Evolving Asynchronous Cellular Automata for Density Classification

Francis Jeanson

University of Sussex, Brighton, UK
f.jeanson@sussex.ac.uk

Abstract

This paper presents the comparative results of applying the same genetic algorithm (GA) for the evolution of both synchronous and randomly updated asynchronous cellular automata (CA) for the computationally emergent task of density classification. The present results indicate not only that these asynchronous CA evolve more quickly and consistently than their synchronous counterparts, but also that the best performing asynchronous CA find equally good solutions on average to the density classification task in fewer computational steps than synchronous CA.

Introduction

For the past 50 years cellular automata (CA) have established themselves as popular platforms to investigate complex phenomena. Their attractiveness stems in part from their ability to expose highly complex or even chaotic behaviour from an initially simple spatial configuration and set of update rules. An important insight that CA may provide is that in contrast to real world systems their dynamical laws are not bound by the classical laws of physics. Instead the laws that dictate the behaviour of a system are fully defined in terms of a state update policy which we may call Φ. Φ is defined by: the neighbourhood r of cells whose states causally impact the state of other cells, the rules that dictate *how* these neighbouring cell states impact other cells, and the global selection policy which specifies the set of cells that are to be updated by those rules. Traditionally work in this field has mostly been preoccupied in finding new sets of rules that give rise to interesting emergent phenomena. Indeed the general behaviour of a CA will exhibit a large diversity of dynamics with respect to rule updates. However, it is important not to omit the role of the neighbourhood and selection policy. The goal of this paper is to focus particularly on the latter by exposing the impact of the selection policy on evolved rules for cellular automata.

The most popular selection policy employed in CA research is synchrony. Here, all cells are selected and updated to their next state at each time step. Synchrony in cellular update implies that an automaton may potentially exploit the entire cell space to perform interesting global computations from local cellular interactions. This is possible because of an advantageous set of update rules found by the genetic algorithm (GA). A good set of rules that does in fact exploit this space efficiently at every time step is rare and ultimately difficult to find even via genetic search. In contrast an independent random updating selection policy where a single cell is updated at each time step does not allow an automaton to exploit space, since only a small region defined by the neighbourhood r of a single cell can be looked-up at any given time. This however may favour a genetic selection of rules that allow a CA to exploit time over space.

Asynchronous Computation

In contrast to synchronous cell state update where all cells of the automaton are updated in unison, asynchronous cellular automata (ACA) employ a selection policy whereby a single or a subset of cells are updated at a single time step. Independent random updating where a single cell is picked with uniform probability and updated over a single step is standard, although a number of alternatives have been explored (Schonfisch and de Roos 1999)[1]. In the past few decades ACA have been more carefully considered. Asynchronous update has been argued to be a more realistic approach in models of biologically inspired complex systems (Dellaert and Beer 1994; Harvey and Bossomaier 1997; Lee et al. 2007). Dellaert and Beer initially hypothesized that one of the main drawbacks from asynchrony comes from the difficulty due to indeterminacy in analyzing their behaviour. Furthermore this indeterminacy also seems to suggest that no general state attractor can be reached by asynchronous cellular automata. Harvey and Bossomaier challenge these worries and show that for random boolean networks (RBNs) asynchronous update may lead to a point attractor with a probability of $1/2^N$; where N is the number of nodes in the network (Harvey and Bossomaier 1997). They also show that loose attractors may be reached for the same type of update mechanism. Furthermore they intro-

[1]They distinguish step-driven from time-driven asynchronous updating. For instance a number of cells could be picked given a certain probability, in a particular sequence, or at a particular time.

duce practical methods for the analysis of indeterministic updating via probabilistic reasoning by careful inspection of node connectivity in RBNs. Interestingly experimental results by Harvey and Bossomaier indicate that the random node update with replacement method arrived more quickly at a point attractor in state space than did random update without replacement and synchronous update in RBNs. Lee, Adachi and Peper (2007) also propose asynchronous updating in two-dimensional cellular automata as a reliable, and biologically sound mechanism for self-replication. To them is appears that natural systems must have acquired tolerance to indeterministic interactions at the cellular level and that novel strategies have developed allowing the system to profit from this indeterminism. Kanada (1997) also emphasizes the significance of modeling ACA for real world applications and biological simulation. According to him synchronous emergent computation causes what he calls 'phantoms' which can be characterized as fragile system states. Slight disturbances from environmental noise will however prevent such systems from existing. His work on one-dimensional ACA shows that random or noisy interactions that naturally occur in these systems may play a positive role for our modeling and understanding of real world systems behaviour.

