

A Brief Note on Discrete Dynamical Learning Classifier Systems

Larry Bull

Learning Classifier Systems Group Technical Report – *UWELCSG08-002*

University of the West of England

Bristol BS16 1QY, U.K.

`larry.bull@uwe.ac.uk`

Jan 2008

Abstract

A number of representation schemes have been presented for use within Learning Classifier Systems, ranging from binary encodings to neural networks. This paper presents results from an initial investigation into using a discrete dynamical system representation within an accuracy-based Learning Classifier System. In particular, random Boolean networks are used to represent the traditional condition-action production system rules. It is shown possible to evolve an ensemble of such discrete dynamical systems to solve versions of the well-known Boolean multiplexer problem.

Introduction

The discrete dynamical systems known as Random Boolean networks (RBN) [Kauffman, 1969] were originally introduced to explore aspects of biological genetic regulatory networks and can be viewed as a generalization of binary Cellular Automata (CA) [von Neumann, 1966]. Since then they have been used as a tool in a wide range of areas such as self-organisation [e.g., Kauffman, 1993], computation [e.g., Fernández & Solé, 2004], robotics [e.g., Quick et al., 2003] and artificial creativity [e.g., Dorin, 2000]. Traditional RBN consist of N nodes, each connected to K other randomly chosen network members, with each performing a randomly assigned Boolean update function based on the current state of those K members in discrete time steps. Such updates are typically executed in synchrony across the network but asynchronous versions have also been presented (after [Harvey & Bossomaier, 1997]), leading to a classification of the space of possible forms of RBN [Gershenson, 2002]. Traditional RBN have well-studied temporal behaviour and analytical methods have been presented by which to determine the typical time taken to reach a basin of attraction and the number of states within such basins for a given degree of connectivity K [e.g., Kauffman, 1993].

To date, no temporally dynamic representation scheme has been used within Learning Classifier Systems (LCS) [Holland, 1976]. A number of representations have previously been presented beyond the traditional binary scheme however, including integers [Wilson, 2001a], real numbers [Wilson, 2000], Lisp S-expressions [Ahluwalia & Bull, 1999], fuzzy logic [Venzuela-Rendon, 1991] and neural networks [Bull, 2002]. The dynamical system approach to understanding cognition [e.g., Port & van Gelder, 1995] highlights the temporal aspect of behaviour. This paper presents an initial study into the use of a simple form of dynamical system within LCS, the RBN.

Random Boolean Networks

As noted above, within the traditional form of RBN presented by Kauffman, a network of N nodes, each with K connections to other nodes in the network, all update synchronously based upon the current state of those K nodes. Since they have a finite number of possible states and they are deterministic, such networks eventually fall into a basin of attraction. It is well-established that the value of K affects the emergent behaviour of RBN wherein attractors typically contain an increasing number of states with increasing K . Three phases of behaviour are suggested: ordered when $K=1$, with attractors consisting of one or a few states; chaotic when $K>3$, with a very large numbers of states per attractor; and, a critical regime around $K=2$, where similar states lie on trajectories that tend to neither diverge nor converge and 5-15% of nodes change state per attractor cycle (see [Kauffman, 1993] for discussions of this critical regime, e.g., with respect to perturbations). Figure 1 shows examples of ten randomly created $N=1000$ networks, each started from 10 random initial configurations, for varying K . The fraction of nodes which change state per update cycle is recorded and shown in the figures. Thus it can be seen that this number is typically low for low K but increases rapidly with $K>2$, as expected.

An RBN-LCS

In this paper we use a version of the simple accuracy-based LCS termed YCS [Bull, 2005] which is a derivative of Wilson's XCS [Wilson, 1995]. YCS is without internal memory and maintains a rulebase of P initially randomly created rules. Associated with each rule is a predicted payoff value (p), a scalar which indicates the error (ϵ) in the rule's predicted payoff and an estimate of the average size of the niches (action sets - see below) in which that rule participates (σ). The initial random population has these parameters initialized, somewhat arbitrarily, to 10.

On receipt of an input message, the rulebase is scanned, and any rule whose condition matches the message at each position is tagged as a member of the current match set [M]. An action is then chosen from those proposed by the members of the match set and all rules proposing the selected action form an action set [A]. A version of XCS's explore/exploit action selection scheme will be used here. That is, on one cycle an action is chosen at random and on the following the action with the highest average payoff is chosen deterministically.

