of olfactory information with oscillating neural assemblies. *Science*, 265:1872 – 1875, 1994.

[9] M. Mitchell, J. P. Crutchfield, and P. T. Hraber. Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D*, 75:361 – 391, 1994.

[10] M. Mitchell, P. T. Hraber, and J. P. Crutchfield. Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 7:89–130, 1993.

[11] D. W. Thompson. *On Growth and Form.* Cambridge University Press, Cambridge, 1917.

[12] T. Toffoli and N. Margolus. *Cellular Automata Machines: A new environment for modeling.* MIT Press, Cambridge, MA, 1987.

[13] S. Wolfram, editor. *Theory and applications of cellular automata.* World Scientific, Singapore, 1986.

[14] J. B. Yunes. Seven-state solutions to the firing squad synchronization problem. *Theoretical Computer Science*, 127:313 – 332, 1994.

* References [1,6,7] and [2,3,10] are available over the internet at the world wide web sites http://www.santafe.edu/projects/CompMech/ and http://www.santafe.edu:/projects/evca/ respectively.