Generating Interesting Patterns in Conway's Game of Life Through a Genetic Algorithm

Hector Alfaro
University of Central Florida
Orlando, FL
hector@hectorsector.com

Francisco Mendoza
University of Central Florida
Orlando, FL
mendozadiaz@msn.com

Chris Tice
University of Central Florida
Orlando, FL
ticetrio@yahoo.com

ABSTRACT

In this paper we describe the application of a genetic algorithm to John Conway's Game of *Life*, a popular form of Cellular Automata. Our intent is to create an arrangement of cellular automata (CA) that will produce "interesting" behavior when placed into Conway's Game of *Life* – where "interesting" behavior includes repeated patterns, reproducing groups of cells, and patterns that change their location in *Life*. We also discuss techniques for improving algorithm performance by applying methods developed in previous research.

1. INTRODUCTION

A Cellular Automaton is an individual within a grid of colored cells which evolve according to basic rules relating which neighboring cells change states. A form of Cellular Automata was first introduced by John von Neumann in the early fifties and incorporated into his "universal constructor." [7] John Horton Conway, a mathematician of the University of Cambridge, introduced a solitaire game known simply as *Life* in early 1970. In *Life*, cells in a two-dimensional space are born, die, or remain alive from one generation to the next according to three simple rules:

1. Survivals. Each live cell with two or three neighboring cells survives for the next generation.
2. Deaths. Each live with four or more neighbors dies (is removed) from overcrowding. Every cell with one or no neighbors dies from isolation.
3. Births. Each dead or empty cell adjacent to exactly three neighbors–no more, no less–comes to life. [7]

Figure 1 shows the rules of *Life* being applied: (a) shows the indicated cell coming to life, (b) shows a surviving cell, and (c) shows a dying cell.

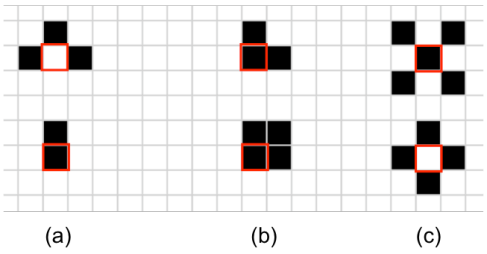

Figure 1: The rules of *Life*.

Many versions of Conway's Game of *Life* exist on the internet and throughout the research community that replicate his hand modeled version letting the user color tiles to test differing systems and their reactions. Examples of initial tile configurations were evolved through time and given names such as gliders, exploders, and tumblers due to their behavior throughout the course of *Life*. For example, gliders are so named because this particular configuration of cells glides through the game board as the rules of *Life* are applied.

2. GOALS

It is our goal to evolve players that exhibit "interesting" behavior for the Game of *Life* by using a Genetic Algorithm (GA). We consider a player to be a collection of cells in an n-by-n sized grid. For our experiments, we set n=10. We say that a group of cells exhibits "interesting" behavior when that group of cells meets one or more of the following prerequisites:

1. Live cells remain alive when the rules of *Life* are applied,
2. Dead cells come to *Life* when the rules of *Life* are applied,
3. A specific pattern of cells remains in that configuration when the rules of *Life* are applied and remains stationary, or
4. A specific pattern of cells remains in that configuration when the rules of *Life* are applied but changes its location in the game board.

We also use our platform to test some of techniques attempted by others in the GA research community on our

domain. Some of the techniques we tried are: triggered hypermutation, random immigrants, and n-way elitism.

3. MOTIVATION

The Game of *Life* can been used to model complex natural systems due to its ability to allow cells to reproduce and die. Their utility is due to their implementation of a form of cellular automata (CAs). The simplicity of CAs tend to mirror simple organisms that exhibit fascinating behavior when part of a group, such as bees in a beehive. Conway's Game of *Life* further increases the number of possible behaviors and allows us to study CAs in a two-dimensional environment. Our ability to model such complex natural systems would contribute to our understanding of said systems. It may seem irrelevant to understand ant colonies or beehives, but our understanding of complex systems would play a key role in understanding some of the more dynamic and impactful complex systems that take part in our daily life. Such complex systems include: the stock market, the human brain, groups of pedestrians.

Sarkur et. al. [6] used CAs to form a receiver and transmitter units in a mobile device to save space and improve reliability thereby reducing cost. The receiver and transmitter units were previously separate to improve security. Sarkur et. al. used a pseudo-noise generator to create a secure field for the mobile device using a feedback shift register which is itself a one-dimensional CA. This is another example of CA's many applications.

