Exploring Phenotypic Robustness with Evolved Cellular Automata

Thomas Adams, Carolyn Atterbury and Nitin Bhandari

***Abstract*— This paper reviews and attempts to reproduce the results in the paper Evolving Cellular Automata to Perform Computations: Mechanisms and Impediments, by Mitchell et al. Similar to Mitchell's work, we evolved a cellular automaton rule (CA) by using a genetic algorithm (GA) to perform one-dimensional majority classification. After finding the optimal rule set, we generated a neutral network by mutating the original rule-set and found that its neutral network consisted of 43% of rules 3 mutations away and contained at least one rule with almost 75% of its genotype changed. We then sampled a new population from the neutral network and ran that population against a new fitness function in order to verify the claim by A. Wagner that the mutational robustness allows for quick adaptation to change. Our results show that the networked populations evolved 100 times as quickly as a control population. Finally, we show that the neutral network contains yet better solutions to the hardest majority classification problems, but they are not usually found as they are worse on easier problems.**

I. INTRODUCTION

Cellular automata (CAs) are a form of discrete simulation, consisting of a set of cells with a finite number of states. Each cell has a corresponding neighborhood of cells, relative to the cell's position[2]. At each time step, each cell changes its state depending on the state of each of the cells in its neighborhood, and these changes are defined in a rule defined for the CA. Over time, the row of cells will either reach a stable state, it will oscillate between states, or it will produce random or pseudorandom behavior. In this paper we will be using a Genetic Algorithm (GA) to evolve a CA rule. A GA starts with an initial population. At the end of each iteration, we select a group of the most fit individuals in the population, and use mutations or crossover to generate the population for the next iteration. As the process continues, the population fitness generally increases and after 100 generations we arrive to a best fit rule evolved through the process. Here we tried to replicate the process of evolving a CA rule based on fitness of majority rule by Melanie Mitchell [3], and expanded on her analysis using the findings of A. Wagner [1] with regard to neutral networks.

A neutral network is, in this case, a set of rules whose fitness remains the same despite the fact that the rules differ from each other. This property of different genotypes producing same phenotype was taken from Wagner's findings, in which he showed that populations with higher neutral networks are more successful when environmental changes occur. We tested this hypothesis by running network nodes through a GA with a different fitness function f' which ranks rules based on their ability to perform a slightly different "supermajority" classification. After comparing the results it was found that even after changing the fitness function the network was robust to environmental changes and still displayed (approximately) correct behavior. In analysing the depth of the neutral network, we found that in some cases rule sets could differ in 75% of the code and still maintain similar results and high fitness compared to other rule sets in the network. We also found that some rule sets performed better on easier classification problems, whereas other rule sets performed better on harder classification problems. This finding indicates that the neutral network is robust against various environmental conditions.

Through these findings we found that simple function performing cellular automata can collectively produce a complex behaviour which can adapt to changes and display phenomenal results like robustness in the face of genetic mutations.

II. EVOLVING THE CA

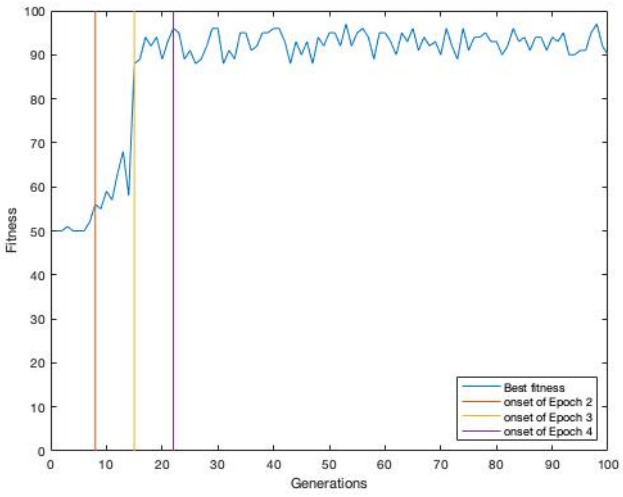

Fig. 1. The onsets of four epochs of innovation each corresponding to the generation discovery of a new, fitter strategyare marked.