The current topic of research introduces a novel application of ACA by applying them to a computationally emergent task: density classification. Although, a priori, asynchronous automata have been considered to bear significance principally for the modeling of biologically inspired systems, the originality of their behaviour and ability to handle noise should give rise to interesting behaviour in a purely computational task. In fact, exploring the potential of asynchronous cellular updating in the well defined density classification task should eliminate any conceptual obscurities and inspire the potentially widespread significance of asynchrony in locally coordinated phenomena. For instance, we may think of the implications for the understanding of termite colony stigmergy (Grassé 1959, Beckers et al. 1994), or even for understanding synergistic effects of neuronal activity in large neural groups (Edelman 1993, Kelso 1995).

Synchronous Density Classification

Mitchell et. al (Mitchell, Hraber and Crutchfield 1993; Mitchell, Crutchfield and Das 1996) have thoroughly investigated the potential of genetic evolution for computational emergence in the density classification task. This task requires that a given one-dimensional cellular automaton determines the initial density that is most present in the initial cellular state within a number of update steps. By the end of these update iterations all the cells of the CA should be in the state identical to the state originally dominating the density of the CA. For instance given a 10 cell, two-state (0 or 1) CA, the 10 cells after K update steps should all be set to 1 if the original density of the CA contained more 1's than 0's - or they should all be set to 0 otherwise. Mitchell describes this problem as a task that the CA needs to accomplish by making use of local information for global coordination. In dynamical terms, a set of update rules of a CA's policy Φ must be found so that given any initial cellular state distribution the CA will follow a path in state space to a point attractor for that initial configuration. Mitchell et al.'s experiments have all been conducted using synchronous cellular update. By using a genetic algorithm to evolve the update rules of synchronous cellular automata (SCA) they have been able to obtain a diverse number of rules that are amongst the best known rules to date for the density classification task given any initial configuration density. Mitchell et al.'s ϕ_d rule has about 95% success rate in comparison the best known rule: the GKL rule with 97.8% (Gaks, Kurdyumov and Levin 1978). A few years later Land and Belew (1995) noted that they obtained from genetic evolution rules performing as well as GKL. In the same paper however they prove that no two-state CA can perform the density classification task perfectly.

In the following section I present results obtained by partially replicating the evolutionary mechanism employed by Mitchell et al. Because the purpose of the experiment was not aimed at discovering better rules I conducted 30 runs on each experiment instead of the 100 that Mitchell et al. examined in order to save computational time for the evolution of asynchronous rules. Exactly the same genetic algorithm was used to evolve rules in both the synchronous and the asynchronous scenario. In contrast to the synchronous scenario, asynchronous updating is performed by independent random selection at a single time step of a single cell which is then updated according to the rule table for that CA.

Evolving Update Rules

In their initial experiments Mitchell, Hraber and Crutchfield (1993) evolve one-dimensional cellular automata with lattice size $N = 149$. This ensures that the initial configuration always contains either a majority of 0's or a majority of 1's. The chromosomes evolved were binary strings encoding the rule outputs for a given CA. These update rules consider a neighbourhood radius $r = 3$ as can be seen in figure 1; hence the total number of cells lookup for the update of a particular cell is $2r + 1 = 7$: the three cells to the left, the three cells to the right and the current cell itself[2]. Hence given that each cell has either binary state 0 or 1, the total number of possible update rules is $2^7 = 128$. Thus a chromosome is represented as a binary strings of length 128. This implies that the search space in which the genetic algorithm must find good solutions to the density classification task is of size 2^{128} - too large for any brute force search. A single run of their GA consisted of evolving a set of 100 update rules over 100 generations. At each generation 100 new

[2]Rule lookup wrap arround the CA for beyond boundary conditions.