Some of the motivation for this research arises from findings in the research community of interesting techniques applied to GAs. However, some of the strongest criticism of these techniques is that they were conceived and tried under extremely specialized conditions. Our research aims to explore these techniques' effectiveness on a different domain.

Specifically, we propose the following hypotheses:

1. We will be able to find players for which the rules of *Life* will create interesting behavior.
2. We expect triggered hypermutation to improve GA performance due to its ability to introduce diversity, as discussed in [4].
3. We expect random immigrants to degrade overall GA performance, as has been shown in [4].

We would like to find answers to the following questions:

1. What is the best way to achieve the interesting behavior we desire?
2. How does random immigrants affect the population's performance? Can we turn the results of [4] around by changing the replacement rate?
3. Can we use triggered hypermutation to increase algorithm performance?

4. RELATED WORK

4.1 Complex Systems

As discussed above, *Life* may be used to model the behavior of complex systems. Complex system [13, 11, 5] exhibit some common traits. Primarily, complex systems are non-linear systems. As such, the behavior of the system cannot be understood by simply breaking down each individual component of the system. These systems often exhibit emergent behavior as a result of the interactions of its components. The components of a complex system are often really simple agents working under a simple set of rules. A subset of complex systems is adaptive systems. Adaptive systems exhibit many desirable features but primarily the ability to adapt to unforeseen changes to their environment.

An example of a complex adaptive system is a model of human segregation [13]. Segregation is not clearly understood by researchers due to the fact that this behavior is non-linear; we cannot simply look at one of the neighborhoods in the United States, for example, and try to explain segregation based on some of its characteristics. Schelling developed a mathematical model in [13], similar to Conway's Game of *Life*, to attempt to devise models of segregation and indeed found ways to explain their complex behavior. Other examples of complex adaptive systems include temperature regulation in bumblebee colonies [11] and the structures of complex hierarchical networks [5]. The main characteristics are always present: emergent behavior, adapting to unforeseen change, many interacting parts, simple individual behavior.

The understanding of complex adaptive systems allow us to explain behavior not currently understood in nature and, ultimately, ourselves. We draw our motivation from this awesome power. We hope our model of *Life* may be an early component of more powerful models that can more accurately explain complex behaviors.

4.2 Using a GA on CAs and the Game of Life

GAs have been used before to evolve CAs [10, 2, 1] successfully. They've been rarely used to evolve players for the Game of *Life*. Kazakov and Sweet [10] were able to successfully design a GA to evolve players for the Game of *Life* by making use of Wolfram's entropy equations [14]. Kazakov and Sweet worked under the assumption that a cell's entropy is directly related to that cell's potential to exhibit "interesting" behavior. They also divided their focus into finding a set of rules that provided the most interesting solution. Our research differs by using Conway's original rules for *Life*. We found that the research in [10] focused largely on the implementation of the entropy equations. Calculating the entropy of a neighborhood of cells is a computationally extensive task. As a result, Kazakov and Sweet had to estimate a cell's entropy. We bypass these issues by not making use of the entropy equations.

4.3 Evolutionary Strategies

As stated in [12] and [8] elitism assures that the best individuals in the population always survives from one generation to the next. Without elitism, it is possible that the best structure may disappear if it is not selected for reproduction or if it is destroyed by crossover or mutation. It has

Table 1: Rewards provided by our fitness function.

Condition	Reward
Live cell that remains alive	+1
Dead cell that comes to *Life*	+1
Stationary cell pattern	+250
Moving cell pattern	+500

been consistently shown [12] that elitism greatly improves the performance of a GA. Taking into account this and what was stated before we decided to use elitism in the hopes of improving the performance of our GA.

Grefenstette used a replacement strategy called random immigrants for the promotion of continuous exploration of a GA's search space [9]. This replacement strategy would replace a percentage of the GAs population (determined by the replacement rate) every generation with new randomly generated individuals. This work was then extended in [4] by Grefenstette and Cobb, where they showed that random immigrants helps a GA when there are changes in the environment, but has a negative effect on stationary environments. We thought it would be interesting to see the effect random immigrants would have in our study by manipulating the replacement rate. As such we have implemented random immigrants as described in [4] and [9] into our GA.

In [3] Cobb developed an "adaptive mutation-based" mechanism, called triggered hypermutation with the purpose of increasing the adaptability of a GA in changing environments. Triggered hypermutation works by temporarily increasing the mutation rate of a GA to a high value (determined by the hypermutation rate) when a degradation of the time-averaged best performance of the population is detected [3, 4]. It was shown [3, 4] that triggered hypermutation does help for cases where there are changing environments. However, the magnitude of improvement seen depends on how the environment is changing and the value of the hypermutation rate [4]. It was also shown in [4] that hypermutation seems to have little to no effect on stationary environments as it would seem that hypermutation is rarely ever triggered. We would like to point out that in [4] elitism was not used. For our experiment, hypermutation was used in conjunction with elitism. We expect the use of elitism to cause hypermutation to be triggered.