The simplest case of immediate payoff reward R is considered here. Reinforcement in YCS consists of updating the error, the niche size estimate and then the payoff estimate of each member of the current [A] using the Widrow-Hoff delta rule with learning rate β :

$$\epsilon_j \leftarrow \epsilon_j + \beta(|R - p_j| - \epsilon_j) \quad (1)$$

$$\sigma_j \leftarrow \sigma_j + \beta(|[A]| - \sigma_j) \quad (2)$$

$$p_j \leftarrow p_j + \beta(R - p_j) \quad (3)$$

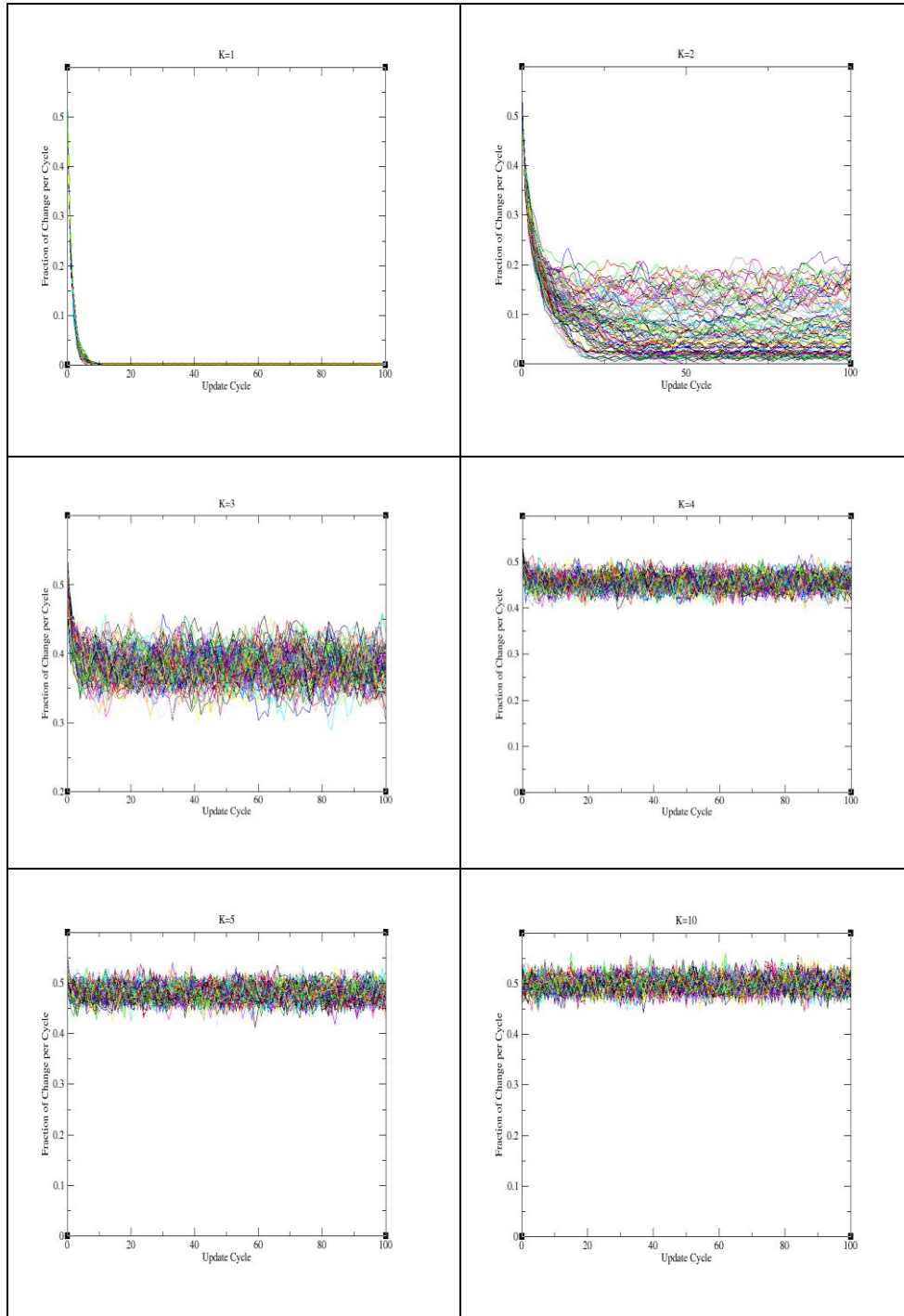


Figure 1: $N=1000$ RBN and various K , showing fraction of nodes which change state per step.