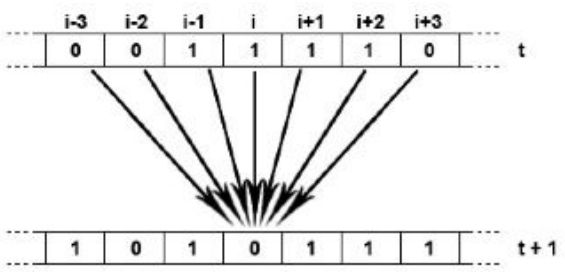

Figure 1: Illustration of cell state transition from step t to step $t + 1$ according to an arbitrary update rule. With r = 3, three neighbours to the left and to the right of the current cell i are looked-up to determine cell i's subsequent state.

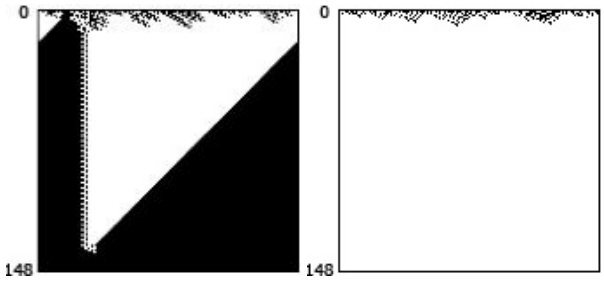

Figure 2: Sample iteration of a successful 1D-SCA in density classification. On the left the rule solving for a greater density of 0s; and the right the same rule for a greater density of 1s.

initial configurations with a biased distribution were created. This biased distribution ensures that there is a uniform distribution over the initial 100 densities. According to them the biased distribution allowed for the GA to find increasingly better solutions to the problem much more easily than if all the initial configurations had a probability distribution of 1/2, which rendered the task almost intractable. Each of the 100 rules are tested on each of these initial configurations over a fixed number $K = 149$ update steps. After the 100 rules have been tested the top 20 rules were selected from the population and copied to the next generation. Out of these 20 elite rules two were picked with replacement at random. Single point crossover was performed between these two rules after which mutation at exactly two loci was applied for each new offspring. In order to pick the best rules for the subsequent generation each rule was given a score corresponding to the number of correct density classifications it accomplished over the 100 initial configurations. The GA in the present experiment on synchronous updating was implemented exactly as described above. Mitchell et al. however collected their results after 300 runs. Because my aim was to compare the dynamics of synchronous updating with respect to asynchronous updating I opted for a more economical set of 30 runs per cellular configuration, in a total of 6 configurations. Initial test runs showed however that rule populations would often get stuck in very low minima. Yet because each configuration was meant to be evaluated over 30 runs only, I decided to add more noise to the mutation rate by allowing for a random mutation to occur at any loci with double probability, i.e. 1/64 for each 128 rule outputs.

The GA used for finding rules in asynchronous updating is identical to the algorithm for synchronous updating. As opposed to the SCA where all cells are updated according to the state of their neighbouring cells at the same time, the ACA implemented here selects a single cell with uniform random probability with replacement. Hence a cell has a probability of $1/N$ of being selected for update at every step while the remaining cells stay unchanged.

Finding Good Solutions Reliably

After running this evolutionary algorithm on 30 runs for SCA, I obtained by the final generations a majority of rules that would correctly classify the density of a given initial configuration about 50% of the time. These results are similar to those obtained by Mitchell et al. This would suggest that most runs found rules that could classify for almost any initial density whether the initial configuration contained more 1s than 0s or vice versa, but not both. All runs typically start with a set of poor performing rules. Out of the 30 runs, 1 run failed to evolved any rule that would correctly classify over 5% of the time; yet 7 runs succeeded in finding rules that classified with a success rate of 97% or higher. Although at first glance this seems to suggest that rules better than GKL were found, it is important not to forget that these rules where only tested on a set of 100 biased initial configurations. Further testing of these rules on larger set of initial configurations would provide a more accurate idea of a rule's actual performance. This is beyond the scope of the present paper however. Figure 2 illustrates the cellular progression of one of the best rules in synchronous updating. From this figure we notice that the evolved rules find a solution quickly for a greater initial density of 1s but takes much longer when the initial configuration contains more 0s. This was true for all the highest performing rules evolved for this SCA, and suggests that although a large number of SCA rules are capable of correctly classifying higher densities of 1s or 0s, a special strategy must evolve to successfully classify the opposite state density - we may call this 'density preference'. It is interesting to note that these rules highly resemble a specific type of rule found by Mitchell et al. which they call ϕ_a. This rule performs what they call block expansion to solve the task (Mitchell 1998).