To replicate the results of the Mitchell paper[3], we created a 1D CA, and evolved it using a GA. The CA had a neighborhood size of 7, meaning the rule-set size was 128, where each "gene" could be either 1 or 0.

The initial GA population consisted of 100 CA rule sets which were each randomly generated and uniform with respect to λ. We tested the rule set on 100 uniformly randomly generated initial conditions (ICs), and picked 20 rule sets that had the highest fitness. The fitness of each rule set was determined by the fraction of the 100 ICs on which the rule produces the correct final pattern. If the IC

has majority 1's, then the correct final pattern will be all 1's. Otherwise, the correct final pattern should be all 0's. It should be noted that we enforce the constraint on the IC's that half of them should reduce to all 1's as the correct answer and half should reduce to all 0's as the correct answer, in order to keep random deviation of IC's from harming the evolutionary path of the rules.

After picking the 20 most fit rule sets, we randomly select two of the top 20 and perform a crossover mutation[3]. The crossover processes produces two new rules for the next population, performing two random mutations (single bit-flips) on each new rule. We do the crossover process 40 times to create 80 new rule sets, for a total of 100 rules for the next generation. We go over this process in more detail in section 7.

As seen in Figure 1, we evolved a CA rule-set in 100 generations. Similar to the Mitchell paper, we observed 4 epochs, although our epochs generally occurred earlier than in the Mitchell paper. We had a more pronounced jump in fitness from epoch 2 and 3, meaning the fitness gain from epoch 3 to epoch 4 was less pronounced than in the Mitchell paper. These subtle differences could have to do with the slight differences in our crossover method. For reference, maps of the execution of rules in epochs 2 and 4 can be seen in 2 and 3. These executions yielded correct and incorrect results, respectively, and both exhibit the signal transferring behaviors noted by Mitchell.

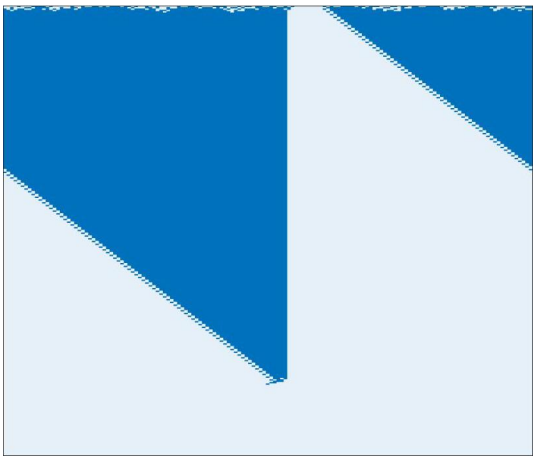

Fig. 2. The execution of the best evolved CA rule on an initial condition with 69/149 ones. The rule worked correctly in this case.

In Figure 4, we compare the fitness of the evolved rule and GKL[3], a successful rule that solves this problem, as the proportion of 1s to 0s in the IC changes. For brevity, such figures will be referred to as fitness landscapes in the future. At $\rho_0 = 0.5$, when the IC is close to being half 0s and half 1s, the fitness decreases to between 40 and 50 percent for ICs with length of 149. The fitness decreases for ICs of larger lengths, which is not unexpected. The Mitchell paper had better results for the 599 and 999 case, which we attribute to more stringent rules on computation length; where Mitchell allows for 10*N steps for each rule, we only allow 320/149*N steps; we kept this both because of time constraints and because it shows the comparative slowness of the evolved rules with respect to GKL.

