Understanding Evolved Strategies for System-Wide Coordination in Noisy Environments

Alexander Franks
Evanston Township High School

November 17, 2004

1 Introduction

Many natural systems exhibit globally organized behavior without the aid of centralized control. Moreira *et al.*[1] cite many examples of natural decentralized systems including

> conventions and norms, social learning in animals and humans, as well as fads, rumors and revolts. Examples are also abundant in biology and medicine: the saltation model of human growth speaks of decentralized cellular coordination, as does the *Dictyostelium's* transition to a multicellular developmental program.

To better understand decentralized systems, researchers are examining the strategies that nodes in a decentralized system must follow in order to achieve global coordination using only local information. In order to explore these strategies, it is first important to be aware of the system's topology and environmental conditions.

Recent important work by Watts and Strogatz has shown that the structure of many networks are somewhere between ordered and random [5]. These so called "small world" networks have a high degree of clustering as well as a short distance between any two nodes. Since there is increasing evidence that most large networks exhibit the "small world" phenomena, it is likely that many decentralized systems communicate via a small world network [4]. Furthermore, natural environments are "noisy". A strategy for global coordination must be able to function in "noisy" environments. This paper examines strategies under noisy conditions within complex networks.

The goal of this paper is to explore the evolution of efficient strategies in noisy and complex networks. To this end, I will use genetic algorithms. Genetic algorithms have become an important tool in science, especially because they may offer insight into how natural evolution may work. It is not just what evolves, but how and why they evolve, that is important.

Following prior work, I use the density classification task as a model for system-wide coordination in decentralized systems. I find that there is a class of efficient "majority-type" rules, whose exact makeup is dependent on the environmental noise. I identify an optimal function for reaching system-wide coordination and explore its meaning. My results suggest that evolution uses the solutions to simple problems as a template for solving more complex problems.

2 Methods

The Density Classification Task – The object of the density classification task is to get all units of a binary string to converge to the majority state. That is, if the initial string contains more 1's than 0's, the final state should be a binary string of all 1's. While this is a trivial task for centralized systems, it is very challenging for decentralized systems. Thus, it is a good model for system-wide coordination and global information processing [6]. Indeed, the system must extract global information from local interactions and converge to the majority state. The model used here is based on the simple computational mechanism of a cellular automaton (CA), in which each agent can be in one of two states, for example 1 or 0. Each time step, every agent updates its state according to a rule. In this CA model, each agent has access to limited information – it knows only k states: that of itself and its $k-1$ neighbors. A key property of cellular automata is that very simple rules can produce extremely complex behavior [2].

The rule, or updating function ϕ, "tells" the agent what state to take on in the next time step, based on the state of its neighbors. The rule is encoded in a binary string in which each possible input pattern has a corresponding output bit (Fig. 1). Each input pattern is itself a k bit binary string representing an agent's own state

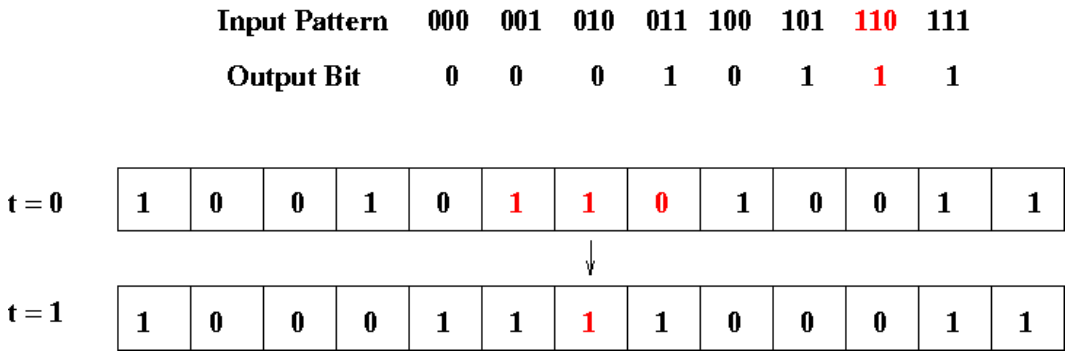

Figure 1: Illustration of a binary state cellular automaton with system size $N = 13$, $k = 3$, noise intensity $\eta = 0$, and rewiring probability $\rho = 0$. The correspondence of input pattern to output bit is the updating rule ϕ; here the majority rule is used. The input patterns are ordered lexicographically and the string of corresponding output bits, in this case 00010111, is the binary string encoding the rule. Each agent reads k states (that of itself and its $\frac{k-1}{2}$ neighbors on either side) and updates its state in the next time step according to the input pattern and the corresponding output bit, as the bits in red illustrate. Time step $t = 0$ is the initial condition. Since 7 of the 13 initial states are 1's (a majority), to be counted a success, the system must reach a consensus of all 1's by $t = 2N$ time steps.

and the states of its neighbors. For example, for the case $k = 7$, an agent "knows" its own state and that of its six neighbors. Thus there are $2^7 = 128$ possible input patterns that a given agent must be able to lookup. Therefore, the size of the string encoding the rule for $k = 7$ is 128 bits (Fig. 1).