5. EXPERIMENTAL METHODOLOGY

5.1 Fitness Function

We devised a fitness function capable of rewarding behavior we wanted to see, primarily, the behavior we discussed in section 2. Table 1 summarizes our reward system.

There are a few steps required to obtain some of the values discussed in the table above, they are:

1. Run the game of *Life* on the given individual 20 times, for each run
   (a) Determine each 3x3 neighborhood's pattern of live and dead cells.
      i. If the given pattern has not been seen before
         A. Add it to the stack of patterns that have now been seen, as well as the location the pattern was observed in.
      ii. If the given pattern has been seen before
         A. If the pattern has been seen before in the same location, add 250 to the individual's fitness
         B. If the pattern has been seen before in a different location, add 500 to the individual's fitness.
   (b) Apply the rules of *Life* to every cell in the game.
      i. If a cell was alive in the current generation, and remains alive for the next generation, add 1 to the individual's fitness.
      ii. If a cell was dead in the current generation, and comes to life in the next generation, add 1 to the individual's fitness.

Table 2: Parameter settings.

Parameter	Value
Population size	400
Selection method	Proportional select
Fitness scaling	Maximization
Crossover type	Uniform
Crossover rate	1.0
Mutation type	Bit Flip
Mutation rate	0.005
Genes in a Chromosome	10
Gene size	10
Elitism	Used
Random Immigrants	Varied
Immigrant Replacement Rate	Varied between 0.01-0.10
Hypermutation	Varied
Hypermutation scaling factor	Varied between 10-50
Hypermutation Threshold	1 digit difference

5.2 Experimental Setup

Since we wanted to answer quite a few number of questions, we decided to first run our GA without any special features, like elitism, triggered hypermutation, and random immigrants. Once we were aware of the kind of solutions that the algorithm was providing without these special features, we used elitism, as we were sure it would improve algorithm performance. Finally, we experimented with triggered hypermutation and random immigrants. They were tested independent of each other and their respective parameters varied. Finally, they were combined to explore any possible interactions.

The parameters in Table 2 were used for every one of our experiments. The genes in a chromosome and size of genes determine the length and width of the neighborhood that the rules of *Life* are applied to. Various neighborhood sizes were tested and various combinations of gene-and-chromosome sizes were tested. A 10×10 neighborhood provided the best performance without great need for large computational power. The solution space in a 10×10 grid is still large, providing 2^{100} possible solutions, or about 1.27×10^{30}. Our algorithm is flexible enough to increase the neighborhood

size as desired; however, the GA takes a long time to evaluate the fitness of larger neighborhood sizes. The immigrant replacement rate determines the percentage of the population that gets replaced when random immigrants is enabled. The hypermutation scaling factor determines the magnitude of hypermutation. Essentially, it is used in the following calculation:

$$m_h = m \times m_s$$

Where m_h = mutation when hypermutation is triggered, m = the mutation rate when hypermutation is not triggered, and m_s is the hypermutation scaling factor.

The hypermutation threshold is another parameter pertaining to hypermutation; it determines when hypermutation should be triggered. For example, a value of 1 means that hypermutation will trigger when there is less than a 1-point difference in fitness variance from one generation to the next. Preliminary tests showed that a value of 1 was reasonable for the hypermutation threshold. In contrast to [3, 4], our implementation of triggered hypermutation does not allow it to be triggered again until 10 generations have passed after it is initially triggered. This is done to give the GA time to readjust and recover from a previous trigger before hypermutation is used again.

6. RESULTS

6.1 Triggered Hypermutation

We found that triggered hypermutation increases the GA's chance of finding the optimal solution. Figures 2 and 3 compare the algorithm's performance with and without triggered hypermutation.

As can be seen from Figure 2, the lack of hypermutation provides a decent point of convergence. However, Figure 3 shows that hypermutation provides an extra boost to the GA that allows it to find even better solutions. The individual with the best fitness value from the tests depicted in Figure 3 had a fitness of 4900, while the best individual from Figure 2 had a fitness of 4780.

We tweaked various parameters pertaining to hypermutation, specifically the hypermutation scaling factor. The scaling factor is responsible for deciding how strong of a mutation should be introduced when hypermutation is triggered.

Figures 4-7 show the effect of changing the hypermutation scaling factor. Small scaling factors, such as 5, don't produce enough mutations in the population to provide significantly better solutions. Alternatively, scaling factors that are too large, such as 50, disrupt the population and can cause candidate solutions to be lost due to a mutation.

