|  |  |
| --- | --- |
| **Larissa Anderson** | **Nialls Chavez** |
| **larissaa@unm.edu** | **ncchavez@unm.edu** |

**Abstract**

Simple mutation in the form of single point crossover and mutation in a genetic algorithm allows for the discovery of well-performing and broadly applicable cellular automata rules with the exploration of only a small subset of the entire cellular automata rule space. Similar to the Gaks-Kurdyumov-Levin rule high performance fitness for the density classification task is seen for certain cellular automata rules of radius 3.

**Author Contributions**

**Larissa Anderson**

* Created plan for CA/GA and subsequent analysis
* Created initial turn-in document
* Complete write-up
* Figures 3-7

**Nialls Chavez**

* Created Python code for CA and GA
* Figures 1,2,8-10

**Outside Sources**

* Documents provided by Professor Melanie Moses.
* Mitchell et al. 1994:  Evolving cellular automata to perform computations: mechanisms and impediments
* Mitchell, M. 1996. An Introduction to Genetic Algorithms. Chapters 1-2.

**1. Introduction**

Cellular automata can be used to simulate real-world systems and perform various tasks. However, predicting the final behavior of systems to which a cellular automata is applied is not necessarily intuitive. In this case we are trying to discover cellular automata rules that can be broadly applied to correctly identify the density of various bit strings. In order to optimize the search for a cellular automata rule that is excellent at density characterization over a range of bit string configurations we have used a genetic algorithm with both crossover and mutation. We predict that using a genetic algorithm will be more efficient than random at locating high-performing rules of similar ability in density classification as the Gaks-Kurdyumov-Levin (GKL) rule.

**2. Results**

***2.1 Analyze your results***

Our results indicate that the genetic algorithm we have applied to our population of cellular automata (CA) rules does result in the creation of cellular automata rules that have a substantive ability to correctly classify the density of a bit string. The performance fitness of our CA rules improves considerably over our generations as we use a genetic algorithm to perform selection and mutation. Both the CA rules with radius 2 and radius 3 have initial maximum fitness values near 0.5. This maximum initial fitness of 0.5 is due to the presence of CA rules containing all 1s or all 0s which will classify all of the initial configuration strings to which it is applied correctly about half the time as approximately half of the initial configurations have a density of greater than 0.5 and half have a density of less than 0.5. For both the radius 2 and radius 3 CA rule sets we do not see the same four distinct “epochs of innovation” seen in the Mitchell et al. 1994 article. However, in the case of our radius 2 CA rules, as displayed in Figure 1, we see small gains above 0.5 within our first 10 generations. At 10 generations there is a marked increase in the maximum fitness in our CA population, and the maximum fitness fluctuates between approximately 0.83 and 0.93 for the next 40 generations. This fluctuation is due to slightly differing fitness with each new randomly generated initial configuration population. It is unclear if we do not see the four distinct epochs of innovation due to a decrease in generations in comparison to the Mitchell et al. 1994 paper or if our transition from epoch 2 to epoch 4 happened to be a smoother transition in this particular run. Our run with radius size 3, depicted in Figure 2, also illustrated an increase in maximum fitness over generations of our CA as it went through our genetic algorithm. The maximum fitness in our CA rule population increased more rapidly in our radius size 3 CA population than in our radius size 2 population. While again there did not to be four clear epochs there was a large increase in fitness near generation 3, from slightly above 0.5 to slightly above 0.8. The greatest fitness values then fluctuated around 0.8 until generation 15 when fitness values above 0.9 emerged. Overall, these runs indicate that our cellular automata with radius size 3 reached a higher performance fitness for our density classification task and reached these higher fitness values faster even with the implementation of the same genetic algorithm.