Hence it isn't trivial for the GA to find good solutions to the density classification task in SCA. It seems that a satisfactory point attractor is only reached for half of the initial conditions on a large majority of the runs. In contrast however asynchronous updating gave interesting results in other dynamical systems. Harvey and Bossomaier for instance,

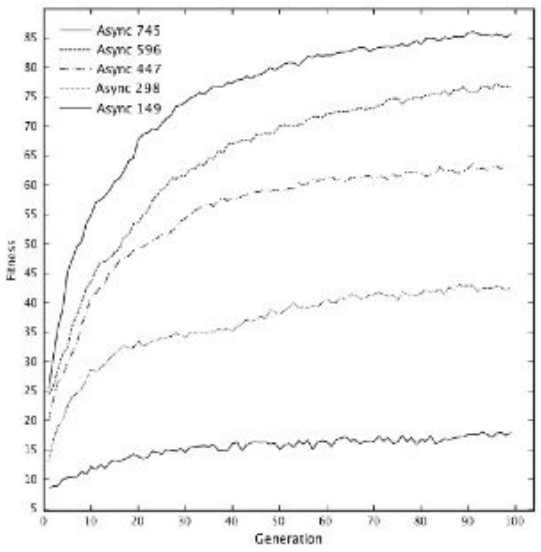

Figure 3: Average fitness of ACA for $K = 149, K = 298, K = 447, K = 596$, and $K = 745$ over 100 generations in 30 runs.

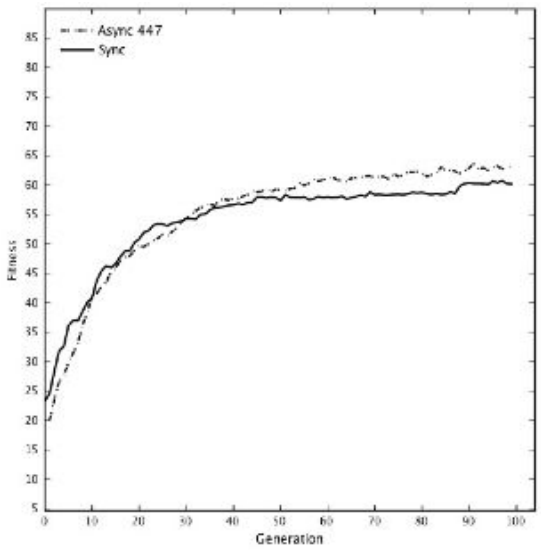

Figure 4: Average fitness of SCA, in comparison to ACA with $K = 447$ over 100 generations in 30 runs.

noticed that asynchronous RBNs tend to arrive a point attractors more quickly than synchronous updating. It should be interesting then to investigate this phenomenon in ACA for the density classification task.

Because ACA require that only a single cell be updated at every step, they require N times less computation over a given number of steps K. It was predicted then that a greater number of update steps would be required to obtain well performing rules in the density classification task with asynchrony. For this reason I conducted 30 runs on 5 different ACA configurations. The first configuration held the K number of update steps to 149. I'll refer to this scenario as Async 149. For the second configuration I decided to double the number of K steps in an ACA's computation: Async 298, and run the GA 30 times. Following the same procedure I ran the evolutionary algorithm on Async 447, Async 596, and Async 745, each extensions of Async 149 by factors of 3, 4, and 5 respectively.

As expected Async 149 did not find any high performing rules for the density classification task. The GA did manage to find a number of rules for Async 149 that correctly classified the CA up to about 22% of the time. These more easily solvable cases stem from conditions where initial configurations had highly biased densities. As factors in the number of update steps K increased, one notices an almost linear increase in performance at first. Evolving Async 298 gave rise to some rules reaching a success rate nearing 45%. Async 447 provided consistent rules surpassing the 50% mark, with a number of rules reaching rates of success of 68%, with an average of about 63%. However this progressive linear increase halted with $K = 596$. Evolving Async 596 indeed rarely gave rules with results any better than 80%, although 75% was consistently reached by at least half the rules after 100 generations. The increase to $K = 745$ confirmed this sudden decrease in rate of improvement, with top rules averaging the 86% mark. This diminishing rate of improvement suggests that there exists a non linear increase in the complexity for finding good rules that solve initial configurations with highly even state densities. Figure 3 illustrates the progression in fitness for each type of scenario over 100 generations.