Several papers have investigated the density classification task. Early papers used genetic algorithms to evolve efficient strategies for global coordination [6, 7] in systems with agents placed in ordered networks and with completely noiseless communication. The efficient strategies they found were somewhat complex and fragile in varied environmental conditions. More recent work by Moreira *et al.* [1] focuses on a simple majority rule, in which an agent adopts the state of the majority of its neighbors. This strategy fails in noiseless environments, but Moreira *et al* show that it is very efficient in noisy small world networks. The small world structure provides the agents with fast access to global information. For large systems, however, there is a high probability that a small group of units will be isolated, and thus

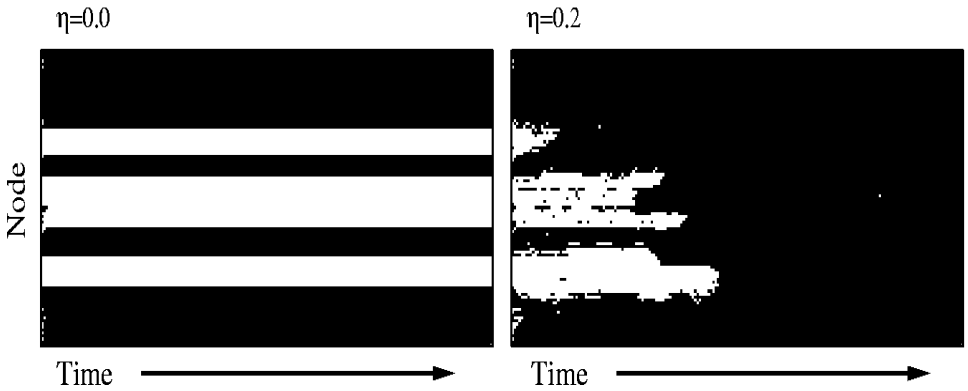

Figure 2: Effect of noise intensity η and rewiring probability ρ on the time evolution of an $N = 99$ system of $k = 7$ cellular automata operating according to the majority rule (time progresses from left to right) [1]. The figure on the left ($\rho = 0.2$ and $\eta = 0$) shows a system that never reaches consensus because the boundary between minority states and the majority state, cannot be broken. The figure on the right ($\rho = 0.2$ and $\eta = 0.2$) shows a system that is able to converge to the initial majority state because noise disrupts the boundaries between opposing states making internally wired "cliques" collapse.

form a persistent minority state. The boundary between the majority region and minority region cannot be broken in a noiseless environment. For this reason noise is necessary – it "breaks" the boundaries between the multiple domains, enabling the global majority state to prevail (Fig. 2) [1]. While the research of Moreira *et al* shows that the majority rule is a simple heuristic that leads to efficient global coordination in a natural environment, it did not investigate how a majority rule might evolve. In this paper, I use genetic algorithms to investigate the evolution of strategies in noisy environments.

Noise – Since some noise is present in all natural systems, it is important to incorporate it into the model. To incorporate environmental noise, the model assigns a small probability that a given node "misreads" the information from a neighbor when updating its state. The noise parameter, η, ranges in intensity from $\eta = 0$ (noiseless), to $\eta = 1$ (random dynamic).

System Topology – Another important factor in this model is the presence of a

small-world network. Small worlds are found in numerous natural systems, such as the nervous system of the *C. elegans* worm and some human social networks [5]. This network structure permits fast spread of information, which helps drive the system toward classification [1]. The network is set up by creating an ordered array (or lattice) of nodes, each with a link to itself and its nearest $\frac{k-1}{2}$ neighbors on each side. Then, with a small probability ρ, each link in the network is rewired. When rewiring, the links are changed to a randomly selected node in the system. For small values of ρ, the network becomes what Watts and Strogatz termed a small-world network: a network with local cliquishness and short mean distances between any two nodes [5].

The efficiency function – An updating rule ϕ's efficiency, $E_\phi(\rho, \eta, \mathrm{N})$, is a function of the noise η, the rewiring probability ρ, and the system size N [1]. The density classification task is completed if *all* of the units converge to the majority state. No partial credit is given. The efficiency is the percentage of a set of initial conditions that a given rule ϕ correctly classifies. Since it is harder to determine the majority state for initial conditions with an approximately equal number of 1's and 0's, the initial conditions are chosen with some with bias. They are distributed evenly from an initial condition of all 1's to an initial condition of all 0's. There are always the same number of initial conditions with a majority of 1's as there are with a majority of 0's.

