A Study of Evolutionary Robustness in Stochastically Tiled Polyominos

Justin Schonfeld
Bioinformatics and
Computational Biology Program,
Iowa State University,
Ames, Iowa 50011
schonfju@iastate.edu

Daniel Ashlock
Department of Mathematics and Statistics
University of Guelph, 50 Stone Road East
Guelph, Ontario, N1G 2R4
dashlock@uoguelph.ca

ABSTRACT

Given an evolutionary optimization problem with many possible genotypes for each phenotype this study investigates if the evolved genes for a given phenotype are more robust to point mutation than randomly sampled genes for the same phenotype. This question is addressed using a cellular representation for polyominos in the plane. The evolutionary computation system optimizes for shapes which pack well onto the surface of a torus when dropped at random. For the majority of the evolved phenotypes the evolved genes for a given shape proved to be significantly more robust to point mutation than those sampled at random for that same shape. A few evolved genotypes, however, were not significantly more robust than those sampled at random and in some cases were less robust. These observations are placed in the context of the fitness landscape for the representation.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Miscellaneous

General Terms

Design

Keywords

Cellular Representation, Robustness, Evolutionary Computation

1. INTRODUCTION AND BACKGROUND

One of the advantages claimed for evolved systems is that they are more robust. One example of this is the energy landscape of proteins. Evolved proteins fold on energy landscapes that resemble huge funnels [1]. Randomly generated sequences of amino acids typically have energy landscapes with many shallow energy minima. This suggests that the natural proteins, produced by evolution, are a very special subset of the possible proteins. The specialness manifests itself not only in the ability to perform various enzymatic tasks but in the way that the protein is able to correctly fold to perform its job with relative ease, when compared to the space of all sequences of amino acids. This specialness also appears in evolved 2-D models of protein folding. Computer simulations of simplified protein folding using a 2-dimensional lattice model have shown that in a very specific environment "optimal" proteins found by an evolutionary algorithm are more robust to mutation than those found by a random walk [6]. This is a specific instance of a phenomenon of general interest: the way the robustness of structures depends on the methods used to create them.

The study here continues in a new direction previous work on robustness in functional optimization [5]. In the previous study evolutionary algorithms using tournament selection were compared to the great deluge algorithm [2], a stochastic hill-climber, and a random walk. For several functional optimization problems the tournament selection algorithm was found to produce more robust solutions than all the other algorithms *except* the great deluge. The great deluge was often, but not always beaten by the tournament selection algorithm. The great deluge also exhibited a huge variation in performance; its performance sometimes failed to be significantly different from that of the evolutionary algorithm because of a confidence interval of enormous width.

This study evaluates the robustness of evolved structures that arise under a representation with a many-one genotype to phenotype mapping. The number of different genotypes that encode a given phenotype that appears during evolution is combinatorially large and so provides a statistical universe for comparison with the genes that arise during evolution. The structures being evolved are shapes made of contiguous unit squares joined face to face. The encoding is a type of cellular encoding, following the clever idea of Frederic Gruau [3] in which direction for constructing a structure, rather than a direct specification of the structure, forms the representation.

The Question

This apparent ability of an evolutionary algorithm to select a more robust solution when robustness is not directly a fitness criteria has the potential not only to elucidate the behavior of evolution but also to aid engineers and computer scientists who tackle problems where robustness may be an important factor. In this study the term "robustness" is used to describe the probability a point mutation will fail to reduce the fitness of a solution. The type of robustness studied here is formally defined subsequently. In this paper we attempt to address the following question:

Question: In the presence of a many-one genotype to phenotype mapping do evolutionary algorithms find genes that are more robust than average among all genes coding for a given phenotype, assuming that robustness of the solution is not directly part of the fitness evaluation?

This study only addresses the preceding question in the framework of a single evolutionary computation problem, but hopefully gives an experimental design that can be used easily to test other problems.

Overview of Experimental Approach

The structures used in this study are *polyominos*. A polyomino is polygon formed from a number of unit squares joined together edge-to-edge, so that a full edge of each square is in contact when two squares are joined. Examples of polyominos are shown in Figure 1 while the dominant evolved polyominos evolved in this study are shown in Figure 4.

Populations of polyominos are evolved using a stochastic tiling criterion to evaluate fitness. Many copies of each polyomino in a population are placed over random points on a toroidal grid in an equitable order. If the space beneath a polyomino is completely open then it drops onto the grid, otherwise it fails to drop. Polyominos are dropped one at a time, occupying more and more of the space on the gird as fitness evaluation progresses. The fraction of the grid occupied by instances of the polyomino derived from a given gene is the fitness of that gene. This is a noisy fitness function and so a large toroidal grid is used to reduce the randomness of fitness evaluation.