6.2 Random Immigrants

Figure 8 shows the effect of introducing random immigrants. Random immigrants did not serve much of a purpose except to disrupt the population. Its presence never allowed hypermutation to be triggered because the introduction of random immigrants increased variance each generation, which

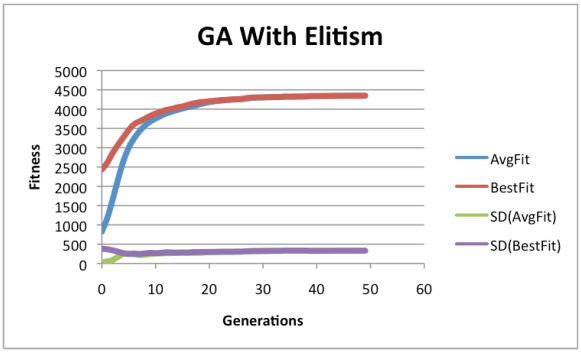

Figure 2: Hypermutation off. Note: only 50 generations are shown as the GA converged to a suboptimal solution past 50 generations.

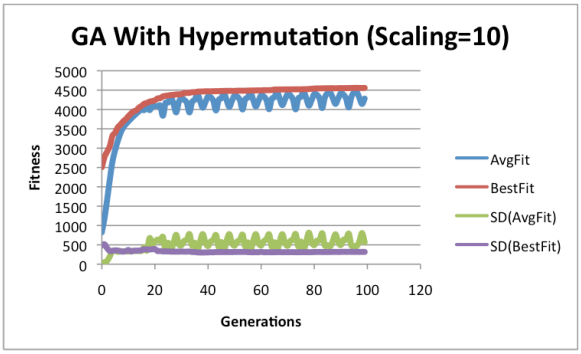

Figure 3: Hypermutation on. Hypermutation scaling = 10

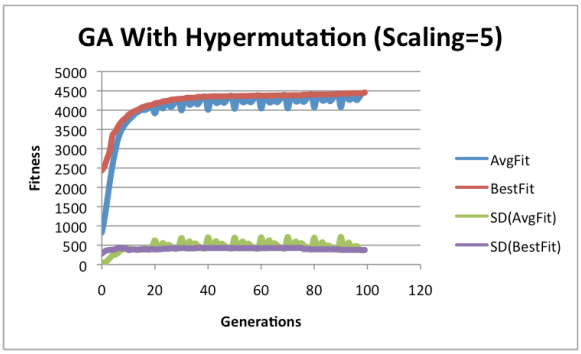

Figure 4: Hypermutation on. Hypermutation scaling = 5

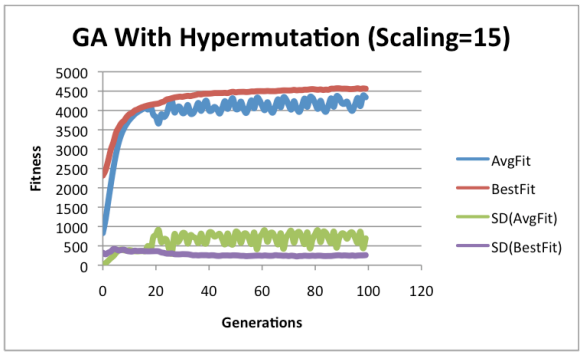

Figure 5: Hypermutation on. Hypermutation scaling = 15

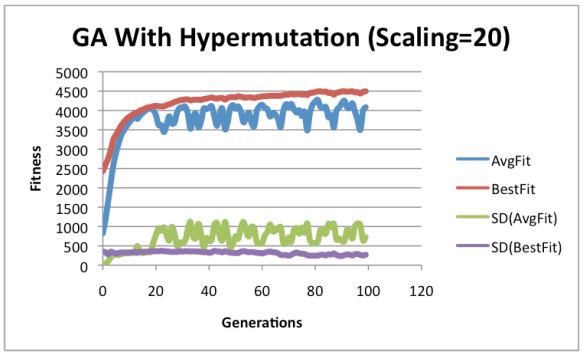

Figure 6: Hypermutation on. Hypermutation scaling = 20

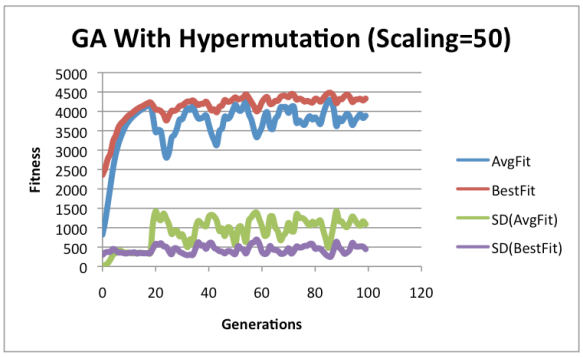

Figure 7: Hypermutation on. Hypermutation scaling = 50

meant the hypermutation threshold was never met. Neither adjusting the hypermutation threshold nor the immigrant replacement rate changed this behavior.