Every time step the rule is tested for convergence by checking the next time step without noise, to see if all units are in the same state. If the system has reached a consensus, and remains so in the next time step, then the updating is halted. If it has converged to the correct state for that initial condition, it is counted as a success. If no consensus is reached after 2N time steps, it is assumed to have failed on that initial condition.

The Genetic Algorithm – Since for $k = 7$ the rule size is 128 bits, there are 2^{128} possible rules: far to many to test exhaustively. For larger systems with a slightly greater k value, even $k = 9$, the number of possible rules will greatly exceed the number of atoms in the known universe. While some efficient strategies for the density classification task are already known, it is necessary to understand what strategies, if any, evolve, how they evolve, and why they evolve. Genetic algorithms, which work by applying principles of evolution, provide an excellent way to model the development of strategies for global coordination.

The genetic material being evolved, in this case, is the rules. To form a new generation, rules undergo "crossover" (described below) and mutation, and are assigned a fitness value which is a measure of their ability to perform. For this model the fitness value is the rule's efficiency as described above. The rules in the population are ranked by fitness, and the top 20% are selected. The remaining 80% of the next generation are obtained by crossovers of randomly selected survivors [3].

Many genetic algorithms use a single-point crossover method – two randomly selected "fit" rules swap two pieces around a randomly selected point in their binary string encodings to produce a "child." A weakness of this technique is that the placement of bits in a rule string is arbitrarily chosen. For the single point crossover to work effectively, good "genes", or schemas, as they are called, must be crossed. This, however, is dependent on the location of the bits. Since the locations are arbitrary (in this case they are in the lexicographical order of input patterns), it is possible that the key bits in good schemas are not localized, and therefore will not crossover. This is known as the linkage problem [3]. In my algorithm, the crossover method was perhaps more "artificial" but attempts to address the linkage problem. A random integer, j, between 0 and $k - 1$ inclusive, was chosen for every crossover.

The two rules are then combined by adding pieces of size 2^j, alternating which rule the piece is taken from. In other words, if the j^{th} bit of an input pattern is a "1", the "child" uses the "mother's" output bit. If it is a "0", it uses the "father's" output bit. Thus, it is clear the crossover method is independent of any lexicographical order. In this way, evolution can crossover larger chunks of local bits, which often contain a good "gene", as well as crossover bits from a more diverse sampling of the population. This helps reduce fluctuations caused by the linkage problem. After this crossover, the new rule undergoes mutation of a few bits. The number of mutations varies depending on the rule size, but is always less than 5% of the number of bits in the rule.

Rules – Before the evolution, the initial rule strings are chosen either completely randomly, or randomly according to a flat distribution, depending on what data was being collected. In the flat distribution, the first rule is all 0's, and the last rule is all 1's. Each rule has one more 1 than the rule before it. This way there are some rules that have a fitness of 0.5. These are the rules that make a consistent guess: the rules whose encodings are almost all 1's or almost all 0's. No matter what the initial condition, they always converge to the same state. These rules have an efficiency of 0.5 because they correctly classify half of the initial conditions. A fitness of 0.5 is actually quite good, considering that the vast majority of rules have a fitness of 0.

Creating the initial rules in this biased way gives the system a push, helping the evolution get off the ground. However, this is also somewhat unnatural, which is why this paper explores completely random initial conditions as well.

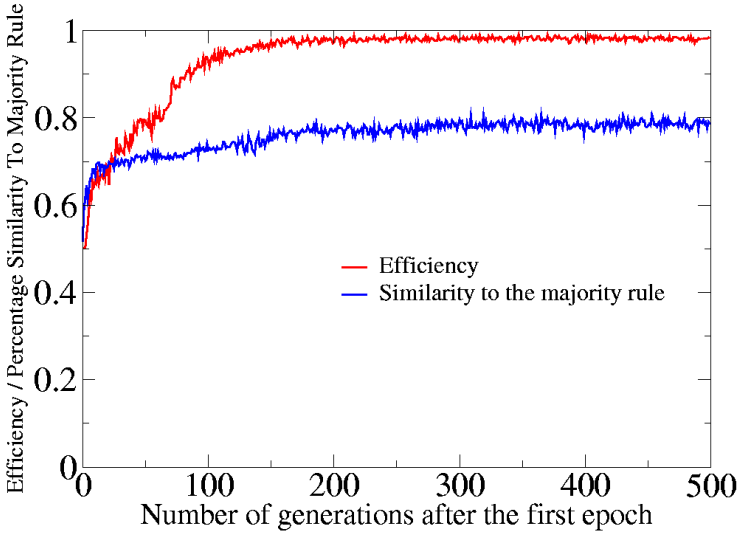

Figure 3: The average of six trials – the efficiency of the best rules at each generation and their similarity to the majority rule under the conditions $k = 7$, system size N = 99, rewiring probability $\rho = 0.5$ and noise $\eta = 0.1$. The first epoch occurs at the first time step in which a rule that has a fitness above 0.5 is present. As the evolution proceeds, the fitness reaches the maximum (nearly 100 percent classification efficiency) within 200 evolutions of the first epoch. The rule's similarity to the majority rule, however, tops out at about 80 percent.