In a population of evolved polyominos there are typically a small number of shapes that occur by the end of an evolutionary run. We call these the *dominant shapes* for that run. Since the population size if fixed this means that each dominant shape is encoded by many genes in the population. Some of the genes generating the same shape are identical, some are not. In addition, distinct runs sometimes share dominant shapes. For a given dominant shape we collect together all the evolved genes that yield that shape withing a single population (evolutionary run). These are the set of *evolved* genes for the shape in that population.

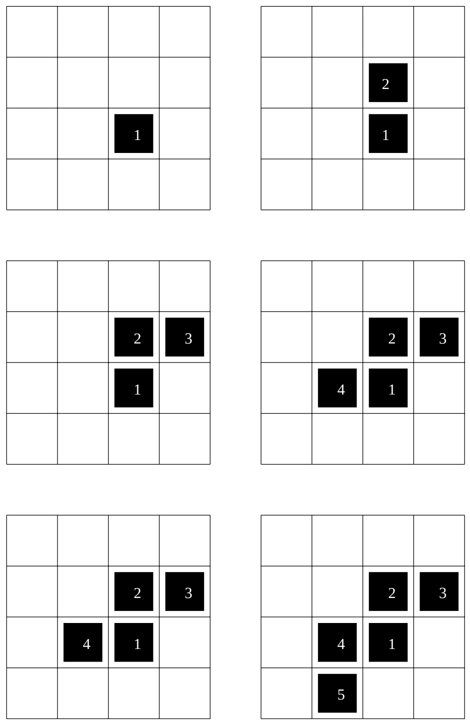
A key feature of robustness evaluation is the ability to sample genes for a given shape. We do this with an inverse gene creator. Given a shape this code fills in a set of instructions, making a uniform random choice in each step, that generates the shape. It first creates a gene that exactly generates the shape and then pads it, in a uniform random fashion, with additional unexecutable instructions. The genes generated in this fashion form a sample of the genes that *can* generate a given shape, enabling the robustness comparison with evolved genes.

The remainder of the paper is organized as follows. Section 2 specifies the representation for polyominos, the evolutionary algorithm, and the inverse gene creator as well as defining exactly the type of robustness the experiments test for, Section 3 specifies the experiments performed and gives the experimental results. Section 4 discusses and explains the results. Section 5 gives future directions for continuing the investigation of robustness.

2. MODEL SPECIFICATIONS

The representation used for polyominos is a simple list of integers. These form a linear gene which may be subjected to crossover and mutation in the usual fashion for string representations. The integers encode commands for how to grow the polyomino. This highlights an advantage of cellular representation. If we were to cross over two polyominos directly it would be challenging to produce a contiguous resulting shape. Probably some form of repair operator would be required, at a minimum. The list of directions is a simple string that can be plugged into fairly generic evolution code. While running the development algorithm that transforms this string into a polyomino some of the commands will not be executable, forming a type of "junk DNA", but this is simple and transparent compared to the repair operators that might be required.

Gene Expression



Int	Binary	Cell No.	Direction
4	00000100	1%1=0+1=1	00=up
13	00001101	3%2=1+1=2	01=right
38	00100110	9%3=0+1=1	10=left
21	00010101	5%4=1+1=2	01=right
31	00011111	7%4=3+1=4	11=down

Figure 1: An example of gene expression for the gene 4,13,38,21,31. Parsing this gene yields the sequence of developmental steps shown above.

Gene expression in this context, is the process that transforms the gene - a string of instructions - into a polyomino.

We will call the squares of our polyominos *cells*. The expression of each polyomino starts with a single square or cell, cell number zero. The gene then specifies how to grow additional cells onto the sides of already existing cells. Each gene is an array of 8-bit positive integers. Each of these integers is interpreted as a list of growth instructions. Each instruction specifies a cell number and a direction. A gene is expressed on a 2-dimensional grid by executing the list of instructions in sequential order along the gene. The last two bits of each integer are used to specify the direction to grow in (00 - up, 01 - right, 10 - left, 11 down). The remaining six bits are used modulo the number of cells currently expressed, plus one. The six bits $(mod\ n) + 1$ pick which of the n currently expressed cells, $1, 2, \ldots, n$ is the starting point for the attempt to grow in the direction given by the two least significant bits. If an instruction tells a cell to grow into an already existing cell then the instruction in not executable and is ignored. Cell numbers are assigned sequentially based on the order in which the cells appeared. Examine the length five gene: (4,13,38,21,31). A visual representation of the development of this gene is given in Figure 1. The first instruction starts at cell 1 and generates a second cell upward one grid square. The second instruction starts at cell 2 and grows to the right expressing cell 3. The third instruction starts at cell 1 and grows to the left expressing cell 4. The fourth instruction starts at cell 2 and attempts to grows to the right, resulting in a failure to execute the instruction. The final instruction starts at cell 4 and grows down creating the fifth and final cell.