6.2.1 An unexpected finding

While crafting our fitness function, we encountered a bug that was changing the value of individuals amidst their fitness calculation. The bug was causing bizarre results and, although we were unable to explain it at first, we did notice that random immigrants had a positive effect on the GA's performance. Figure 9 shows the GA's performance on the faulty fitness function without random immigrants. The GA has trouble adjusting to the constantly changing landscape. When random immigrants was turned on, see Figure 10, the GA was a doing a much better job at keeping track of the changing landscape, regardless of how erratic the changes were. Triggered hypermutation, in contrast, was hurting performance whenever it was applied, as Figure 11 shows.

6.3 Interesting Behavior

Figures 12-14 show some of the patterns we saw as a result of the best solutions.

The starting configuration shown in Figure 12 was found without elitism, triggered hypermutation, or random immigrants. Thus, its fitness is low compared to that of 14. However, some interesting behavior can still be observed. The fitness function successfully rewarded repetitive patterns. Figure 12(a) shows the starting configuration that contains some repetitive patterns of cells. However, Figures 12(b) and 12(c) show how the fitness function rewarded this given configuration due to its repetitive patterns, some of which moved from their initial location. Despite its relatively low fitness value, the configuration from Figure 12 produced some interesting patterns, like the one on Figure 13. The last tile of Figure 13 is a stationary state; the rules of *Life* do not change the cell's configuration after this point.

Figure 14 shows a highly-fit individual found by using elitism and hypermutation. This individual received fitness boosts due to its highly symmetric structure. Recall that individuals that exhibited repetitive patterns were more strongly rewarded, and rewards for repetitive patterns that changed their location on the board were even stronger. In contrast to Figures 12 and 13, the individual from 14 does not reach a static configuration, instead it reaches an oscillating state. The configuration of cells displayed in the last row of 14 is commonly referred to as a Pulsar.

7. CONCLUSIONS AND CONTRIBUTIONS

7.1 Hypothesis #1: Interesting Life

The solutions found by our GA exhibited interesting behavior, as discussed in Section 6.3. We attribute this finding to the fact that we crafted our fitness function to reward interactions that results in interesting behavior, namely:

- we rewarded cells that either remained alive or came to life,
- we rewarded configurations that were repeated throughout the board or across multiple generations of *Life*

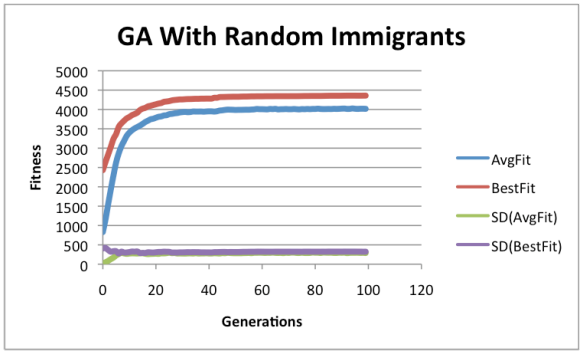

Figure 8: Hypermutation and random immigrans

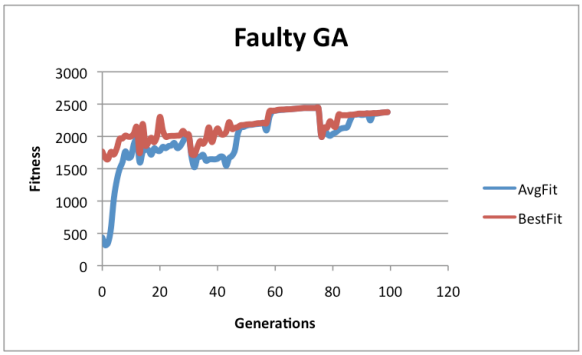

Figure 9: No random immigrants on faulty fitness function.

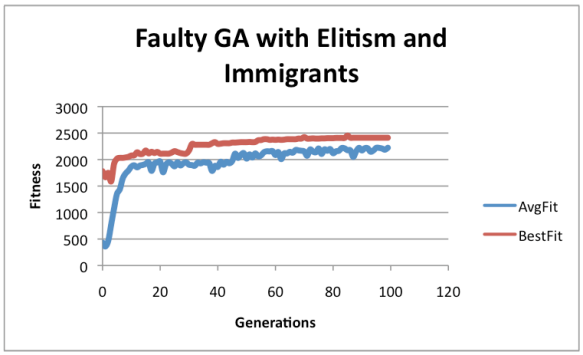

Figure 10: Random immigrants on faulty fitness function.