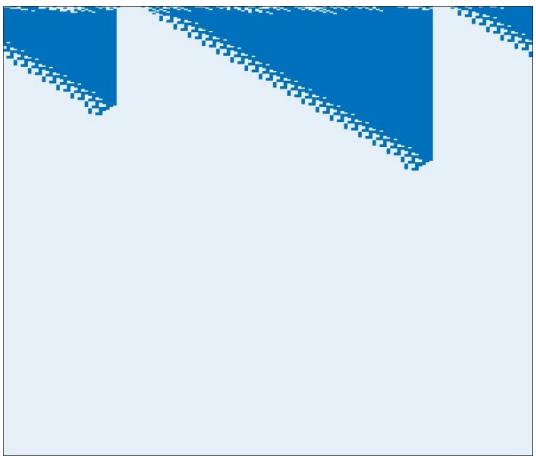

Fig. 3. The execution of the fourteenth evolved CA rule on an initial condition with 77/149 ones. The rule worked incorrectly in this case.

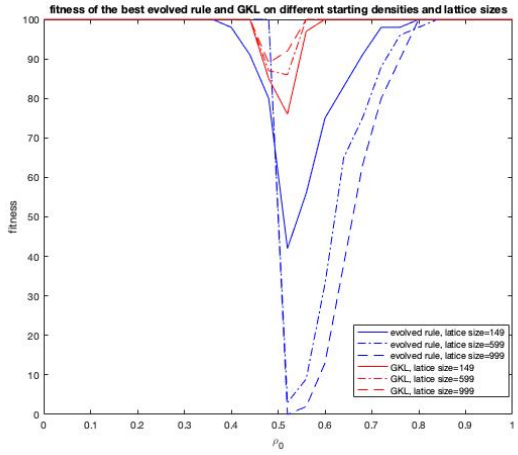

Fig. 4. Fitness landscapes of the evolved rule (blue) and the GKL rule (red) for lattice sizes 149 (solid), 599 (dot-dashed), and 999 (dashed)

III. Generating a Neutral Network

After a series of crossovers and mutation, we came up with a fittest cellular automaton rule set which was able to give the desired results. We refer to the most fit rule set as CA_0. This CA is basically a simple majority classifier meaning that, given a random set of 0s and 1s as an IC, if the IC consists of majority 1s then CA_0 will convert the cells to all 1s, otherwise it will convert to all 0's.

Using CA_0 as a base genotype, we created a neutral network which consists of all the neutral mutants of CA_0. A neutral network is a network consisting of different genotypes which are mutated from the base genotype (in this case, a change in the bit string of a rule), but produces the same phenotype (fitness results)[1]. Here a rule set is considered to have the same phenotype if its score is at least 3 less than

the score for CA_0. In this case we test each mutated rule set on 100 uniformly randomly generated ICs, and check to see if its score is comparable to CA_0. Starting at CA_0, we built up a neural network of mutants with hamming distances of 1, 2, and 3 bits.

To create a neutral network we used a simple breadth-first-search and tried to changed each and every bit of CA_0 one at a time. For each change, we checked whether the mutated CA has the same phenotype i.e. it displays the same property of majority rule. The CA which were 1-bit mutation away had just one bit flipped out of 128. CAs that were 2 mutations away had 2 bits flipped, and CAs 3 mutations away had 3 bits flipped from the original CA_0. The network is made in a breadth-first search fashion where we search all mutations that are 1 bit away before moving to mutations that are 2 away. In Figure 5, we plotted a bar graph displaying the proportions of nodes in the neural network out of all possible combinations with respect to the number of bit mutations away from CA_0. We found that 91 out of 128 possible 1-bit mutants, 4061 out of 8128 2-bit mutants and 147767 out of 341376 3-bit mutants still exhibited the same phenotype. Since around 70% of the 1 bit mutations were added to the neutral network, and 50% of 2 bit mutations were added, we already have a sense that the neutral network will be pretty large. These proportions are shown in Figure 5 below.