Fitness Evaluation

The fitness of a shape is determined by how well it stochastically packs onto a toroidal grid. The population of genes is shuffled. The fitness evaluation routine then cycles through the population of genes and sequentially attempts to place shapes onto the toroidal grid. A pair of (x,y) coordinates are chosen at random to be the lower left corner of the shape on the grid. If the shape doesn't land in a position where every square it would occupy is empty then nothing changes, otherwise it occupies those grid squares. This process continues until at least half of the area of the board is covered. Shapes are assigned fitness values equal to the number of squares on the board that instances of them cover.

The Evolutionary Algorithm

Table 1: Evolutionary Algorithm

```
01. Initialize Population of Shapes
02. Do $g$ Times
03.     Evaluate fitness by stochastic packing
04.     Shuffle the population into size 4 tournaments.
05.     The two most fit members of each tournament copy
            over the two least fit.
06.     Copies undergo two point crossover
07.     Each copy undergoes single point mutation
08. Report results
```

The evolutionary algorithm used is given in Table 1. It uses single tournament selection algorithm with tournament size 4. The parameters used for population size and other similar details are listed in Table 3 in Section 3. The single point mutation used is replacement of the integer at one randomly selected position with another integer chosen uniformly at random from all eight bit integers.

Random Inverse Gene Creation

Table 2: Random Inverse Gene Creation Algorithm

```
01. Initialize a length $l$ gene vector with $g$ growth
        instructions and $f$ failures
02. Pick a cell at random from the shape to be the
        initial cell
03. For each list entry L
04.     If L is a growth instruction
05.         Pick an expressed cell at random
06.         Pick an unexpressed expressible neighboring
                cell at random
07.         Write a growth instruction
08.     If L is a failure instruction
09.         Pick an expressed cell at random
10.         Pick a neighboring expressed cell
11.         Write an unexecutable instruction
```

The algorithm for randomly sampling genes that create a given shape is called the *random inverse gene creator*. It is given in Table 2. It takes a given shape as input and randomly samples the space of all genes which can generate that shape. The number of executable instructions g and failures f can be computed from the length k of the gene and the number of cells c in the shape using the following obvious relation. First, $c = g + 1$ because the first cell is, in some sense, free. Second, $k = g + f$.

The algorithm first randomly assigns positions in the gene for failures (instructions that cannot be executed). A square is picked uniformly at random from the shape to be the initial square for the shape generated by the gene, cell zero relative to the gene's encoding of the shape. For each position in the gene the algorithm generates either a growth instruction or a failure, as specified by the positions of the failures. Note that a failure can be anywhere except the first instruction and, subject to that limit, they are placed at random. When generating a growth instruction the algorithm randomly picks an unexpressed square of the shape which is adjacent to at least one expressed square. If the square picked is adjacent to more than one expressed square then the square it grows from is chosen at random. A growth instruction is then generated that causes the unexpressed square picked to be grown from the expressed square picked. See Figure 2 for an example. To generate a failure an expressed square is picked at random, and then a neighboring expressed square is picked. Notice that after the first instruction all expressed squares are adjacent to expressed squares. A failure instruction is then generated by having the first expressed attempt to grow a new square into the second. See Figure 3 for an example.

Analysis

If we wish to test the hypothesis that evolved genes for a shape are, on average, more robust that genes picked at

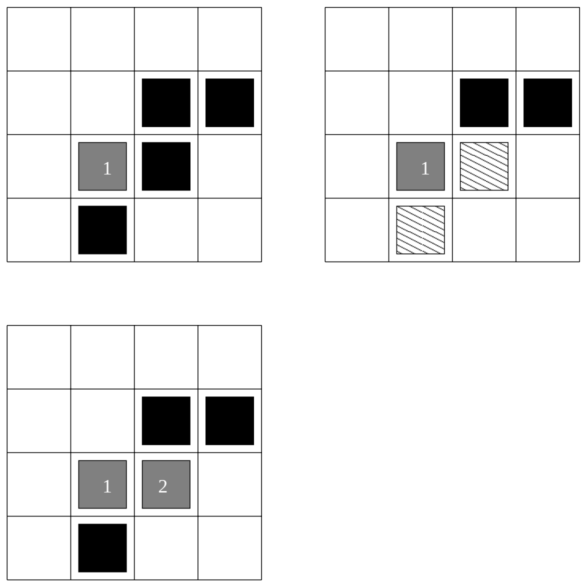


Figure 2: Example of generating a growth instruction. Displayed squares represent squares in the final shape. Grey squares represent expressed squares in the shape. Hatch-marked squares represent possible choices of squares to grow into. In this example a grown instruction is generated that causes square two to grow from square 1.