In comparison the average success of rules discovered for SCA reaches roughly 60% (Figure 4) which happens to perform *worse* than Async 447 after 100 generations. However it is important not to forget that although the average success is relatively low, evolving SCA did give rise to the highly performing rules discussed above with rates nearing 100% success. This indicate then that SCA is prone to a much higher deviation than its ACA counterparts. We notice form Figure 5 that the standard deviation of evolved rules for asynchronous update increases as the number of steps K increases. This can be explained by the increasing specialization of a set of rules in the population. In other words, better performing rule sets have greater opportunity to 'prove themselves' when given more time to accomplish the task. We also notice from Figure 5. that the standard deviation of each ACA begins by increasing over the first generation quarter, but then all deviations progressively diminish.

The decrease in standard deviation in fitness means that rules that are more fit are found more consistently and provide increasingly similar performance. Hence evolving ACA gives rise to an increasingly reliable set of rules for the

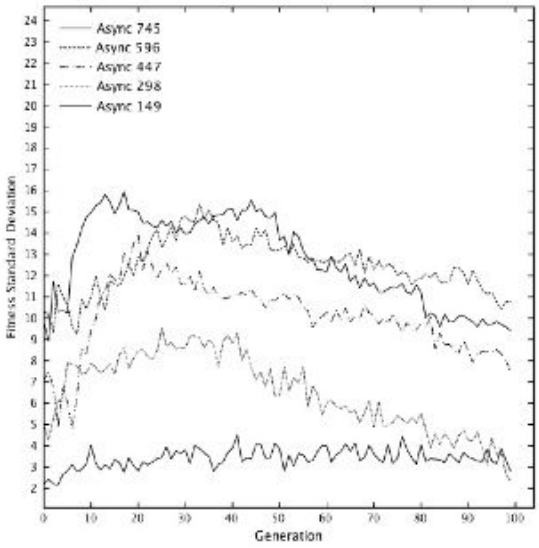

Figure 5: Standard deviation of rule fitness for Async 149, Async 298, Async 447, Async 596 and Async 745, over 100 generations.

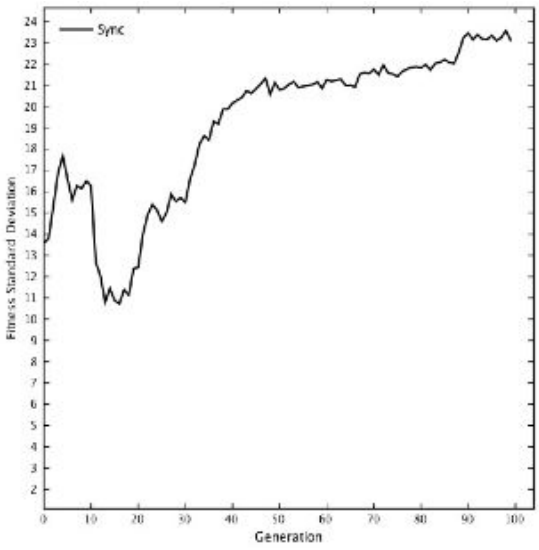

Figure 6: Standard deviation of rule fitness for SCA over 100 generations.

given task. A few hypotheses may begin to explain these results. Perhaps the GA evolves at every run a set of rules that are increasingly similar genetically as generations go by. This would inevitably cause a narrowing pool of rules that the GA always succeeds in finding. This hypothesis however doesn't fit the results which show that there is an initial increase of the standard deviation. A hypothesis that I consider more probable is that the GA manages to narrow down the pool of successful rules only after generating first a highly diverse pool of rules in which some perform very well and other quite poorly, after which a sort of population neutrality appears then to form by mutation and crossover which allows for this increasing and reliable specialization of the rule population after the first generation quarter.

In contrast the evolved SCA are increasingly more volatile as generations progress. Figure 6 not only shows a much higher deviation for SCA than for ACA but how this deviation increases after each generation. As mentionned earlier this high deviation is explained by the majority of rules which get stuck in local optima by achieving the correct classification of initial configurations for one of two states only, i.e. density preference. The few cases that find on the one hand high performing rules or on the other hand very poor rules will inevitably cause this strong deviation. In comparison ACA do not seem to suffer from this density preference (Figure 7). From this data then it is clear that evolving ACA provides good rules much more reliably than does the evolution of SCA for this task.