3 Results

Initially, rules were evolved for the $k = 7$ case with system size $N = 99$, to see if rules do in fact evolve toward the majority rule. The data shows that as the evolution proceeds, the best rule's similarity to the majority increases initially, but quickly plateaus at 80% similarity (Fig. 3). The figure shows the average of half a dozen trials – there was not significant deviation from the average in any single trial. Fig. 3 suggests that there is a class of rules with a large similarity to the majority rule, that have high efficiency (0.96 and greater). There is no evolutionary incentive to push rules closer to the majority rule, because they are already equally efficient. While one might hypothesize that the mechanism for reaching consensus is similar to the one observed in the majority rule [1], the evolved rules are only 80% similar to the majority rule and it is therefore difficult to know how similar their mechanisms are.

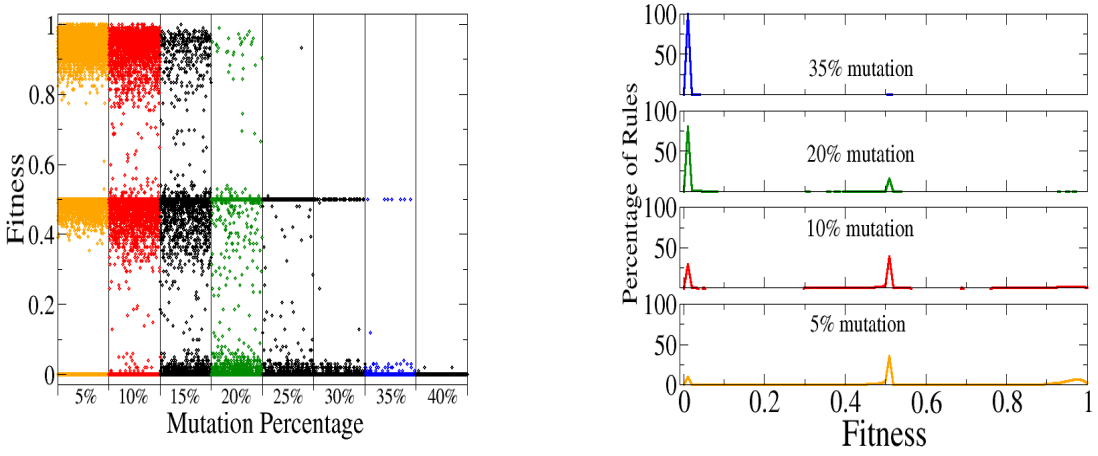

Figure 4: Fitness of majority rule degradations: random mutations of the majority rule by X% where $N = 399$, $k = 7$, $\eta = 0.2$ and $\rho = 0.2$. Ten thousand rules were randomly generated for each mutation percentage. While this is only a very small percentage of the total number of rules, it gives an idea of the densities of fitness values. The range of fitness values spans nearly the entire spectrum when 80% similar or greater, but most fitness values fall into three categories of relatively high densities: 0, 0.5 and above 0.9. The density of the efficient rules decreases drastically as its distance from the majority increases. The normalized histogram on the right shows the percentage of rules with a fitness of 0 increases dramatically as the mutation percentage increases. Even at 5% and 10% mutation, the number of rules above 0.8 fitness is small.

A scatter plot of the efficiencies of rules that were varying distances from the majority rule (randomly mutated) reveals that the density of efficient rules, even those very similar to the majority, is quite small (Fig. 4). There are several rules with efficiencies close to 0.5 (meaning they correctly classify half of the initial conditions) and many with an efficiency of 0. This also indicates that not all bits are of equal value. Since clearly not all rules that are 80% similar to the majority rule are highly fit, it is necessary to understand what other conditions must be satisfied for a rule to have a high efficiency.