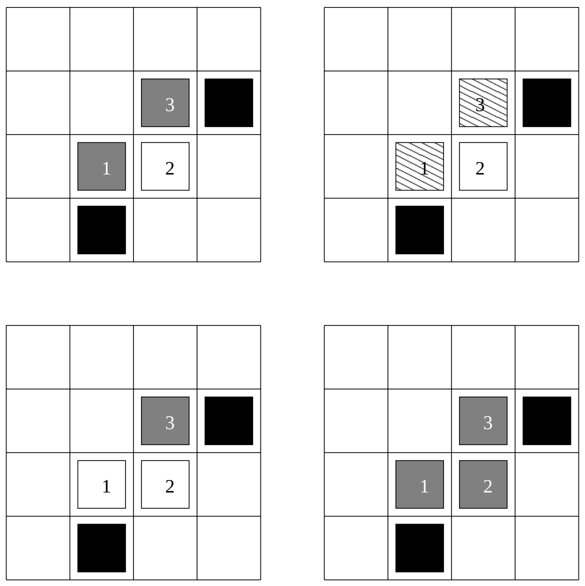


Figure 3: Example of generating a failure instruction. Displayed squares represent squares in the final shape. Grey squares represent expressed squares. The white squares represent the two squares involved in the failure instruction. The hatch-marked squares represent possible choices for the second square in the failure instruction. The failure chosen attempts to grow from square 2 to square 1.

random from those that generate the shape then we need a careful definition of robustness.

Definition 2.1. **Robustness** *The robustness of a gene is the probability that a single mutation will fail to transform the shape the gene expresses.*

Note that as robustness is a probability, we have

$$0 \le Robustness \le 1.$$

A robustness of 1 indicates a gene that cannot have the shape it encodes changed by point mutation while a robustness of zero indicates a shape that is changed by any possible point mutation. Recall that a point mutation consists of changing one of the eight bit integers in a gene at random to another eight bit integer selected uniformly at random.

The robustness for a gene is approximated by performing repeated point mutation trials on the gene. A trial was considered a success if a point mutation did not change the expressed shape of the gene, making point mutations a form of Bernoulli trial[4]. Bernoulli trials are experiments with two possible outcomes, like flipping a coin. The number of successes in a collection of Bernoulli trials follows the binomial distribution[4] and, for large numbers of trials, can be modeled with the normal distribution using the de Moivre-Laplace Theorem[4] which states the aymptotic of a binomial distribution is a normal distribution. For the genes encoding each dominant shape in the final population of each evolutionary run we calculate the combined number of successes for the evolved genes that encode the shape and compare it with the combined number of successes of n randomly sampled genes which encode the shape.

3. EXPERIMENTS

A set of thirty evolutionary runs was performed. The parameters for the evolutionary algorithm and the robustness assessment used in this study are given in Table 3. The dominance threshold is the number of genes, minimum, that must encode a shape for the shape to be considered dominant. Note that dominant does not mean majority, rather it means "apparently surviving". Point mutations sampled are the number of point mutation trials used to approximate the robustness of a single evolved or synthetic gene. Number of synthetic gene samples is the number of synthetic genes generated when assessing the comparative robustness of an evolved gene for a given shape. The other parameters are standard evolutionary algorithm parameters.

The statistical test used to determine if the evolved genes within a single population for a given shape were significantly more robust than randomly sampled genes for that shape was a z-statistic for the comparison of the probability of success in two collections of Bernoulli trials [4], page 335. For all the evolved genes in a given evolutionary run that produced a dominant shape 1000 point mutations were sampled and the number x that failed to modify the shape as well as the number of point mutations n attempted were recorded. For each of 200 sample genes generated with the random inverse gene creation algorithm the number of point mutations y that failed to change the shape of the gene and the total number of point mutations m attempted were recorded. The z-statistic used to compare these two collections of Bernoulli trials is:

$$Z_{\alpha/2} = \frac{\frac{x}{n} - \frac{y}{m}}{\sqrt{\frac{\left(\frac{x+y}{n+m}\right)\left(1-\frac{x+y}{n+m}\right)(n+m)}{nm}}} \tag{1}$$

Parameter	Value
Evolutionary algorithm	
Number of evolutionary runs	30
Population size (p)	60
Gene size	60
Tournament size (t)	4
Number of generations (g)	500
Toroidal grid width	600
Toroidal grid length	600
Probability of a point Mut.	0.200
Robustness Assessment	
Dominance threshold	4
Point mutations sampled	1000
Number of random genes sampled	200

Table 3: Evolutionary algorithm and robustness assessment parameters.