The original YCS employs two discovery mechanisms, a panmictic (standard global) Genetic Algorithm (GA) [Holland, 1975] and a covering operator. On each time-step there is a probability g of GA invocation. The GA uses roulette wheel selection to determine two parent rules based on the inverse of their error:

$$f_j = (1 / (\epsilon_j^\nu + 1)) \quad (4)$$

Here the exponent ν enables control of the fitness pressure within the system by facilitating tuneable fitness separation under fitness proportionate selection (see [Bull, 2005] for discussions). Offspring are produced via mutation (probability μ) and crossover (single point with probability χ), inheriting the parents' parameter values or their average if crossover is invoked. Replacement of existing members of the rulebase uses roulette wheel selection based on estimated niche size. If no rules match on a given time step, then a covering operator is used which creates a rule with the message as its condition and a random action, which then replaces an existing member of the rulebase in the usual way. Parameter updating and the GA are not used on exploit trials.

In this paper, to aid the generalization process, the panmictic GA is altered to operate within niches (see [Butz et al., 2004][Bull, 2005] for discussions). The mechanism uses XCS's time-based approach under which each rule maintains a time-stamp of the last system cycle upon which it was part of a GA. The GA is applied within the current [A] when the average number of system cycles since the last GA in the set is over a threshold θ_{GA} . If this condition is met, the GA time-stamp of each rule is set to the current system time, two parents are chosen according to their fitness using standard roulette-wheel selection, and their offspring are potentially crossed and mutated, before being inserted into the rulebase as described above.

To use RBN as the rules within this system the following scheme is adopted. Each initial randomly created rule has a randomly assigned degree of connectivity K , here $1 \leq K \leq 5$. Then for each of the N nodes which form a given rule, a random Boolean function is created for the 2^K possible input states. There are as many nodes as input fields for the given task and the first connection of each node is set to the corresponding node of the input. The other $K-1$ connections are assigned at random within the RBN as usual. In this way the current input state is always considered along with the current state of the RBN itself per network update cycle.

Matching consists of executing each RBN rule for T cycles based on the current input. Nodes are initialised to the state of the corresponding current input. In this initial study well-known Boolean problems are explored and hence there are only two possible actions. Thus if a fraction of nodes ϕ within the given rule have been logical '0' for the last ψ update cycles, the rule is said to match the current input and advocate action '0'. Conversely, if the fraction ϕ of nodes were in state '1' for the last ψ cycles, the rule is said to match and advocate action '1'. Otherwise the rule is deemed not to match the current input. Thereafter match set and action set processing proceeds as described above. A cover operator has not been found necessary in the tasks explored here.

Due to the different degrees of connectivity within the rules, the representation scheme is of variable length. Hence mutation is only used here and applied to the elements of the Boolean functions at rate μ ; the output states of the Boolean functions of each node are essentially concatenated together to form a binary string of length $N \cdot 2^K$. Similarly, at rate μ , for nodes with $K > 1$, the connectivity of one of the K internal connections for the given node is re-assigned at random. All other GA processing is as described above.

Experimentation

Two versions of the well-known multiplexer task are used in this paper. These Boolean functions are defined for binary strings of length $l = k + 2^k$ under which the k bits index into the remaining 2^k bits, returning the value of the indexed bit. A correct classification results in a payoff of 1000, otherwise 0.

Figure 2 (top) shows the performance of the RBN-LCS on the 11-bit multiplexer problem with most parameters taken from [Bull, 2005]: $P=2000$, $\mu=0.01$, $\nu=10$, $\theta_{GA}=25$, $\beta=0.2$, $T=100$, $\psi=2$ and $\phi=0.75$. After [Wilson, 1995], performance from exploit trials only is recorded (fraction of correct responses are shown), using a 50-point running average, averaged over ten runs. It can be seen that optimal performance is reached around 30,000 trials, with the average error dropping to roughly 10% of the payoff range. Figure 2 also shows the average connectivity K within each of the rules. As can be seen, this converges to around 1.75, i.e., connectivity evolves close to the aforementioned critical regime identified for RBN in general, and thus many rules contain nodes using a connection to one other node along with the input. Figure 2 (bottom) shows the performance of the system with $P=5000$ on the 20-bit multiplexer. Again, average error drops to around 10% of the payoff range and the average connectivity of rules is approximately 1.75.