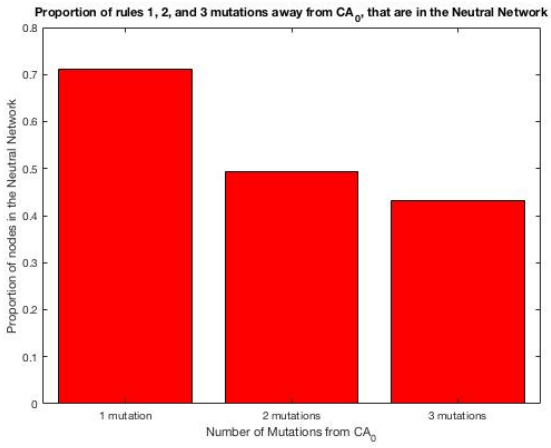

Fig. 5. The number of mutants in the neutral network as a proportion of possible mutants for each hamming distance sampled

After mapping some of the neutral network we tried to test Wagner's hypothesis which states that a neutral network increases robustness to environmental perturbations[1]. We were interested to see how different starting populations reacted to a different fitness function when put through the GA. To evaluate this hypothesis we created a new fitness function, f', to run through the GA. Instead of picking a simple majority, f' produces a final pattern of 1's only if the initial input has at least 60% ones. To test the hypothesis, we then took 100 genotypes (rule sets) from the neutral network and used them as an initial population to use with the f' fitness function in the GA described previously. The starting population had a high fitness going into the GA, as seen in

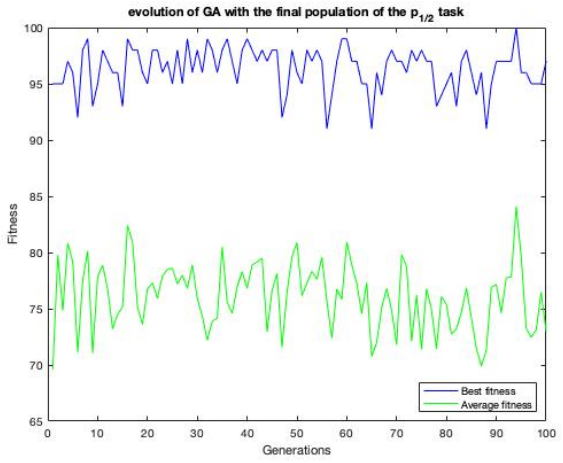

Fig. 6. Evolution trajectories for GA's seeded from the original GA

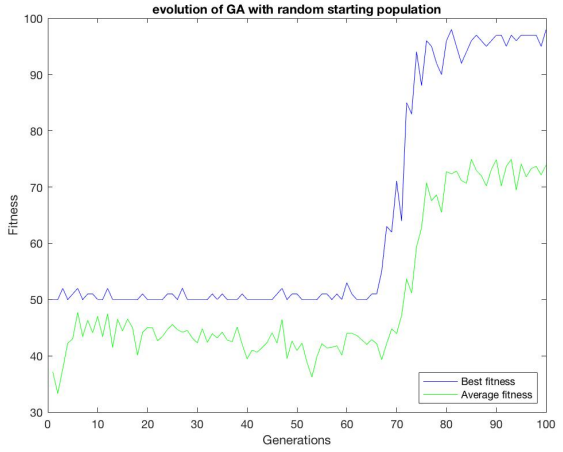

Fig. 7. Evolution trajectories for GA's seeded with random populations

Figure 8. In this case the rule set with the highest fitness had a starting fitness around 95, while the average fitness amongst the rule sets had a starting fitness around 80.

To get a control for this new task, we also took 100 initial random rule sets and used them to evolve a better rule set using f as the classification function. Using this initial population, the GA took a very long time to evolve at all, but quickly reached a comparable level after long enough. This is not an atypical amount of time, and sometimes the GA would fail to evolve anything within 100 generations. This shows that, if anything, the f' fitness function seems more difficult to satisfy with a CA than the original fitness function.