In the course of thirty evolutionary runs, forty dominant shapes were observed. Twenty-eight of them being unique, others occurred in multiple, distinct runs. A shape is considered to be a dominant shape if greater than 5% of the genes (at least 4 genes) in the population code for that shape. After performing a statistical analysis summarized in Table 4 it was found that thirty-seven of the dominant shapes were expressed by evolved genes which were significantly more robust to point mutation than randomly sampled genes for the same shapes. Two of the dominant shapes were expressed by evolved genes which were significantly less robust to point mutation than randomly sampled genes (N:22, AA:14). These same two shapes also were expressed by evolved genes, in different runs, that were significantly more robust to point mutation than randomly sampled genes. One shape was also expressed by evolved genes which were not significantly more or less robust than the randomly sampled genes (H:11). The shapes H and S, 5x4 and 4x4 squares respectively, appeared most frequently. The 4x5 square, on the other hand, only appeared once.

4. DISCUSSION AND CONCLUSIONS

The cellular encoding for shapes used in this study was chosen to give a many-one mapping from genes to shapes thus creating room for diverse encodings of a given shape, both robust and non-robust. The hypothesis that evolution usually finds genes that exhibit above-average robustness is strongly confirmed by the experiments with 37 of 40 dominant shapes having significantly more robust dominant shapes. This assessment uses the 95% confidence criterion with a z-value of 1.96 but the significant z-values in Table 4 range from 2.24 to 32.64. The former represents modest

Shape Type	Shape Class	Shape ID	Count in Population	Z-value
A	5x5	0	32	10.30
B	5x5	1	22	10.71
C	5x5	33	50	2.98
D	5x5	17,36	57,4	17.45, 2.46
E	5x5	34	4	3.60
F	5x5	35	50	4.62
G	5x4	12	52	4.23
H	5x4	2,11, 31,32	60,54, 59,54	14.03, 0.18 2.71, 18.96
I	4x6	13	58	4.07
J	4x6	27	20	9.53
K	4x6	32	56	20.12
L	4x5	3	54	19.92
M	4x5	20	32	8.72
N	4x5	19,22	17,56	8.03, -11.15
O	4x5	18,26	6,35	2.24, 15.29
P	4x5	10	56	11.25
Q	4x5	28	56	20.67
R	4x5	29	4	2.59
S	4x4	6,9, 23,37	56,55, 57,56	32.64, 8.14, 7.33, 8.95
T	3x5	4	55	8.25
U	6x4	7,15	55, 49	4.86, 10.31
V	6x4	39	25	13.89
W	6x4	16	6	6.58
X	6x4	24,38	56, 30	20.13, 13.58
Y	6x4	21	55	11.82
Z	6x4	25	57	4.44
AA	6x3	5,14	56, 57	3.17, -6.96
AB	6x3	8	56	14.11

Table 4: For each shape class this table gives the following information. The shape type indexes the depiction of the shapes in Figure 4. The shape class is the size of bounding rectangle for the shape. The shape ID is a unique identifier assigned each time a shape is a dominant shape in a population. Population count is the number of occurrences of the shape in a 60 member population. The Z-value is the Gaussian statistic for significance of relative robustness.

but real significance while the latter corresponding to an extraordinary level of robustness. One of the dominant shapes did not have significantly different robustness from sampled genes at the resolution of this study. Intriguingly two dominant shapes had representations that were significantly *less* robust than the sampled genes.

These two peculiar cases are made even more interesting by the fact that one of them, Shape **N** in Figure 4 was dominant in two populations. In the first it has a genetic encoding with robustness statistic $z = 8.03$ indicating substantially above-average robustness. In the second the value is $z = -11.15$. This suggests, first of all, that shape **N**, a 5×4 rectangle with its lower left corner missing, has at least two very different encodings that can take over an evolving population. Second it suggests that the fitness of a shape can be substantially independent of the robustness of its genetic encoding in the evolutionary computation system presented here.

Putting this result into the metaphor of fitness landscapes the algorithm has located two distinct hills for shape **N** one of which is broader and flatter than the other. This result