Rule Dynamics – To discover more about the class of efficient rules, several rules were evolved, using biased initial conditions. The rules' final bit string representations were saved. After selecting the top performers from 100 trials, I looked at the percentage of rules that shared each bit with the majority rule. Several bits were the

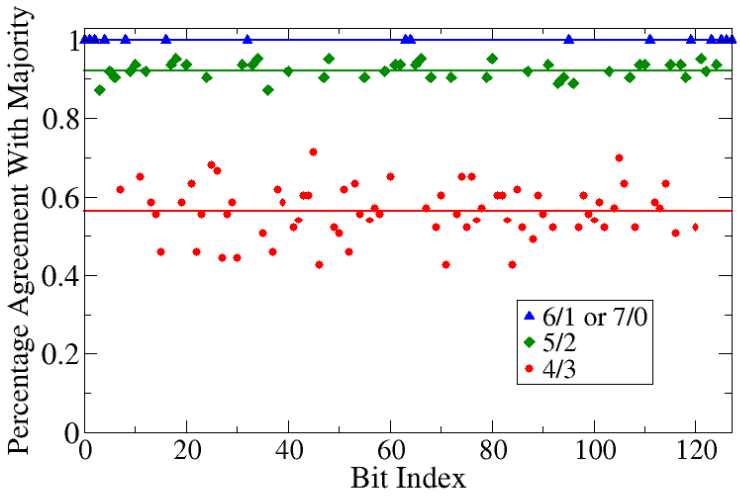

Figure 5: Percentage of evolved rules containing each bit. The importance of bits in a 128 bit rule, where $k = 7$, $N = 99$, $\eta = 0.1$ and $\rho = 0.5$, is related to the ratio of input bits. The graph shows the percentage of rules that agree with the majority for each bit. It is categorized by the ratio of bits in the input pattern. For $k = 7$, bits corresponding to an input pattern ratio of 7-0 or 6-1 agree with the majority in all efficient rules, whereas 4-3 ratio bits average only 56% agreement. Bits corresponding to a 5-2 ratio average 92% agreement. This data seems to suggest that it is the ratio of bits in the input pattern that is significant, not the exact pattern.

same in 100% of the highly fit evolved rules and in the majority rule, whereas other bits agreed with the majority in only about 50% of the rules. This is concrete evidence that certain bits are *necessary* in an efficient rule, while others have little effect. Further analysis reveals that the importance of a bit is related to the weighting of 1's to 0's in the input pattern corresponding to that bit. For example, Fig. 5 shows that if the ratio of 1's to 0's is 3-4 or 4-3, the corresponding bits agree with the majority rule only 55% of the time. Bits whose input pattern has a bit ratio of 7-0 or 6-1, however, must be identical to the majority rule bits for *all* efficient rules.

In other words, what Fig 5 demonstrates is that bits, whose corresponding inputs are increasingly further from a 1:1 ratio, become increasingly more essential. If the heavily weighted bits (for example those with a 7-0 or 6-1 ratio of input patterns)

disagree with the majority, it is impossible for the rule to reach the correct consensus, because as the system gets close to convergence, these bits perpetuate a minority state. In general, for *any* rule of size 2^k, the bits corresponding to an input pattern ratio of k to 0 or $k-1$ to 1 *must* agree with the majority in fit rules. While these heavily weighted bits will always agree with the majority rule, the data suggests that agreement of the "fringe" bits hinges on the noise present in the environment. Furthermore, it is not the exact input pattern that is significant, it is the ratio of bits in the inputs pattern.

Dynamics of the Genetic Algorithm – In evolving rules several unanticipated problems arose. Importantly, when the system size, N, was increased beyond 99, strategies with $k = 7$ connections or greater were unable to evolve past a fitness of 0.5. An efficiency of 0.5 can be a critical boundary in the evolution of the rules. Since half of the initial conditions have a majority in one state, and half a majority in the other, a fitness of 0.5 means that the system can reach the correct consensus for initial conditions with one majority, but not with the other. Finding a rule that can correctly converge to the majority state for both types of initial conditions is *very* difficult.

Determining the relative importance of bits in the rule sheds light on this problem. For a $k = 7$ rule, 16 out of the 128 bits are in 100% of the efficient rules. These are the bits that correspond to an input pattern bit ratio of 7-0 or 6-1. Rules with a fitness of 0.5 have *at least* half of these bits agreeing with the majority. The reason that it is not difficult to reach an efficiency greater than 0 for some trials is that the initial rules are chosen with bias – there are rules where all the bits are 1's and all the bits are 0's. These rules start with the important bits of one majority state already in place. For a rule to start correctly classifying the other initial conditions, *all* of the remaining key bits must agree with the majority. For $k = 7$ there are 8 of these

bits. Thus, out of $2^8 = 256$ possible combinations, only one will produce the desired outcome.