Finding Good Solutions Rapidly

Synchronous update implies that all cells are updated in unison, however as seen in Figure 4 SCA rules that perform well at the density classification task are sparse. Yet for ACA well performing rules are much easier to find for the GA. Figure 4 even shows how Async 447 has rules performing better than SCA on average. However Async 447 performs single cell update at every step. Thus after 447 update steps it has performed exactly $1 * K = 447$ cell updates. In comparison SCA perform $N * K = 149^2$ cell updates. At first glance this appears to mean that Async 447 found on average better solutions than SCA with about 50 times less computation. This isn't quite exact however. I mentioned earlier that evolved SCA rules will on average solve half of the initial configurations within half a dozen steps due to density preference. Solving the other half of initial configurations though will typically take at least 100 update steps. By averaging for both possible initial configurations is it reasonable to assume that on average roughly 55 steps are required to solve the density classification task when $N = 149$ with the best evolved rules. This means that most rules for SCA require about $55 * 149$ cell updates which still represents 18 times more computation then that required by Async 447. Overall this suggests that ACA can in fact perform faster than SCA at this task given that less perfect target rules are

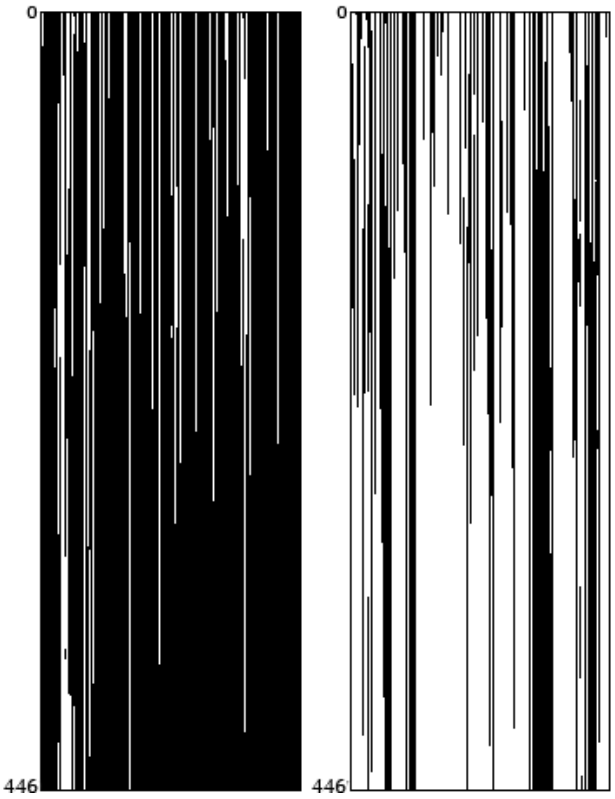

Figure 7: Sample iteration of a 1D-ACA in density classification. On the left the rule solving for a greater density of 0s; and the right the same rule for a greater density of 1s. Although perfect decisions are not made for these more difficult initial configurations, it is easy to see how the left ACA progressively eliminates cells in state 1. And vice versa for the ACA to the right.

required by a system or user exploiting these dynamics.

Conclusion

From the results collected here asynchronous cellular update in one-dimension automata may exhibit important computational qualities. Essentially these results first show that a genetic algorithm can find high performing update rules for ACA more reliably than for SCA. I've shown that rules for independent random updating of cell states to classify initial densities are found more consistently than when the updating is synchronous with similar success rates. This was particularly made evident by a different standard deviation between both update methods: Whereas SCA show consistent overall increasing deviation as rule generations go by, ACA show initial increase during the first quarter followed by significant decrease in standard deviation of fitness. Because the standard deviation of ACA isn't simply lower than that of SCA but actually decreases suggests that a quite radically different phenomenon is taking place when finding rules for asynchronous updating via genetic algorithms. Interestingly the evolved rules for ACA do not appear to suffer from density preference as do SCA. It is suspected that this phenomena is intimately tied to the reliability of ACA for finding good solutions. It appears then that as a tradeoff from exploiting space to find high performing rules, synchronous update renders the computation of a CA highly prone to instability. In contrast asynchronous update policies seem to allow for more robust activity by exploiting time, while providing sufficiently good performance.