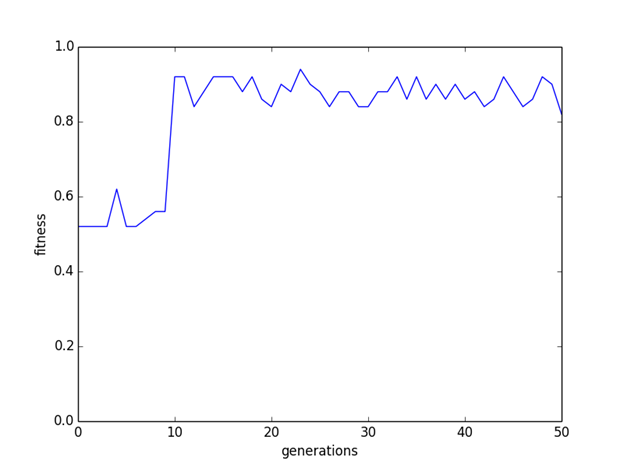


Figure 1: Best fitness per generation for one run for a cellular automata rule of radius two modified by a genetic algorithm.

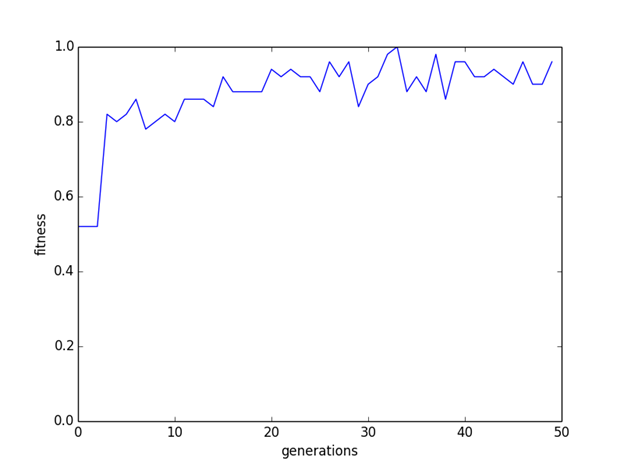
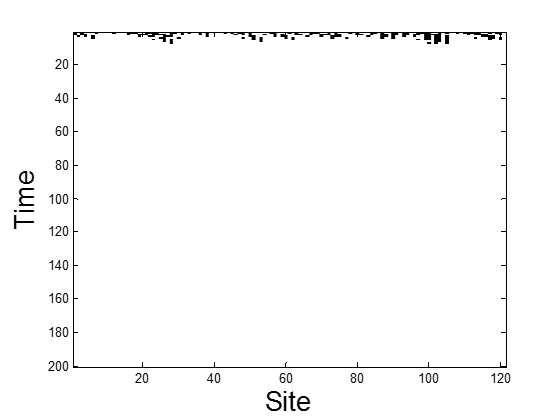
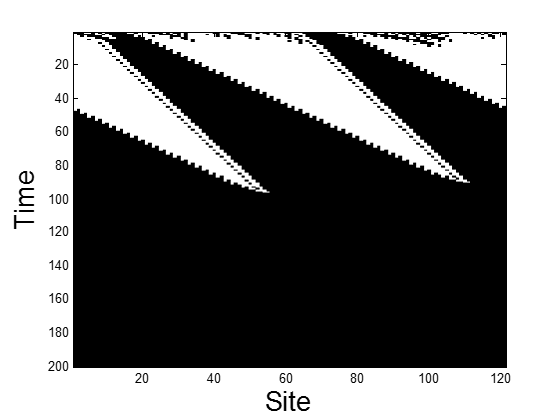


Figure 2:  Best fitness per generation for one run for a cellular automata rule of radius three modified by a genetic algorithm.

These highly fit strings are capable of correctly classifying bit strings with densities near 0.5 as is shown in Figure 3. However, the ability to correctly classify strings with densities extremely near 0.5 is difficult for both our CA rule and the GKL rule mentioned in the Mitchell et al. 1994 paper. The proportion of correctly classified strings by their initial configuration density for both the GKL rule and our highest fitness performing string, with a fitness value of 0.98 are showing in Figure 4 and Figure 5.



a)



   b)

Figure 3: Space-time diagrams for a rule of radius 2 and performance fitness of 0. 98 for initial configurations with densities 0.45455and 0.55372 for a) and b) respectively.

