6-2017

Effective ANN Topologies for Use as Genotypes for Evaluating Design and Fabrication

John R. Peterson
Union College - Schenectady, NY

Effective ANN Topologies for use as Genotypes for Evolutionary Design and Invention

By

John R. Peterson

* * * * * * * * *

Submitted in partial fulfillment

of the requirements for

Honors in the Department of Computer Science

UNION COLLEGE

May, 2017

Abstract

PETERSON, JOHN R. Effective ANN Topologies for use as Genotypes for Evolutionary Design and Invention. Department of Computer Science, May, 2017.

ADVISOR: John Rieffel

There is promise in the field of Evolutionary Design for systems that evolve not only what to manufacture but also how to manufacture it. EvoFab is a system that uses Genetic Algorithms to evolve Artificial Neural Networks (ANNs) which control a modified 3d-printer with the goal of automating some level of invention. ANNs are an obvious choice for use with a system like this as they are canonically evolvable encodings, and have been successfully used as evolved control systems in Evolutionary Robotics. However, there is little known about how the structural characteristics of an ANN affect the shapes that can be produced when that ANN controls a system like a 3d-printer. We consider the relationship between certain structural characteristics of an ANN and the ability of that ANN to produce complex geometric shapes by controlling a 3d-printer. We develop an understanding of shape complexity for 2d-shapes in a simulated 3d-printer in order to use Genetic Algorithms to optimize ANNs with fixed structures to produce complex outputs and assess the relationship between topologies of ANNs and the systems success in producing complex outputs under evolutionary optimization.

Contents

1 Introduction

Genetic Algorithms have shown significant utility over the past twenty years in the optimization of problems that are difficult for humans to systematically approach. There is significant appeal to the possibility of building automated systems to produce solutions to complex problems with limited or no human intervention. One problem that clearly falls into this category is that of physical design. Often, a design task is proposed only as a loosely-coupled set of constraints to be optimized to where the relationships between these constraints make designing an appropriate product a very difficult task. Genetic Algorithms have been used very successfully to produce designs for a number of these difficult design problems, ranging from antennae [8], to cantilever bridges made from LEGOS [5], to VLSI circuits [2]. To date, much of the work done using Genetic Algorithms for design have focused on evolving descriptive models of objects, which some researchers have said results in a *Fabrication Gap*, a gap between what can be automatically designed, and what can be automatically manufactured [11]. In particular, evolved designs, when entirely decoupled from the process needed to manufacture them, may be difficult or even impossible to manufacture once designed.

One proposed solution to this Fabrication Gap is to evolve the manufacturing procedure used to produce a physical design, rather than the design itself. The hope is that while other approaches result in only a description of an object optimized to a set of design parameters, this approach would result in both a design and a validated procedure for the manufacture of that design. Over the past few years, work at Union College on EvoFab [12][7] has begun the process of implementing an evolutionary system in this philosophy. With EvoFab, we have begun to approach the problem of evolving manufacturing procedures by incorporating a 3d-printer, modified to be controlled by an Artificial Neural Network (ANN), into a GA system which evolves populations of ANNs optimized to control the 3d-printer to produce solutions to physical design tasks.

In the EvoFab system, ANNs are used as solutions for evolutionary optimization (genotypes) which control a 3d-printer in order to produce an expressed geometric result (phenotype). ANNs are a natural choice for genotypes to be used in this system. Previous iterations of the EvoFab project relied on linear instruction sets as genotypes for evolution. However, it became clear that these linear instruction sets did not behave well under the operations needed for GAs (mutation and crossover). Small changes made to the genotype via mutation resulted in physical phenotypes that were drastically dissimilar to the parent. Similarly, crossover was of little value because linear snippets of assembly instructions were very sensitive to context and had no value when used as building blocks [7]. In contrast, ANNs are generally understood to be canonically evolvable genotypes, where a minor mutation to the genotype will produce a minor

change in the expressed phenotype, and where the relative lack of context sensitivity of snippets of the genotype encoding permit reasonable results under crossover.

However, while we know that ANNs are appropriate encodings for use as genotypes for Genetic Algorithms, we do not have a clear sense as to their behavior as control systems for 3d-printers. ANNs have been successfully used as evolved control systems in evolutionary robotics [4]. However, relevant problems and desirable solutions in evolutionary design differ significantly from those in evolutionary robotics. In particular, it is important for a manufacturing system to be able to produce "interesting" outputs, where the shapes produced are sufficiently complex to be usefully composed into a complex physical product. The question of how "interesting" objects produced by an ANN-controlled 3d-printer can be is particularly important when those ANNs are used as genotypes for a GA system tasked with novel invention and fabrication. In particular, we ask: *What structural characteristics of an ANN impact the geometric complexity of the shapes those ANNs can produce when controlling a 3d-printer?*, and *What structural characteristics of an ANN are likely to improve that ANNs suitability as an evolved control system for invention and fabrication with a 3d-printer?*.