To further test the hypothesis, we ran the GA a third time with a starting population pulled from the ending population of our initial GA with the original fitness function f. Similar to the population pulled from the neutral network, the third population had an average starting fitness around 75, and a best fitness around 95. This can be seen in Figure 6. This result seems to show that the rules explored by the original

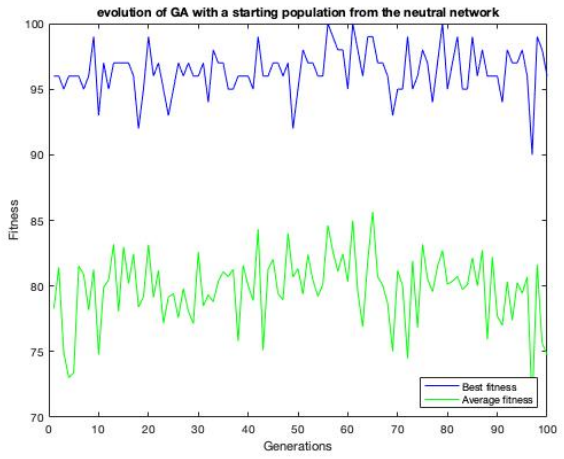

Fig. 8. Evolution trajectories for GA's seeded from the neutral network

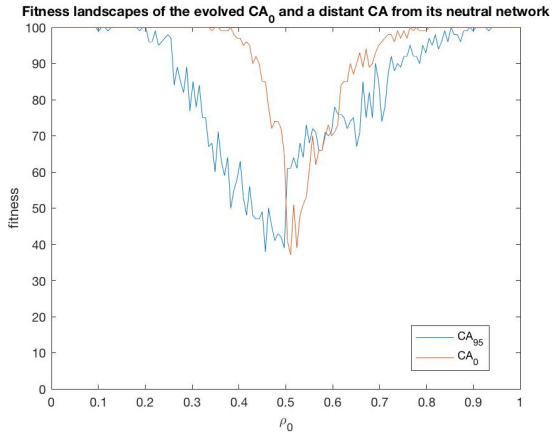

Fig. 9. Fitness landscapes of the CA_0 (red) and CA_{95} (blue). Note the differences between the rules as they are extremely distantly related, and that there are places (e.g. around $\rho_0 = 0.55$) where CA_{95} outperforms CA_0

GA at any one time include a number of rules from the neutral network or a similar space, as the final population for f obviously had enough genetic diversity to evolve good rules quickly.

Of the three populations, the randomly generated initial population performed the worst, seen in Figure 7. This is to be expected as its starting population was completely random, though the extent of its inferiority was rather surprising. Even though the populations from the neutral network had not been directly evolved from the GA, they maintained high fitness due to proximity to CA_0 which performed the best with f. By creating a new fitness function f', we were able to test how the various initial populations performed under a different environment. We found that the populations from f were robust against environmental changes, as they each started with a high fitness level when facing a new fitness function.

IV. Further Analysis on Neutral Network

To get a better sense of the size of the neutral network, we used a depth first search algorithm to find the largest number of mutations that lead to neutral behavior. Starting with CA_0, we found a total of 13,305 nodes over the course of a full day, the most distant of which had a hamming distance of 95 from the original. This means that the largest branch found was 95 nodes deep, and only 2 branches that were 2 mutations deep had been fully explored by that point. We do not know the exact size of the full neutral network, but we assume that it must be very large. The leaf node on the longest branch in the neutral DFS tree we refer to as CA_95, as it is 95 mutations away from CA_0. In discovering CA_{95}, we know that up to 75% of the CA_0 genome can be changed where it does not affect the phenotype.

We compared the fitness of CA_0 and CA_{95}, as seen in Figure 9. Overall CA_0 performs better most ICs, but CA_{95} performs better close to the $\rho_0 = 0.5$ mark. In an environment where the number of 1s and 0s is slightly above 50%, CA_95 would be the CA rule set with higher fitness. In environments where there is more of a clear majority between 1s and 0s, CA_0 would be the rule set with higher fitness.

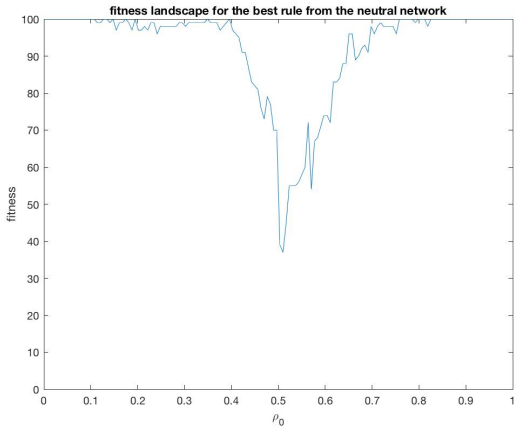

Fig. 10. Fitness landscape of CA_{best}, which is 84 mutations away from CA_0.