With respect to the degree of generalization seen it is firstly important to note that it is obviously occurring. For example, in the 11-bit multiplexer there are 2^{11} possible inputs, i.e., 2048. A typical solution contains around 1500 non-unique, accurate rules out of the 2000, each participating in action sets of around 70 rules, with numerosities which vary between 2 and 20. Of course, it may be that with different parameters better generalization is possible and this remains open to future exploration; the generalization mechanism of such accuracy-based LCS is not dependent upon the representation scheme [Butz et al., 2004]. In the 20-bit multiplexer there are $\sim 1 \times 10^6$ possible inputs and the system is again clearly generalizing effectively with its fixed rule base of 5000 rules since, on average, each RBN must handle approximately 200 inputs to cover the whole space. The numerosities indicate that individuals evolve to handle more than this number however since around 4000 non-unique, accurate rules are typically found.

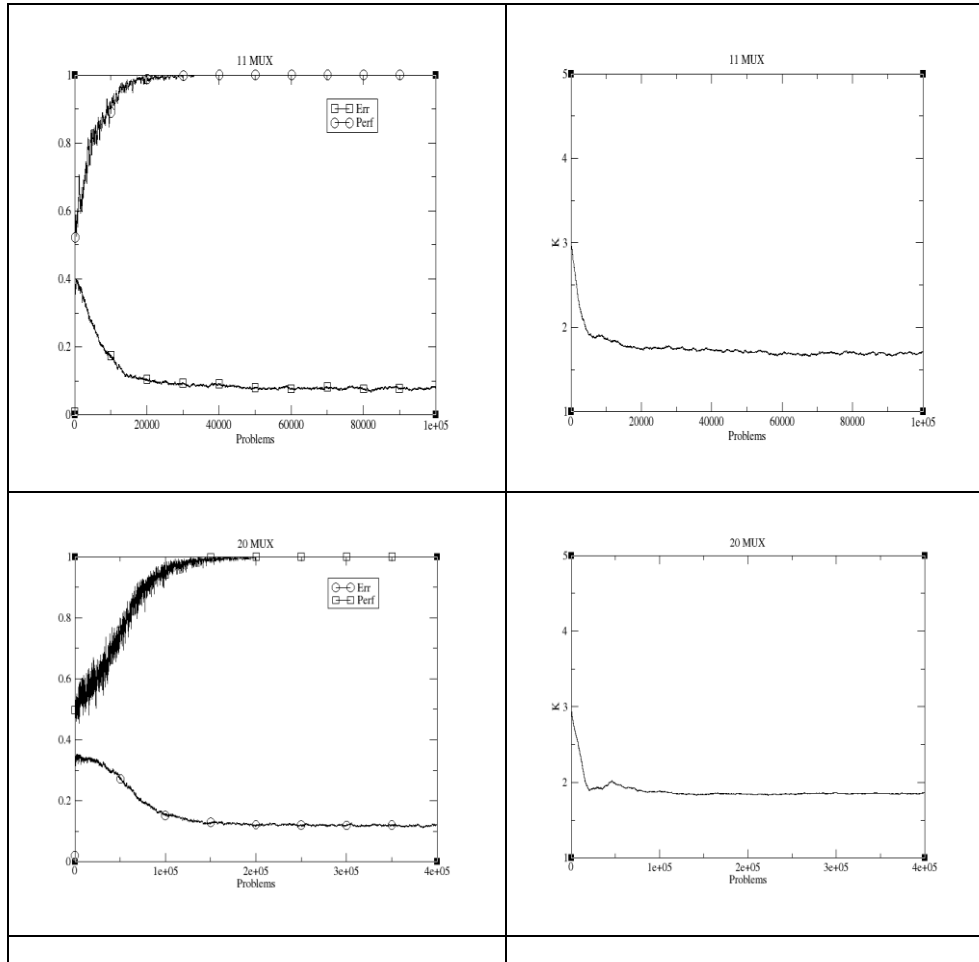


Figure 2: Performance of the RBN-LCS on versions of the Boolean multiplexer problem.

Conclusions

This paper has presented a new form of LCS based on dynamical systems (DS-LCS). In particular, a discrete DS-LCS has been explored based on random Boolean networks - the RBN-LCS. This is loosely related to Forrest and Miller's [1989] use of RBN to model the internal rule chaining of traditional LCS. It has here been shown that the evolutionary search of the GA is able to find a set of RBN that collectively solve a computational task under the reinforcement learning scheme of LCS. This is only initial work and current research is exploring many of the myriad of possibilities DS-LCS present. Most importantly, they are being applied to tasks where the evolution of their temporal behaviour can be exploited directly, such as time-series data mining and adaptive control. The recent extension to computed predictions

within LCS (following [Wilson, 2001b][O'Hara & Bull, 2004]) is also being considered.