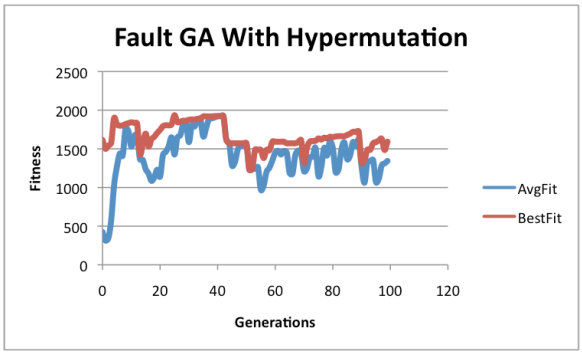

Figure 11: Triggered hypermutation on faulty fitness function.

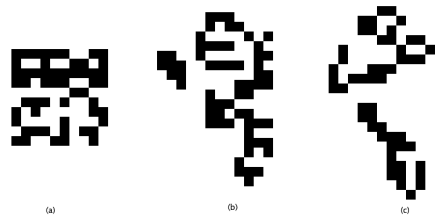

Figure 12: A configuration with a fitness of 4460. (a) Starting configuration; (b) and (c) progression once rules of *Life* are applied.

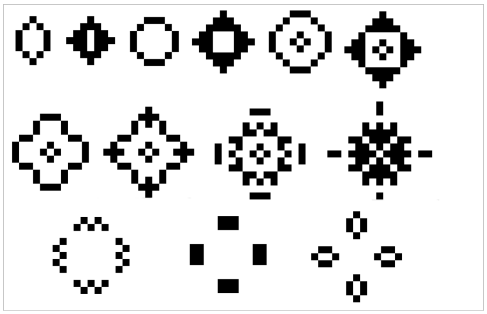

Figure 13: Isolated portion of configuration shown in Figure 12 after about 20 generations.

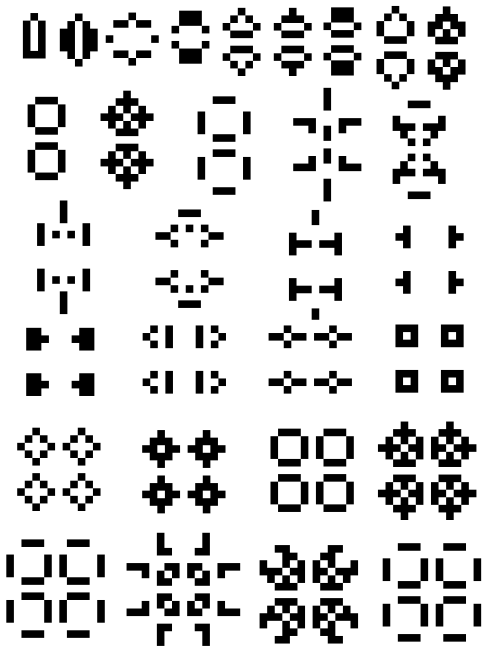

Figure 14: Resulting pattern found by a configuration with fitness of 4900. The configuration in the last row oscillates infinitely as the rules of *Life* are applied.