Furthermore the results suggest that ACA can decide with much less computation the density of an initial configuration than SCA if this density is relatively biased. On average ACA performed about 18 times faster (computing time) than SCA with similar results. This concurs with the idea that although ACA may take more time (update steps) to arrive at a point attractor they require much less overall computation (computing time) than do SCA. This agrees with precedent results found by Harvey and Bossomaier (1997) on dynamics of asynchronous random boolean networks. Further work should be conducted to examine more precisely the threshold in update steps at which ACA find high performing rules for density classification. Also, a better understanding of how ACA exploit the state space should be developed.

The choice of density classification as a nontrivial task for the global arrangement of cell states from local interactions is proposed here as a simple yet well defined problem for exploring the potential of asynchronous updating in complex dynamics. Because CA behaviours are fundamentally dictated by their update policy Φ, it is reasonable and perhaps useful to regard Φ as the underlying 'physics' of these systems. The spatially distributed nature of cells and their update over time motivates the use of CA for real world models of global dynamics from local interactions. Hence the results obtained herein could potentially contribute to the better understanding of complex dynamics in natural phenomena. Arguably, dynamical properties of asynchronous cell selection may give insight into temporally dissociated interactions such as in chemical reactions, neural group activity, population dynamics etc. The two observed advantages of asynchronous random cell updating in the present experiments (reliability and rapidity) have quite distinct implications. Although both aspects may be practically exploited for engineering prospects, the reliability characteristic of asynchrony in the context of natural phenomena relates purely to the 'availability' of the underlying physics (update rules) which give rise to the behaviour of interest. Here, results imply that under conditions of asynchronous random cell selection these rules are more readily available for density classification from genetic search. Although specific to this task, such flexibility could speculatively be shared by other natural phenomena as mentioned above. The second aspect - which shows that density classification is obtained

more rapidly in ACA than SCA on average - may predict, however, that convergence towards stable attractors in natural temporally dissociated phenomena is likely to occur with higher frequency. This, of course, is contingent upon the fact that the dynamics of the present task can be extrapolated to other real world phenomena.

References

Beckers, R., Holland, O.E., Deneubourg, J.L, (1994). From local actions to global tasks: stigmergy and collective robotics. In P. Maes and R. Brooks, editors, *Artificial Life IV*, pages 181-189. MIT Press, Cambridge, MA.

Dellaert, F., Beer, R. (1994). Towards an evolvable model of development for autonomous agent synthesis. In P. Maes and R. Brooks, editors, *Artificial Life IV*, pages 246-257. MIT Press, Cambridge, MA.

Edelman, G. (1993). Neural darwinism: Selection and reentrant signaling in higher brain function. *Neuron*, 10:115-125.

Gaks, P., Kudryumov, G.L., Levin, L.A. (1978). One-dimentional uniform arrays that wash out finite islands. *Probl. Peredachi Inform.*, 14:92-98.

Grassé, P.P. (1959). La reconstruction du nid et les coordinations inter-individuelles chez *Bellicositermes natalensis* et *Cubitermes* sp. La theorie de la stigmergie: Essai d'interpretation des termites constructeurs. *Ins. Soc.*, 6:41-83.

Harvey, I., Bossomaier, T. (1997). Time out of joint: Attractors in asynchronous random boolean networks. *ECAL97*.

Kanada, Y. (1997). Asynchronous 1D cellular automata and the effects of fluctuation and randomness. Extended paper from the Artificial Life IV Poster.

Kelso, J.A.S. (1995). *Dymanic Patterns*, pages 37-43. MIT Press, Cambridge, MA.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*, pages 44-55. MIT Press, Cambridge, MA.

Mitchell, M., Hraber, P.T., Crutchfield, J.P. (1993). Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems* 7:89-130.

Mitchell M., Crutchfield, J.P., Das, R. (1996). Evolving cellular automata with genetic algorithms: a review of recent work. In Proceedings of the First International Conference on Evolutionary Computation and Its Applications *EvCA'96*. Moscow, Russia: Russian Academy of Sciences.

Land, M., Belew, R.K. (1995). No perfect two-state cellular automata for density classification exists. *Phys. Rev. Lett.* 74: 5148-5150.

Lee, J., Adachi, S., Peper, F. (2007). Reliable self-replicating machines in asynchronous cellular automata. *Artificial Life*, 13:397-413

Schonfisch, B., de Roos, A. (1999). Synchronous and asynchronous updating in cellular automata. *BioSystems*, 51:123-143.