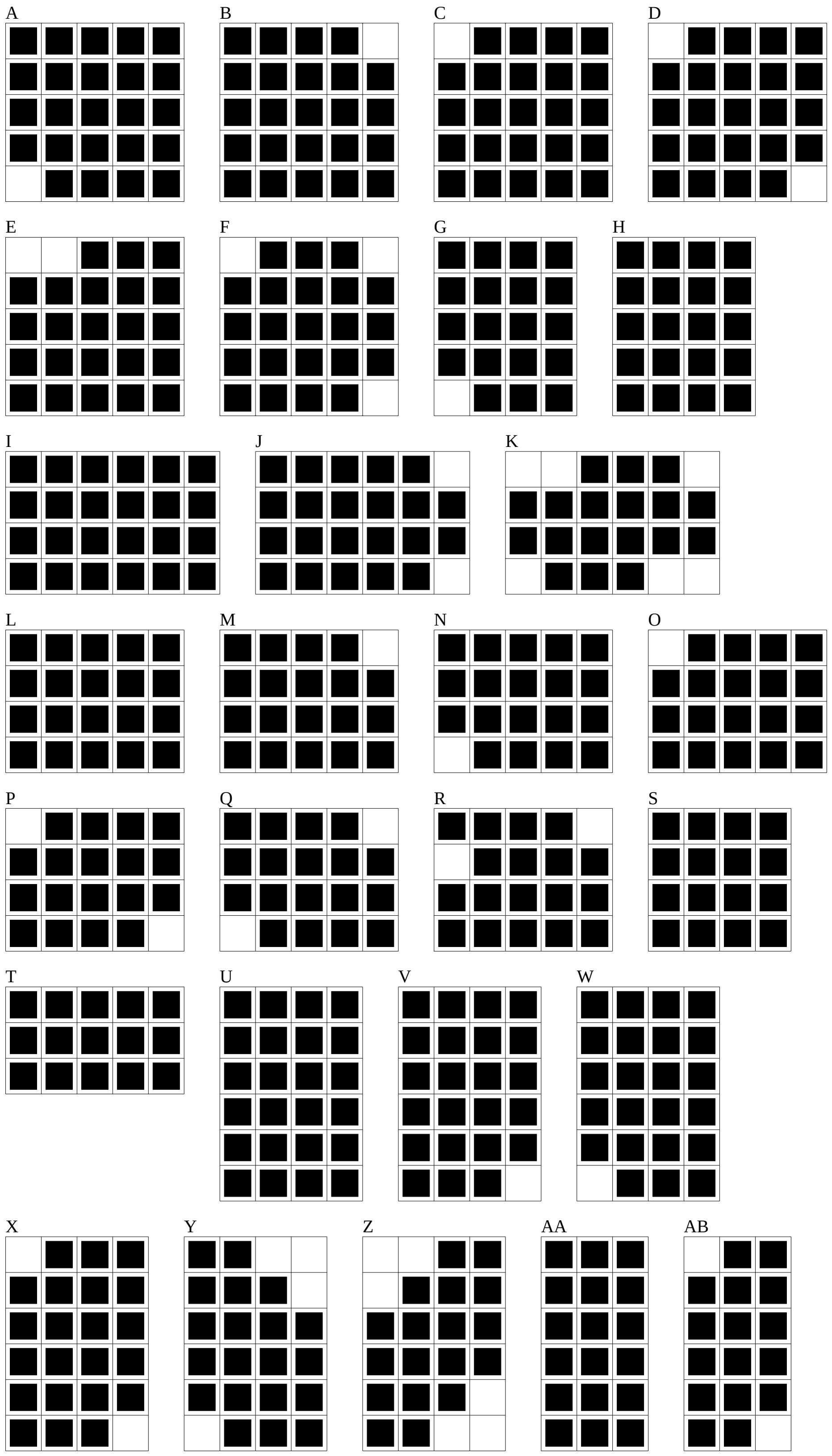


Figure 4: Each of the twenty-eight dominant shapes.

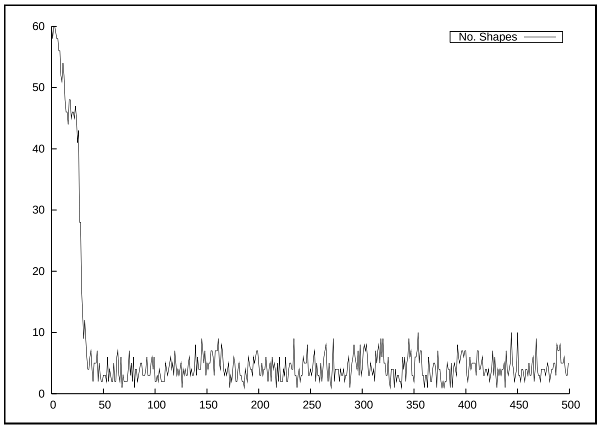


Figure 5: The number of distinct shapes in the population as a function of evolutionary time for a typical run (trial 5).