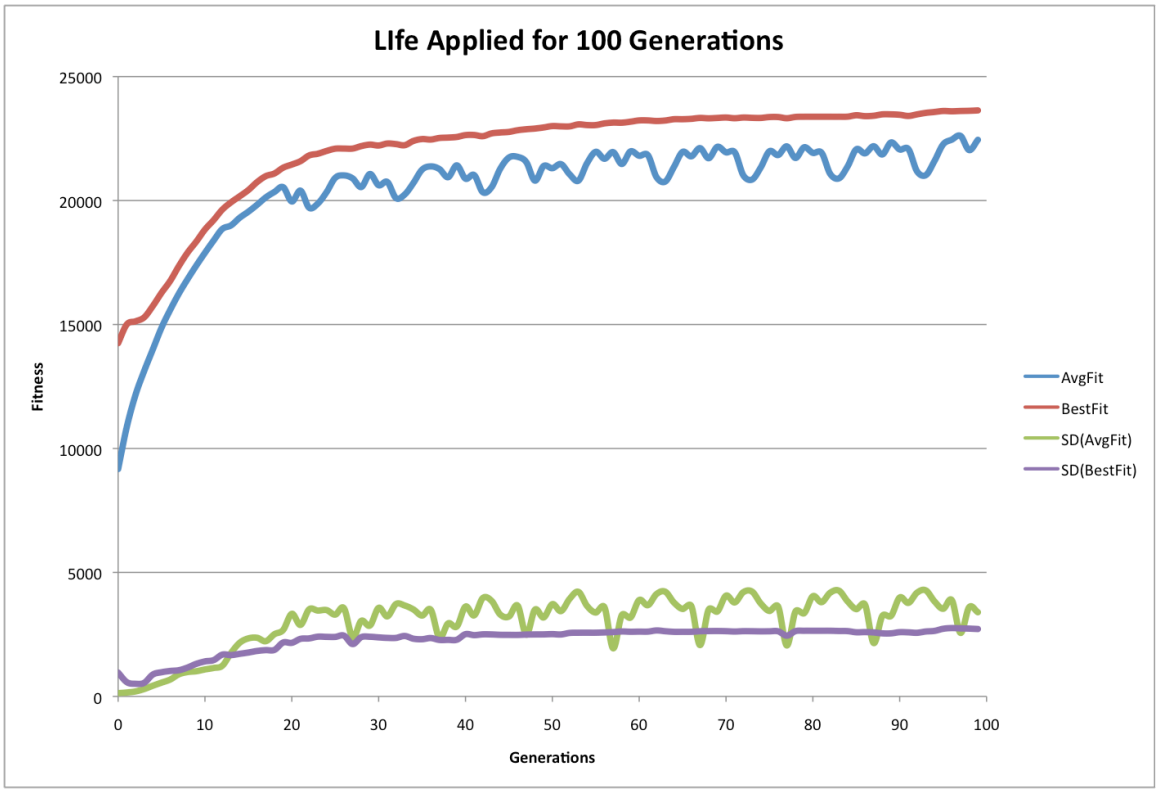

Figure 15: The fitness function is modified to apply the rules of *Life* for 100 generations instead of 20.

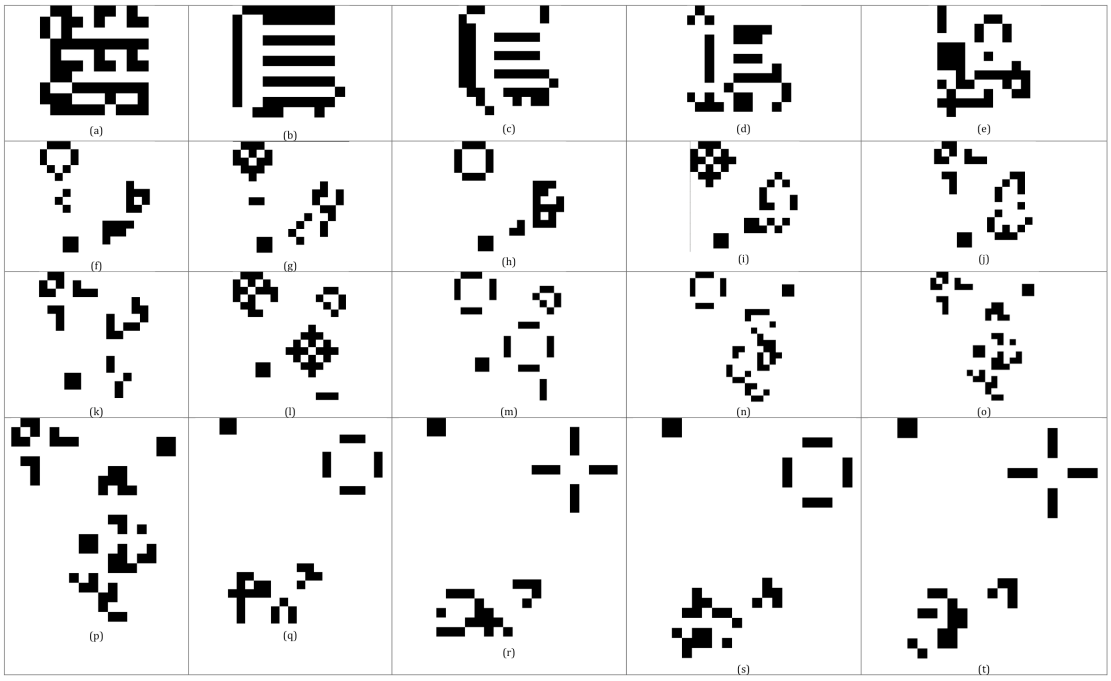

Figure 16: Best individual when running *Life* for 100 generations. Fitness of this solution is 32800.

Our fitness function ran the Game of *Life* for 20 generations while calculating an individual's fitness. After completing our experiments we decided to run *Life* for 100 generations instead. The results are shown in Figure 15.