What makes it especially difficult for these bits to fall into place is that to improve to a fitness beyond 0.5 *all* of these bits must have the correct value. Each bit alone cannot improve the fitness and therefore there is no evolutionary pressure to make the rules improve gradually. Only a crossover between two "guessing" rules, each of which has one half of the necessary bits, can produce a rule with an efficiency above 0.5. This is the reason that the arbitrary lexicographical order of bits in the rule can be detrimental. While the crossover method used in this study helps some with this linkage problem, it is never possible to perfectly crossover the important bits.

When one considers that the initial rules were chosen with bias, it becomes even more clear that the natural evolution of strategies is extremely difficult. Choosing the initial rules completely randomly makes it nearly impossible for rules to evolve. There are no longer any "guessing" rules with a fitness of 0.5 in the initial population. Rules must overcome two barriers: 0 *and* 0.5. The only rules that are likely to get past a fitness of 0 are small rules, for example the $k = 5$ case. To achieve a fitness greater than 0, six key bits must fall into place. This means that one in 64 rules would start out with a fitness greater than 0 if the initial rules were randomly chosen. However, for general k, the chance that this occurs is $\frac{1}{2^{k+1}}$, so it becomes nearly impossible for even moderately sized values of k.

The Average Output Function – Since these rules were evolved with a fixed value of noise it is not clear how noise effects the makeup of the rules. In order to determine a relationship between the environmental noise and rule evolution, several $k = 5$ rules were evolved, using completely random initial conditions, at different values of noise. Fig 6 shows the relationship between noise, and the similarity of the rules to the

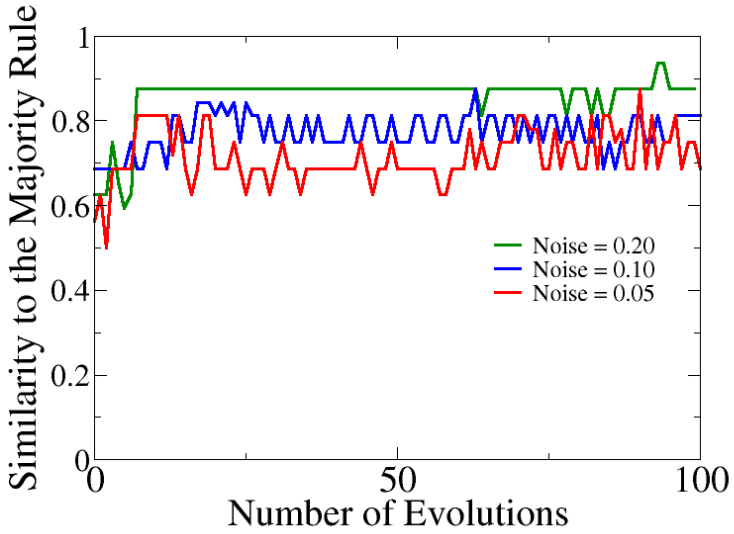

Figure 6: Similarity to majority rule over 100 evolutions for rules evolved at different noises. $\rho = 0.2$, $k = 5$ and $N = 399$. Rules evolved at larger noise values plateau at a higher average similarity. The fact that they are present in noisier environments means that there is less room for them to have any "internal" noise (due to disagreement with the majority rule).

majority rule. As the value of noise, η, decreases, the rules' average similarity to the majority decreases. This is in fact not surprising because as the environmental noise decreases, an evolving rule can have more "internal" noise caused by dissimilarity to the majority rule.

To truly understand this phenomenon, it is necessary to consider the various evolutionary pressures that are exerted on these rules. There are two key properties for highly fit rules: they must be highly accurate, that is, converge to the correct final state, and they must do so quickly (in this case in $2N$ time steps). However, these properties are favored by noise intensities on different sides of the spectrum. For high accuracy, a small value of noise is clearly favored. For high speed on the other hand, *high* noise intensity is actually favored. Moreira *et al* show that noise is required to break the boundaries between opposite states, and the more noise that is present, the

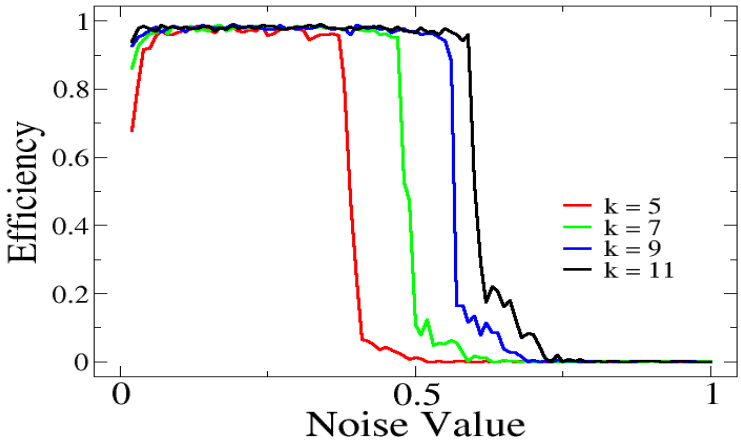

Figure 7: Fitness of majority rules at $\rho = 0.2$ and different noises. The $k = 5$ majority rule's efficiency drops sharply after noise values of about 0.37. Larger rules are more efficient in noisy environments because the majority state of the input pattern is less likely to be switched with rules that have more neighbors.

faster this occurs (Fig. 2, [1]). Thus, evolution must find a rule and noise value (that is, internal noise and external noise combined) which optimizes both accuracy *and* speed.