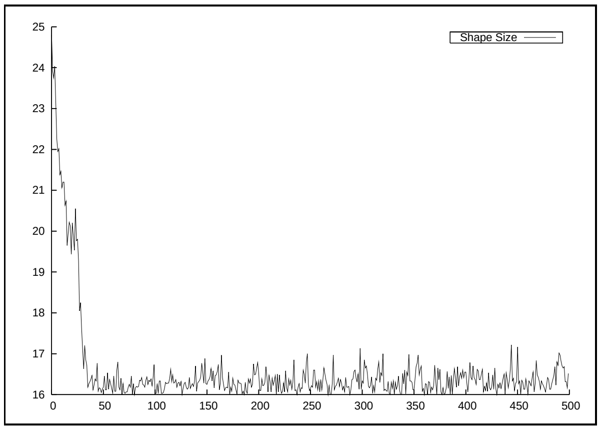


Figure 6: The average size of the shapes in the population as a function of evolutionary time for a typical run (trial 5). The dominant shape here was a 4x4 square with shape ID 6.

is related to the results reported by Chris Adami's group [7] on a more complex representation. Their paper coins the term "survival of the flattest" to describe the tendency of evolution to select broad hills in the presence of a high mutation rate. In the case of Shape **N** there is not a trade-off of fitness for genetic stability but rather two different peaks with exactly the same fitness (shape determines fitness in this system) and clearly different levels of genetic stability.

We can conclude that in this evolutionary system high levels of robustness are likely but not certain. It is likely that there a founder effects that explain the wide scatter of robustnesses. A question that this leaves open is the ability of a good shape to discover more robust representations as evolution continues. The results for shape **N** suggest that some encodings of a good shape have no high-fitness mutational path to higher robustness.

We conjecture that the source of kind of robustness in evolutionary computation reported here is durability against disruption by the variation operators. As can be seen in the example present in Figure 8 there is substantial pressure to conform to fit in with the dominant shape within an ecology. Suppose that an ecology arrives at a point in evolution where most of the genes encode the same shape. At this point the e only remaining source of selection pressure is for the ability to not change the shape encoded by a gene when it is subjected subjected to crossover and mutation. Forms that are more robust in the sense of retaining their shape when mutated or crossed over will have a higher effective reproductive rate in spite of having an identical (shape based) fitness.

Examining Figure 4 it is clear that the fitness function favors compact shapes. The most compact shape possible is a square and squares like shape **S** or near squares like shapes **A**-**F** are common. Rectangles with similar length and height, with or without a missing corner, are also also common. Shapes **Y** and **Z** suggest that near-hexagons are also possible. All the shapes observed as dominant shapes are arguably near shapes that tile the plane well.

This favoring of near tiling shapes answers an obvious question about the trajectory of evolution in this system. Shapes can either cooperate (nearly tile) or fail to cooperate (attempt to block other shapes fall). If the shapes were attempting to block other shapes fall then one would expect limbs, long projections. This sort of projection is common in initial populations, an example of which is shown in Figure 7. The reason for this is probably relatively small population size. By the end of 500 generations of evolution the population is highly inbred. As a result effective strategies for blocking other genes would block other instances of the shape a given gene encoded.

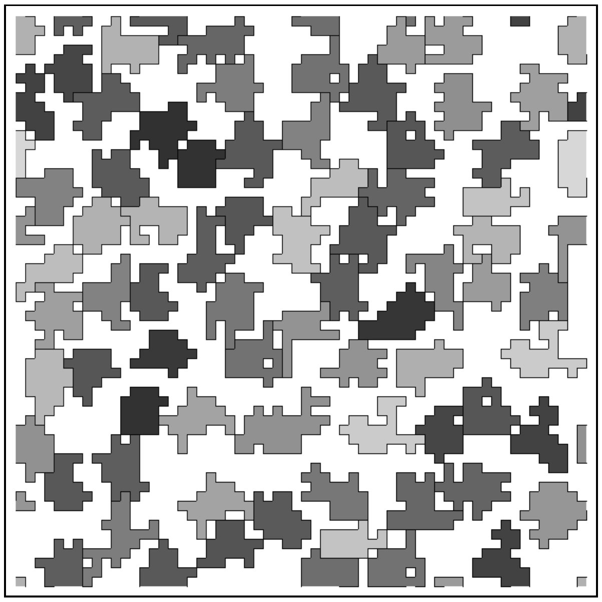


Figure 7: A picture of shapes as they dropped during fitness evaluation of generation zero in a reduced-size world.

It is interesting to compare Figure 7, a starting population, with Figure 8, a final population. Both these figures are drawn from preliminary studies on a smaller toroidal grid than the runs used to study robustness. Notice that in the final population far less diversity of shapes. Most genes code for a 4x4 square with three non-dominant shapes encoding for squiggly things. Recall that each gene is permitted to drop multiple copies of its shape during fitness evaluation. The fitness function cycles through the population in a random order. The number of distinct shapes and average size of shapes as function of evolutionary time for evolutionary run number 5 are given in Figures 5 and 6. This run is typical of the behavior of all thirty runs. The average size of shapes drops sharply at the beginning of evolution as does the number of distinct shapes. This suggests that the need to occupy space with a single polyomino is not the driving force of evolution in this system. Rather, co-existence of instances of the polyominos derived from a particular gene seem to be the order of the day.