Intrigued by the fitness landscape of CA_{95}, we ran through the 13,305 rule sets in the neutral DFS tree and tested each rule set on 100 binomial randomly generated ICs. The 13,305 rule sets had an average performance of 60.48% in classifying of the ICs correctly, whereas our original rule set CA_0 classified 67% correctly. Though the majority of the network did not perform as well as CA_0, there were many rule sets that performed better, including CA_{best} which classified 77% of the ICs correctly. The fitness landscape of CA_{best} can be seen in Figure 10. As opposed to the original fitness function, CA_{best} has significantly better fitness up to and including $\rho_0 = .503$, but drops heavily in fitness directly after hitting that point. This is a very interesting result, as one would expect $\rho_0 \approx 0.5$ would be the absolute hardest IC to classify, yet this rule only has trouble

around $\rho_0 \in (0.51, 0.6)$. In finding CA_{best}, we have another example of a CA in the neutral network that performs better against an (albeit marginally) different environment. This is another indication that the neutral network is robust against environmental changes. An example of the running CA_{best} is given in Figure 11; this is a case where the evaluation was correct.

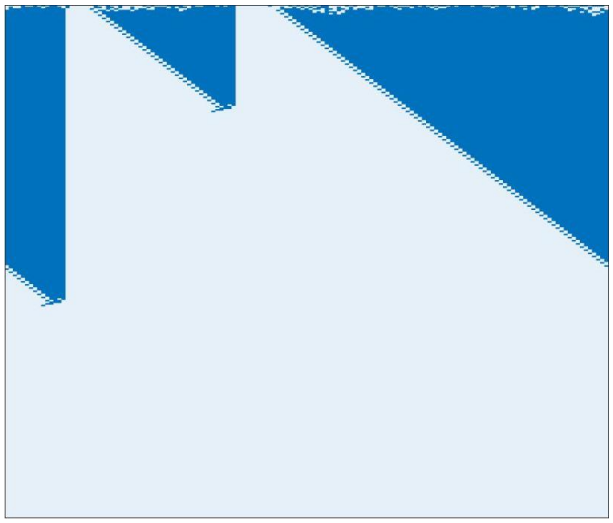

Fig. 11. An instance of CA_{best} running on an IC of 61/149 ones.

V. Methods

Since the CA considers 7 cells for each update, each CA rule set is 128 bits long[3]. For our purposes, the rule merely works as a lookup table, where the index of a particular cell's next value is given by the integer representation of its neighbors The GA had an initial population of 100 randomly generated rule sets. These initial conditions were generated such that the proportion λ of situations generating a 1 was uniformly distributed among all initial rules. Each rule set was tested on 100 randomly generated initial conditions, which were each 149 bits long. Again, to replicate Mitchell's results, the initial conditions were selected such that ρ_0 was uniformly distributed with half of the initial conditions above and below $\rho_0 = 0.5$. Fitness was calculated for each rule set in the population by taking the fraction of correct answers that were generated from the 100 ICs over the course of a poisson distributed number of steps with mean 320[3]. The number of steps was tied to each IC in order to preserve the required speed across all rules. At each step in the GA, the next generation was chosen using elitism and crossover of rules randomly chosen from the elites, producing 2 opposing children per crossover. Notably, our crossover was performed with a mixture of one- and two-point crossovers, with a 50/50 chance of either type of crossover occurring. In any case, the two children were subjected to a fixed number of random mutations, which was set to 2 for all results given in this paper. We tried raising the number of mutations to 5 or even 15, which sped up the onset of Epoch 2 but produced worse long-term results.