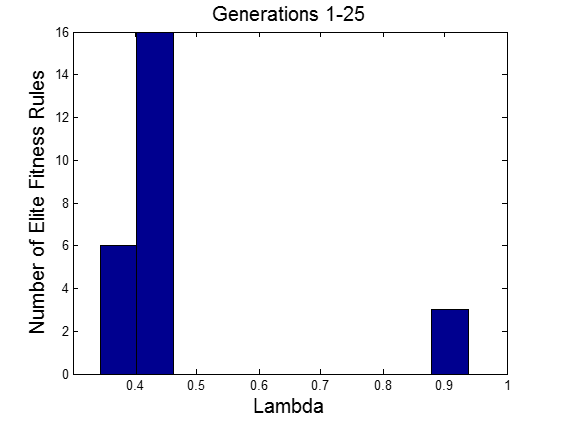


Figure 4: The average performance of the GKL rule for the density classification task over three runs for an initial configuration population of 50.



Figure 5:  The average performance of our best performing rule for the density classification task over three runs for an initial configuration population of 50.

Similar to the lambda values seen for elite rules in the Mitchell et al. 1994 our maximum fitness per generation also tends to have densities near 0.5 as seen in Figures 6 and 7. As our genetic algorithm progressed we saw our CA rule densities cluster nearer to 0.5, as the outliers in either low or high density values were not present in our generations 25-50 for either our radius size 2 or radius size 3 cellular automata rules.



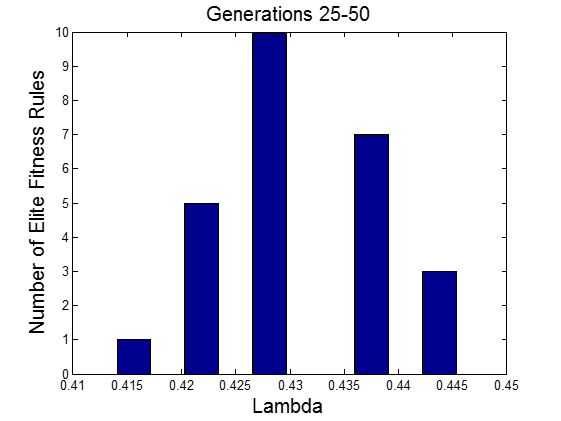


Figure 6: A histogram of the lambda (density) of the elite rules for every set of 5 generations for our cellular automata rules of radius 2.

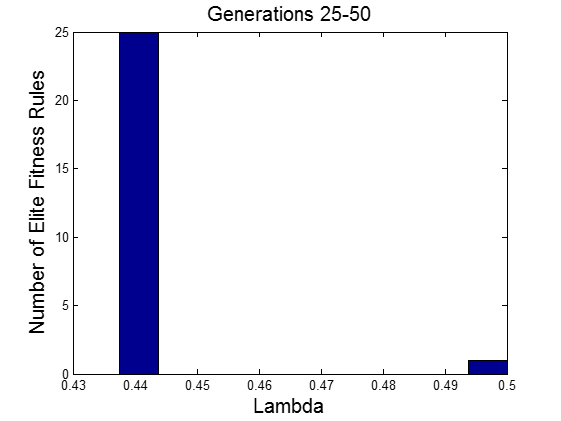
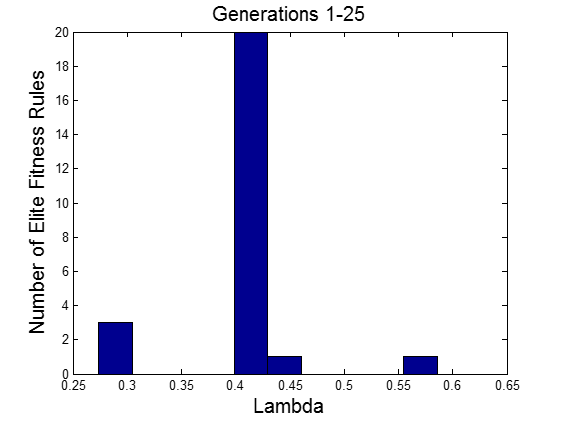


Figure 7:  A histogram of the lambda (density) of the elite rules for every set of 25 generations for our cellular automata rules of radius 3.