It is instructive to examine the average output of a rule versus the ratio of bits in the input pattern. The majority rule with no noise always outputs a 0 if the input contains a majority of 0's and a 1 if there are a majority of 1's in the input pattern. Once any noise, internal or external, is present in the rules, the function takes on a logistic shape [8]. I plotted the curves for several different rules and noises. These are the graphs of the average output functions for particular rules and noise values. Evolution does not pressure rules to evolve to the majority rule (or any other optimal rule), but it may find an optimal average output function that is essentially independent of the environmental noise. If evolution does find an optimal average output function based on the "internal" and "external" noise, the majority rule at

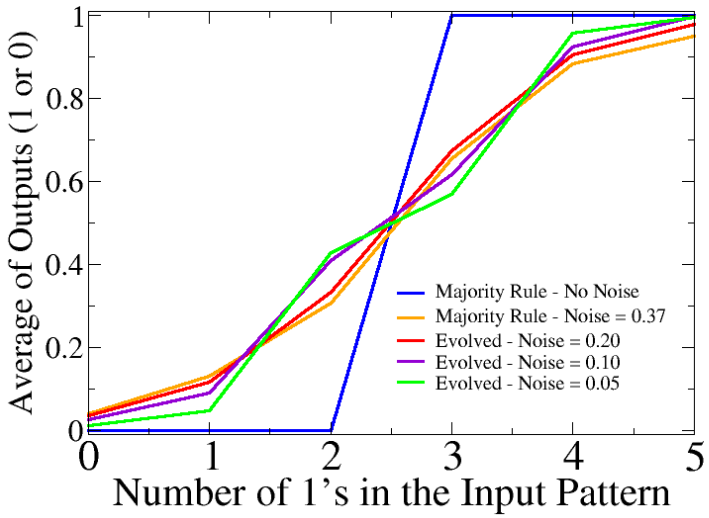

Figure 8: Average output versus input pattern. The majority rule tested at $\eta = 0.37$ (the highest noise value where the $k = 5$ majority rule is still efficient – Fig. 7) produces essentially the same curve as rules evolved and tested at $\eta = 0.2$, and as rules evolved and tested at $\eta = 0.1$. This suggests that this curve is the optimal average output function and incorporates both internal and external noise.

the maximum environmental noise value (Fig. 7) should produce the same curve as evolved rules at the noise value where they were evolved.

Fig. 8 confirms that this is indeed the case. Four average output curves, three evolved at different noises and with different similarities to the majority, and the majority rule at $\eta = 0.37$ all lie on essentially the same curve. The majority rule, which has no "internal" noise, tolerates the highest environmental noise. The other rules do evolve "internal" noise because their "external" noise is not as high.

The rules evolved at the lowest noise values ($\eta = 0.05$ or $\eta = 0.1$) diverge the most from the apparent optimal output function (Fig. 8). This makes sense, because these rules have the highest "internal" noise. I previously showed that all $k - 1$ to 1 ratio bits *must* agree with the majority rule, which is why the 4-1 bit points diverge the most from the curve (they are a special case). Perhaps the 2-3 ratio bits are more

noisy than might be expected because the 1-4 ratio bits cannot have any internal noise. What is very important to realize is that on average they still lie on the same curve and this is not a coincidence. Perhaps for larger rules (e.g. $k = 7$ or $k = 9$) the optimal curve would be clearer to make out because their are more bits, and thus more points on the curve.

Curves with sharper transitions than the optimal case (e.g. the majority rule with no noise) represent functions that are perhaps more accurate, but they likely do not converge quickly enough. Curves that are less sharp (flatter) represent functions that can quickly break the boundaries between opposing states, but are not accurate enough. The evolutionary pressure is such that as the environmental (external) noise increases, the rules are forced to become more similar to the majority rule so that they maintain the optimal function. The functions want to maximize the noise, without compromising accuracy and evolution successfully does this. This is consistent with the data in Fig. 6, where the rules that were evolved in less noisy environments are less similar to the majority rule.