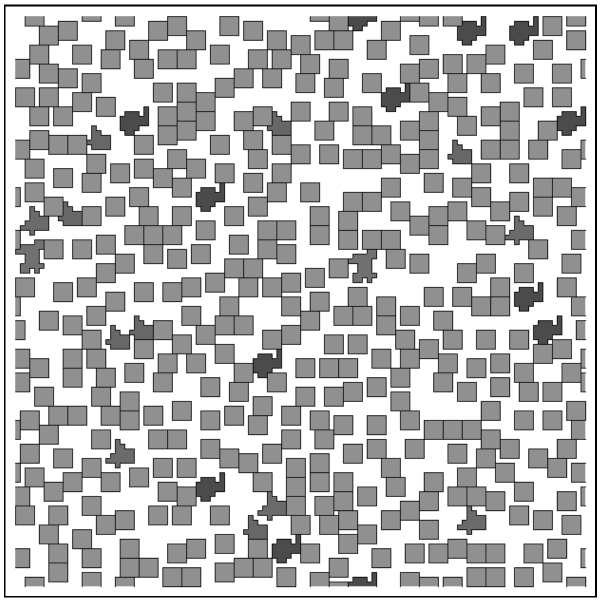


Figure 8: A picture of shapes as they dropped during fitness evaluation of generation 499 in a reduced-size world.

5. FUTURE WORK

The basic system implemented for a particular cellular encoding of shapes in this study can be repeated for other representations and fitness functions. The ability of evolution to locate robust solutions is an important feature of evolutionary algorithms that requires careful quantitation. In addition to the type of robustness treated here, robustness to mutation, robustness against variation in fitness criterion should also be studied. In a system that samples fitness cases such an assumption of robustness is implicit. Succeeding on those fitness cases encountered in evolution is supposed to produce general solutions. Quantitating robustness thus might improve the theory of allocation of fitness trials.

Focusing more narrowly on the shape system several areas for additional study are available. Adopt the definition that cooperation between shapes is represented by shapes that almost tile and failure to cooperate by shapes that block tiling and waste space. Would increasing the population size or restricting mating generate less cooperative shape? This notion could be explored further by designating "predator" and "prey" shapes with different fitness criteria. Prey keep the fitness criteria used in the current study. Predators are less common and are dropped first. Their fitness is the number of instances of the prey shapes that they prevent from dropping. We conjecture this would create predator shapes that block far more space than they occupy, relative to compact "prey" shapes.

The tentative conclusion from Section 4 that some genetic encodings for high fitness shapes have no mutational path to higher robustness can be checked. The software already developed for this work samples the point mutants of a gene to check the probability a point mutation will change a shape. This software can be modified to sample the neutral fitness mutation graph of a gene. This graph has some subset of the gene space as its vertices. The edges of the graph are pairs of genes that (i) encode the same shape and (ii) differ by a point mutation. Starting with a given gene, point mutations that do not change the shape are added to the graph. These, in turn, are also checked and their neighbors incorporated. The graph has potentially immense size. Sampling it will provide evidence about fitness-neutral paths to higher robustness as well as the "robustness landscape" near the original gene.

Finally, the ability of evolution to discover robustness in this system can be surveyed for different parameters of the evolutionary algorithm in the usual fashion. This would consist of sensitivity studies to the type of crossover, rate and type of mutation, population size, and other factors such as population structure.

6. REFERENCES

[1] C. H. C, M. P. Eastwood, M. Prentiss, Z. Luthey-Schulten, and P. G. Wolynes. Folding funnels: the key to robust protein structure prediction. *Journal of Computational Chemistry*, 23:138–146, 2002.

[2] G. Dueck. New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 90:86–92, 1993.

[3] F. Gruau. Automatic definition of modular neural networks. *Adaptive Behaviour*, 3(2):151–183, 1995.

[4] R. J. Larsen and M. L. Marx. *Introduction to mathematical statistics and its applications*. Prentice Hall, Engelwood Cliffs, New Jersey, 1981.

[5] J. Schonfeld and D. Ashlock. Comparison of robustness of solutions located by evolutionary computation and other search algorithms. In *Proceedings of the 2004 Congress on Evolutionary Computation*, volume 1, pages 250–257, Piscataway, New Jersy, 2004. IEEE Press.

[6] D. Taverna and R. A. Goldstein. The distribution of structures in evolving protein populations. *Biopolymers*, 53:1–8, 2000.

[7] C. O. Wilke, J. L. Wang, C. Ofria, R. E. Lenski, and C. Adami. Evolution of digital organisms survival of the flattest. *Nature*, 412:331–333, 2001.