***2.2 Analyze mutational robustness***

In these simulations we only explore a minute fraction of the rule space. With the generational limitations imposed, even without any elitism in our genetic algorithm we are exploring a maximum of 100 new CA rules per generation, and with a maximum of 50 generations only 5000 rules in total. However, we employ 20 percent elitism within our genetic algorithm, therefore we have 100 new rule sets for the first generation but a maximum of 80 new rule sets for every generation thereafter, which gives us a total of 4020 CA rule sets. These maximums assume that every evaluated rule set is unique. The total rule space is 232 for our radius of 2 rule sets and 2128 for our radius 3 rule sets, therefore the region we explore is at maximum 1.1642e-06 for radius 2 rule sets and 1.4694e-35 for our radius 3 rule sets.

The mutational robustness of CA rules adept at our density classification task was explored with the modification of one our CA rules with the highest performance fitness. The modified rule had a performance fitness of 0.94 when applied to a set of 50 bit strings of length 121 with a uniform density distribution. The mean and maximum performance fitness of CA rules between 1 and 5 mutations away from this highly fit rule is shown in Figure 8 and Figure 9. This particular rule appears to be somewhat mutationally robust with our population one single point mutation away having both mean and maximum fitness of over 0.9. This mean fitness as shown in Figure 8 does decrease as we increase our mutational distance more than one mutation away from our initial fit string.  However, there are still fit rules at some mutational distance away, with a maximum fitness of greater than our original 0.94 found at two mutations from our original CA rule, as shown in Figure 9. Higher fitness values within this short mutational distance could be possible if we sampled a larger proportion of the rules present within this mutational distance. However, this does appear to show that the fitness landscape can be traversed, in this case a fairly large hamming distance representative of 5 percent of the rule length without a substantial loss in fitness.

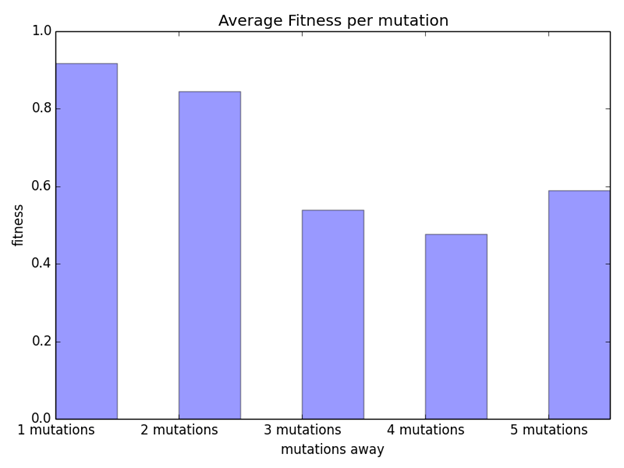


Figure 8: The mean fitness of rules between 1 and 5 mutations away from our initial rule with performance fitness of 0.94 over one run of 50 initial conditions.

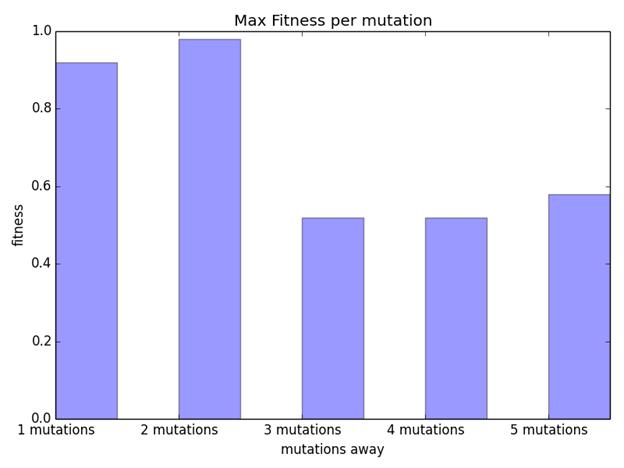


Figure 9: The maximum fitness of rules between 1 and 5 mutations away from our initial rule with performance fitness of 0.94 over one run of 50 initial conditions.