Our new understanding of the optimal average output function (Fig. 8) raises interesting new questions. Evolving rules starting with relatively high noise value η gets increasingly more difficult. For rules to be somewhat fit, their average output curves must be relatively close to the discovered optimal curve. If rules begin evolving in an already noisy environment, the rule encoding must *initially* be relatively close to the majority rule (low internal noise) for its fitness to be improved. This presents a clear problem, because if the initial rule population is random, fitness will not increase unless a rule already somewhat close to the majority rule is present.

All of this data suggests that evolution needs to start with simple problems in "easy" environments, evolve solutions, and then use those solutions as templates for

the more complex problems. It is too difficult to evolve solutions for complex problems from random initial conditions. Rules with $k > 5$ or for high noise values become essentially impossible to evolve. If rules are evolved at $k = 5$ first, and then there is some pressure for it to evolve further (such as an increase in noise beyond the $k = 5$ efficient region, Fig. 7) then perhaps larger rules can evolve.

Finally, recognizing that there is in fact an optimal average output function that is based on a composite of a rule's "internal" noise and the "external" noise has implications for all research regarding decentralized systems. It also gives insight into the evolutionary process and its ability to optimize solutions when there are conflicting pressures. The majority rule is significant because it has no "internal" noise. It is now clear that it isn't only the rule itself that is significant, but the function that is implemented due in part to the rules makeup, and in part to the environmental noise.

4 Discussion

The results described in this paper are an indication of the evolutionary value of majority-type rules in reaching consensus. These rules seem to follow a natural strategy that has co-evolved over time with networks and noise levels – a fact that may have significant implications for any natural decentralized system. A majority-type mechanism may be relevant to everything from school choice decisions where social networks lead to a "local structure of feeling" [9], to neuronal coordination in scores of different species. It even has applications in economic decision making [8]. Although the model used here is simple, it exhibits complex behavior like that found in many real world systems. An understanding of this model may lead to a better understanding of natural decentralized systems.

Perhaps most importantly, the data suggest that evolution may build complex solutions based on solutions to simpler problems. Evolution did not start with chromosomes containing millions of DNA nucleotides, but rather started evolving smaller chromosomes eventually working its way up to the chromosome size of today. It is not immediately clear how evolution uses simpler pieces as a template for a more complex "chromosome", but this is something that clearly needs to be explored. Some scientists have examined growing the "chromosome" size, but not increasing the difficulty of the problem to be solved [3]. My results suggest that not only should the initial rules evolve, but the complexity of the problem should as well (e.g. the noise level).

This data opens many doors in the study of the coordination of decentralized systems and provides insight into important evolutionary mechanisms. I identify the specific curve that optimizes the density classification task and explore its meaning. It is now clear what mechanisms produce naturally efficient strategies and in what environments they are efficient. Majority rule decisions should be a factor in any research regarding natural decentralized systems, especially if the average output function is in fact a universal function for system-wide coordination. There are still many questions, such as what rules would evolve if each node followed a different rule or had a different threshold. Finally, rules for $k = 9$ or $k = 11$ could be evolved by starting from smaller rules with smaller k values and increasing both the noise and k. This may offer insight into the evolutionary process and how feasible it is to grow "chromosome" size for genetic algorithm problem solving.

5 Acknowledgements

I would like to thank Professor Luis Amaral and Doctor Andre Moreira for the time they so graciously took from their busy schedules to mentor me on this project. I

would also like to thank my physics teacher Doctor Mark Vondracek for his inspiration and support throughout the research process, and my parents for their patience and encouragement.

References

[1] Moreira, A., Mathur, A., Diermeir, D., and Amaral, A.N. (2004). Efficient System-wide Coordination in Noisy Environments. *Proc. Natl. Acad. Sci. U.S.A.* **101** (33), 12085–12090.

[2] Wolfram, S. (2002). *A New Kind of Science.* (Wolfram Media, Champaign, IL).

[3] Mitchell, M. (1998). *An Introduction to Genetic Algorithms* (MIT Press, Cambridge, MA).

[4] Amaral, L.A.N. and Ottino, J.M.. (2004). Complex Networks: Augmenting the framework for the study of complex systems. *Eur. Phys. J.B* **38** 147–162.

[5] Watts, D. J. (1999). *Small Worlds.* (Princeton University Press, New York, NY).

[6] Crutchfield, J. P and Mitchell, M. The evolution of emergent computation. (1995). *Proc. Natl. Acad. Sci. U.S.A.* **92**, 10742.

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

[8] Diermeier, D. and Van Mieghem, J., Spontaneous Collective Action. CMSEMS Discussion paper No. 1302. Northwestern University. August 2000. Revised August 2001.

[9] Ball, S. and Vincent, C. (1998). 'I Heard It on the Grapevine': 'hot' knowledge and school choice. *British Journal of Sociology of Education* **19**, 377–400.