References

- Ahluwalia, M. & Bull, L. (1999) A Genetic Programming Classifier System. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela & R.E. Smith (Eds) *Proceedings of the Genetic and Evolutionary Computation Conference – GECCO-99*. Morgan Kaufmann, pp11-18.
- Bull, L. (2002) On Using Constructivism in Neural Classifier Systems. In J. Merelo, P. Adamidis, H-G. Beyer, J-L. Fernandez-Villacanas & H-P. Schwefel (Eds) *Parallel Problem Solving from Nature - PPSN VII*. Springer Verlag, pp558-567.
- Bull, L. (2005) Two Simple Learning Classifier Systems. In L. Bull & T. Kovacs (eds) *Foundations of Learning Classifier Systems*. Springer, pp63-90.
- Butz, M., Kovacs, T., Lanzi, P.L. & Wilson, S.W. (2004) Toward a Theory of Generalization and Learning in XCS. *IEEE Transactions on Evolutionary Computation* 8(1).
- Dorin, A. (2000) Boolean Networks for the Generation of Rhythmic Structure. In *Proceedings of the Australian Computer Music Conference*, pp38-45.
- Fernández, P. & Solé, R. (2004) The Role of Computation in Complex Regulatory Networks. In E. Koonin, Y. Wolf & G. Karev (Eds.) *Power Laws, Scale-free Networks and Genome Biology*. Landes.
- Forrest, S. & Miller, J.H. (1989) The Dynamical Behaviour of Classifier Systems. In J.D. Schaffer (Ed) *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann, pp304-310.
- Gershenson, C. (2002) Classification of Random Boolean Networks. In R.K. Standish, M. Bedau & H. Abbass (Eds.) *Artificial Life VIII*. MIT Press, pp1-8.
- Harvey, I. & Bossomaier, T. (1997) Time out of Joint: Attractors in Asynchronous Random Boolean Networks. In *Proceedings of the Fourth European Artificial Life Conference*. MIT Press, pp67-75.
- Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press.
- Holland, J.H. (1976) Adaptation. In R. Rosen & F.M. Snell (Eds) *Progress in Theoretical Biology* 4. Plenum.
- Kauffman, S.A. (1969) Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets. *Journal of Theoretical Biology* 22:437-467.
- Kauffman, S. A. (1993) *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford.
- O'Hara, T. & Bull, L. (2004) Prediction Calculation in Accuracy-based Neural Learning Classifier Systems. Technical Report *UWELCSG04-004*. Available from <http://www.cems.uwe.ac.uk/lcsg>
- Port, R., & van Gelder, T. (1995) *Mind as Motion: Explorations in the dynamics of cognition*. MIT press.
- Quick, T., Nehaniv, C., Dautenhahn, K. & Roberts, G. (2003) Evolving Embedded Genetic Regulatory Network-Driven Control Systems. In *Proceedings of the Seventh European Artificial Life Conference*. Springer, pp266-277.

- Valenzuela-Rendon, M. (1991) The Fuzzy Classifier System: a Classifier System for Continuously Varying Variables. In L. Booker & R. Belew (Eds) *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann, pp346-353.
- Von Neumann, J. (1966) *The Theory of Self-Reproducing Automata*. University of Illinois.
- Wilson, S.W. (1995) Classifier Fitness Based on Accuracy. *Evolutionary Computing* 3: 149-175.
- Wilson, S.W. (2000) Get Real! XCS with Continuous-Valued Inputs. In P-L. Lanzi, W. Stolzmann & S.W. Wilson (Eds) *Learning Classifier Systems: From Foundations to Applications*. Springer, pp209-222.
- Wilson, S.W. (2001a) Mining Oblique Data with XCS. In Lanzi, P. L., Stolzmann, W., & Wilson, S. W. (Eds) *Advances in Learning Classifier Systems - IWLCS-2000*. Springer, pp158-176.
- Wilson, S.W. (2001b) Function Approximation with a Classifier System. In Spector, L., D., G. E., Wu, A., Langdon, W.B., Voight, H. M., & Gen, M., (Eds.) *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO 01*. Morgan Kaufmann, pp 974-981.