***2.3 Improve the performance of your density classification CA***

In order to improve the performance of our density classification CA we modified our genetic algorithm to further mutate CA rules we had previously evaluated. This method reduced redundancy in our evaluation of cellular automata and allowed us to explore more of the cellular automata rule space than in our previous strategy. In this case it more quickly converged to fitness levels above 0.6, effectively eliminating the first epoch that in our previous runs oscillated around fitness values of 0.5. However, after the fitness values reached approximately 0.8 it did not improve substantially again. It is unclear if this is due to the number of generations, if it would have eventually reached fitness values of above 0.9 given more time, or if it is the result of this particular run.

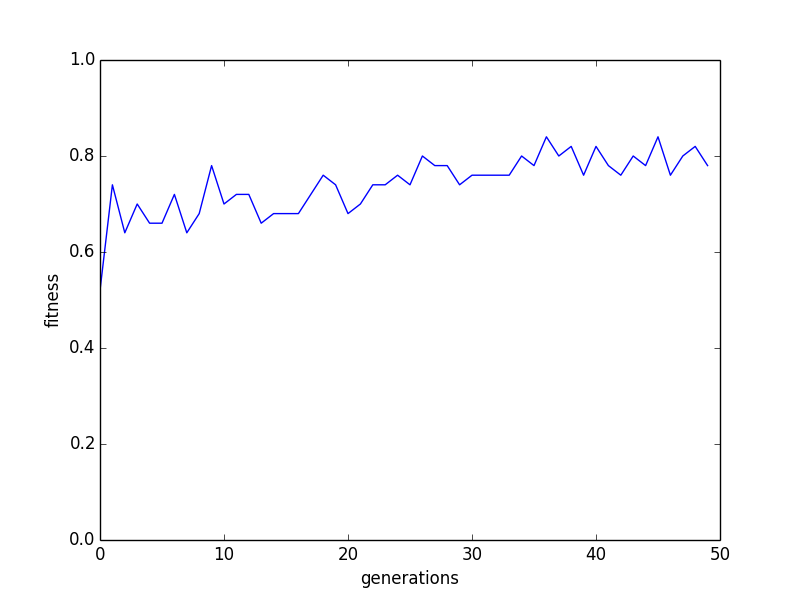


Figure 10:  Best fitness per generation for one run for a cellular automata rule of radius two modified by a genetic algorithm and including a reference step to improve our density classification performance.

**3. Discussion**

***3.1 CA and GA***

Our results for both our radius 2 and radius 3 CA rules show a large increase in fitness for density classification over less than 20 generations. Despite the relatively small exploration of the overall rule space significant gains were made to identify a suite of rules that may be broadly applicable. As is mentioned in the Mitchell et al. 1994 article density classification becomes more difficult in cases where the initial density is near the 0.5 breakpoint as seen in Figures 4 & 5. While classification becomes difficult directly near that breakpoint it can still be accurate relatively close to 0.5 as seen for one of our highly fit rules in Figure 3. As the density classification task we are evaluating is trying to determine if a bit string has a density of more or less than 0.5 we saw densities of highly fit rules to have a density of 0.5 as was found in the Mitchell et al. 1994 paper. Overall our results were in concordance with the findings from the Mitchell et al. 1994 paper and it validates the application of a genetic algorithm to find cellular automata that are proficient in certain tasks.

***3.2 CA Mutational Robustness***

The results for our mutational robustness show that there are regions of comparable fitness immediately surrounding our highly fit rule. The most compelling part of our mutational robustness results were the identification of a rule of higher fitness two mutations away from our highly fit rule. It is indicative of the presence of other extremely fit rules within the immediate region of our initial fit rule.

After several mutations both the maximum and mean fitness values drop substantially. However, as these runs represent only a small proportion of the rules within a mutational distance of 5 this trend may not hold under an exhaustive exploration of all possible CA rules within this distance. Based on our histograms of rule density versus maximum fitness we would expect there to be highly performing rules with densities near 0.5 that may be more than 5 mutations away from this particular rule. Furthermore, while this rule is not particularly mutationally robust this may not be true for all of our rules. There could be peaks in our fitness landscape represented by our highly fit rules surrounded by a region that is of greater fitness than is the case for this particular rule.