Figure 16 shows the best individual from the experiments above. Only select generations are showed. We believe the highly repetitive patterns, like those shown in Figure 16(a), (b), (c), (l), (m), (n) and so on, contributed greatly to the exceptional fitness value. The last few generations (shown in Figure 16 (q)-(t)) represent an oscillating terminal state and a glider.

7.2 Hypothesis #2: Triggered Hypermutation

Our experiments show that triggered hypermutation indeed helps the GA find better solutions. Elitism serves a great purpose because it guarantees that solutions never drop in fitness from one generation to the next. However, it has the downside that the population can quickly converge to a suboptimal solution since only the best solutions so far are rewarded and kept from one generation to the next. However, the use of triggered hypermutation allows the exploration that mutation provides to kick in at precisely those moments when the GA has a tendency to converge to a suboptimal solution. In fact, we can witness in Figures 2 and 3 an apples-to-apples comparison of the GA with and without triggered hypermutation.

On the other hand, we expected triggered hypermutation to provide an even greater boost than we saw. At first, we believed that we didn't see more improvement due to our scaling factor. However, we saw that altering the scaling factor does not provide the behavior we wanted. Increasing the scaling factor resulted in great perturbations to the population. Decreasing the scaling factor did not introduce much diversity to the population, so its effects were negligible.

7.3 Hypothesis #3: Random Immigrants

We hypothesized that the influx of random immigrants would disrupt the population and hurt overall performance. Our hypothesis was correct. We can explain this by referring back to the findings of [4]. The influx of random immigrants provided randomly generated individuals to the population. These random immigrants hurt average performance and may have caused the solution found by elitism to be lost. In fact, we often saw that the insertion of random immigrants decreased performance from one generation to the next. When triggered hypermutation was tried along with random immigrants, it had no effect. This can be explained by realizing that triggered hypermutation requires the population to be mostly converged and to see little to no variance from one generation to the next. The insertion of random immigrants results in large jumps to the variance from one generation to the next. Thus, hypermutation is never triggered.

The results discussed in Section 6.2.1 showed that random immigrants does, in fact, efficiently track changing environments.

7.4 Contributions

Our research has taken a different approach than [10]. We have shown that it is possible to create interesting patterns in Conway's Game of *Life* by using a Genetic Algorithm and employing a few approaches previously attempted. Our findings on triggered hypermutation have proven it is useful when used in conjunction with elitism. This differs from the findings in [9], and provides a more interesting use for the technique. Our findings on random immigrants, on the other hand, confirm the findings in [9]; they show that the use of random immigrants is only useful when dealing with changing environments. Our research provides a basis for future research in the Game of *Life* and its ability to model complex systems. Our fitness function can be modified to reward the desired behavior of future researchers.

8. REFERENCES

[1] E. S. O. Bailleux and J. Chabrier. *Lecture Notes in Computer Science*. Springer, Berlin / Heidelberg, 2003.

[2] M. M. J. Chrutchfield and R. Das. Evolving cellular automata with genetic algorithms: A review of recent work. *Proceedings of the First International Conference on Evolutionary Computation and Its Applications (EvCA'96)*, 1996.

[3] H. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environment. *NRL Memorandum Report 6760*, 1990.

[4] H. Cobb and J. Grefenstette. Genetic algorithms for tracking changing environments. *Proceedings of the ICGA*, 1993.

[5] A. B. et al. Scale-free and hierarchical structures in complex networks. *AIP Conference Proceedings*, 661, 2003.

[6] S. S. et al. Cdma technology based reliable wireless mobile communication system on a single chip using cellular automata concept. *Engineering, Computing and Architecture, ISSN 1934-7197*, 1, 2008.

[7] M. Gardner. Mathematical games: The fantastic combinations of john conway's new solitaire game of life. *Scientific American*, 223:120–123, 1970.

[8] J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man & CyberneticsSMC*, 15:122–128, 1986.

[9] J. Grefenstette. Genetic algorithms for changing environments. *Parallel Problem Solving from Nature*, 2:137–144, 1992.

[10] D. Kazakov and M. Sweet. Evolving the game of life. *Proceedings of the Fourth Symposium on Adaptive Agents and Multi-Agent Systems (AAMAS-4), AISB convention*, 2004.

[11] A. W. C. Kleineidam and J. Tautz. Collective control of nest climate parameters in bumblebee colonies. *Animal Behaviour*, 63:1065–1071, 2002.

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

[13] T. Schelling. Dynamic models of segregation. *Journal of Mathematical Sociology*, 1:143–186, 1971.

[14] S. Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55, 1983.