All of the figures and results were generated using Matlab. We graphed the fitness landscape of different rules by separating the total lattice size into a certain number of buckets, and ran the code on IC's that were forced to have a particular number of 1's to determine its fitness in a given place. In Figure 4, we used 25 buckets, while in all other figures we used the full 149 buckets as they were less intensive to find. The genetic trajectories were all auto-generated by our genetic algorithm code using the most fit and the average fitness of all rules for each generation, respectively. The space-time graphs of the running CA's were produced as heatmaps of the historical data produced by our running cellular automata, and an example of how to do this is shown in the published code (ga.m).

In generating the neutral network, we tested each mutation on the same 100 ICs. Out of the 100 trials, if the mutated rule set had at most 3 fewer correct runs than the original rule set, we considered it a "neutral" rule, and added it to the neutral network. All nodes in the neutral network had similar (at least 3% less) or better fitness than the original rule set. Our hope was that this slightly less strict interpretation of neutrality would offset the inherent randomness of using 100 of the 2^{149} possible ICs, so that the rules are probably, on average, the same.

Finally, for the study on the breadth and depth of the neutral network, we merely performed different algorithms by which we generated the neutral network. In our results for part IV, we performed a BFS on all possible mutations of the base rule, all possible mutations of those successful mutations, etc, stopping when we reached a depth of 3 nodes. The results for part V were similarly generated, but we performed a depth first search of successful mutants instead. For sake of time we could not perform a full search, but we still evaluated the fitness of all found rules on the harsher binomially generated ICs rather than the uniform ICs the original function and its network were found for.

VI. Discussion

In this paper, we successfully evolved a CA to perform a 1D density classification. In creating a neutral network based on our evolved rule set CA_0, we were able to get a sense of the size of the neutral network. 70% of rule sets that were 1 mutation away were included in the neutral network. We also discovered that the neutral network could be 95 mutations deep, meaning that rule sets up to 75% different from CA_0 still have high fitness in performing the density classification problem. Since the neutral network is so large, this creates a system that is robust against environmental changes, as each rule set performs better against different environmental factors, presented in different ICs. By randomly selecting rule sets from the neutral network to try with our new fitness function f', we found the initial population of rule sets performed well against a new fitness function. This also contributed to our conclusion that the large neutral network is robust against environmental changes.

It is interesting to note that the size of the neutral network in this case was enormous despite the seeming inflexible

nature of the problem. In all cases, the broad shape of the CA's appears to be the same, with the initial few steps determining completely what the rest of the run will do in each case. This might be a case in which the only necessary rules are those that maintain the long-term shapes that travel across the lattice, and all of the others are merely incidental and only probably resolve correctly. This might explain why the results for the 50% classification are actually worse than a 50/50 guess, while also explaining why such heavily mutated rules were found in the neutral network.

The fact that a rule was found in the neutral network that was seemingly superior in the harder parts of the classification task may show that the GA was failing to explore the entire neutral network of the best solution. It is important to remember, however, that the found rule was judging not based off of uniformly distributed, but binomially distributed ICs. While it is not as different as the f' rule introduced in section III, the different distribution of ICs is tantamount to having a different fitness function, and results evolved for one are not necessarily suited for another. Thus it was that the fitness landscape of CA_{best} was shorter, but broader than that of CA_0; it worked better on the harder cases, but it sacrificed its ability to perform on easier cases.

REFERENCES

[1] Wagner, A. (2012). The role of robustness in phenotypic adaptation and innovation. [online] https://royalsocietypublishing.org. Available at: https://doi.org/10.1098/rspb.2011.2293 [Accessed 1 Apr. 2019].

[2] M. Mitchel, Complexity: A Guided Tour. Oxford University Press, 2009.

[3] Mitchell, Melanie P. Crutchfield, James T. Hraber, Peter. (1994). Evolving cellular automata to perform computations: Mechanisms and impediments. Physica D: Nonlinear Phenomena. 75. 361-391. 10.1016/0167-2789(94)90293-3.

VII. CONTRIBUTIONS

Thomas wrote the code to implement the genetic algorithm and search the neutral network, with help from Carolyn. Nitin and Carolyn wrote and formatted the paper with editing and input from Thomas.