***3.3 Density Classification CA Improvement***

The method to reduce redundancy in CA rule evaluation was applied to our CA with radius size 2 as the rule string is shorter we expected a higher probability of redundancy than for our CA rules radius size of 3, as the lengths of the rule sets are 32 and 128 respectively. The application of this particular strategy did not show a significant increase in performance of our CA density classification task. This may be due to the small proportion of the rule space we are evaluating in these limited runs. However this approach may become more useful over more extensive exploration of the rule space. As we explored so little of the CA rule space redundancy in our CA rules was revealed to be somewhat negligible. However, in a more extensive application of CA rules with mutation by a genetic algorithm the redundancy may prove to be a larger issue. This could be particularly true if the CA rule is evaluated over a larger population of initial configurations, representing a larger computational investment in that CA rule.

 However, despite the plateau in CA fitness it did show a quicker convergence to moderate CA fitness levels, possibly due to the elimination of some small amount of redundancy. As the number of mutations and crossovers should not have been reduced and the procedure was otherwise identical it is unlikely that the reduction in final fitness values is due to our modification and is instead an artifact of this particular run. In order to address this inconsistency in maximum fitness it would be useful to run the genetic algorithm for more than 50 generations however due to the limitations of this project we did not pursue this avenue of analysis.

**4. Methods**

***4.1 GA Parameters***

Similar to the methods described in Mitchell et al. 1994, we use a strategy that employs both single point crossover and single point mutation in our genetic algorithm. Initially, cellular automata was run, for a population of 100 CA rules each applied to an initial configuration population consisting of 50 bit strings of length 121 for 200 generations, the fitness of each CA rule is calculated. Fitness of the rule is determined by calculating the density of the original bit string to which it is applied and then evaluating if the CA rule accurately classifies its density as greater or less than 0.5. If the initial configuration has a density of less than 0.5 and after the CA rule has been applied, either for 200 generations or until a static configuration has been reached, the final configuration should be all 0s. For an initial configuration density of more than 0.5 the final configuration should be all 1s. If either of these conditions are exactly met that configuration is given a fitness value of 1 and the average of these configuration fitness values is the assigned performance fitness of the rule that was applied to the initial configuration. The 100 CA rules are then ranked by these performance fitness values and the top 20 percent of these rules are put into the next CA rule population unchanged. The remaining 80 percent are paired, a random pivot point is assigned and they switch the portions of the bit string on either side of the pivot point. After this crossover has occurred the rules are then mutated. This is a single point mutation where a random bit is chosen and flipped. These 80 new rules are then combined with the elite 20 unchanged rule and run as the new rule set for the cellular automata.

In order to improve our genetic algorithm we compared our 80 new rule sets, obtained after single point crossover and single point mutation, to previous mutated rule sets. If the newest rule set matched a rule set that had already been evaluated we then mutated it by randomly flipping another bit to maximize the diversity of rule set that we were exploring. The rule was mutated a maximum of three new times before it was placed in the new generation for evaluation through application to a new set of initial configurations.

***4.2 Initial Configurations***

The first generation of CA rule sets consisted of a population of 100 bit strings of length 32 or 128 biased with uniform density between 0 and 1. The length of the rule depended on the radius size the CA was using, either 2 or 3. The initial configurations to which the population of CA rule sets were applied were newly created for each generation of the genetic algorithm. The initial configurations consisted of a population of 50 bit strings of length 121, the population had a uniform density of between 0 and 1 and always included at least one bit string containing all 0s and one containing all 1s.

***4.3 Data Analysis***

All runs of the cellular automata and the genetic algorithm applied to the CA rule set were performed in Python version 3.4.3. Figures representing the resulting data were created in Python or the data was exported as a comma delimited file and figures were created in Matlab version R2014a. Our comparison of efficiency of the genetic algorithm, the differences between the CA of radius 2 and radius 3, and the mutational robustness of a fit CA rule were based off of the trends displayed in these figures. No formal statistics were used to compare the data presented in our figures or to inform our results and discussion.

**References:**

Mitchell, M. 1996. An Introduction to Genetic Algorithms. Chapters 1-2. MIT Press, Cambridge, Massachusetts London, England.

Mitchell, M., J. P. Crutchfield, and P. T. Hraber. 1994. Evolving cellular automata to perform computations: mechanisms and impediments. Physica D: Nonlinear Phenomena 75:361–391.