2 Approach and Constraints

The aim of this work is to develop an understanding of the relationship between the structural characteristics of ANNs and the geometric complexity of the shapes produced by ANNs functioning as control systems for a 3d-printer system. To approach this problem, we must develop a quantitative understanding of the "geometric complexity" of a shape. With a quantitative understanding of "geometric complexity", we can use this measurement as an objective function for evolutionary optimization. Through large numbers of trials on populations of ANNs with different fixed structural topologies, we can establish an understanding of how those the particular characteristics featured in those topologies impact "geometric complexity".

Because of the additional challenges imposed when working in the physical EvoFab system (namely hardware stability issues), our work focuses on results and testing using a simple printer simulation. Furthermore, since the current generation of EvoFab is restricted to 2d outputs, we direct our focus toward geometric results in two dimensions.

3 EvoFab: A System for Evolutionary Fabrication with 3d Printers

3.1 EvoFab 0.3 System Design

EvoFab 0.3 uses closed-loop manufacturing specifications instead of the open-loop linear instruction sets typical to 3d-printers. As opposed to evolving populations of linear instruction sets, this system evolves Artificial Neural Networks (ANNs) which output motor instructions based on sensor input from a sensor array on the printer hardware. Our conjecture is that these ANNs will be significantly more evolvable than the linear sequences used earlier in EvoFab, and will be less brittle under mutation and crossover.

EvoFab 0.3 consists of the fabrication hardware and a control/GA system written in Python. The fabrication hardware is based on a RepRap 3d printer kit sold by Velleman [13], modified to accommodate a pneumatic extruder, two extra end-stops, a photo resistor array around the tip of the extruder, a conveyor-belt to replace the static build platform, and a replacement control board. Typically, members of the evolved population are 3-layer feed-forward ANNs with 8 inputs, one for each sensor in the photo resistor array, and four outputs which control motor motion. The system is shown in Figure 1. A system diagram is shown in Figure 2.

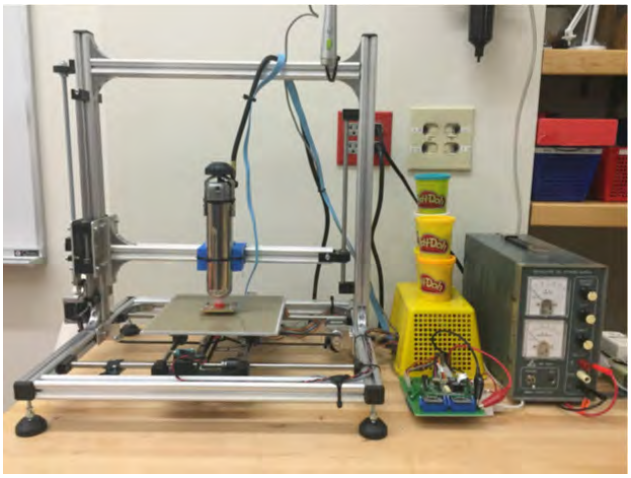


Figure 1: The EvoFab fabrication system

When running, EvoFab 0.3 runs each of the ANNs in order. After an arbitrary completion time, the ANN is stopped and the hardware output is evaluated by a computer vision system. Then, the conveyor belt activates, moving the output off the build surface and allows the next candidate ANN to be evaluated. After evaluating and ranking all candidate ANNs based on fitness, the system performs a cull step on a given percentage of the population. These culled members are replaced by fitness-proportional breeding of the remaining population. The evolution and evaluation procedure for EvoFab 0.3 is shown in Figure 3.

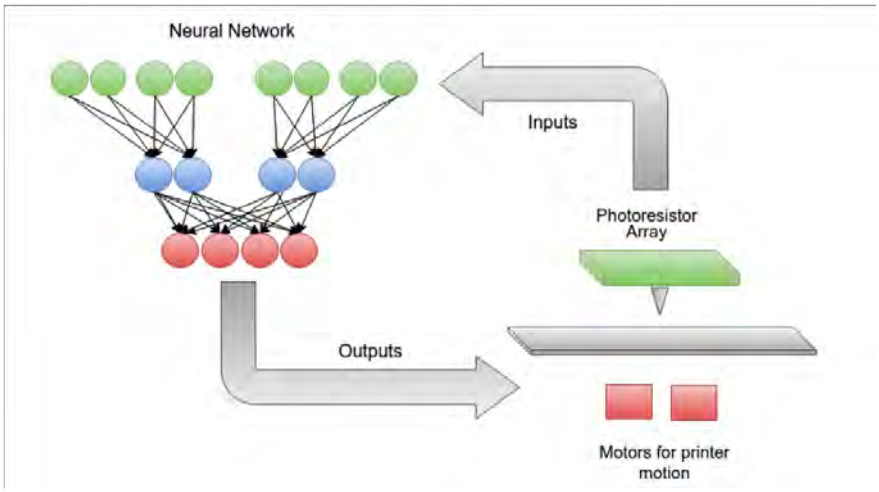


Figure 2: System schematic of the EvoFab system

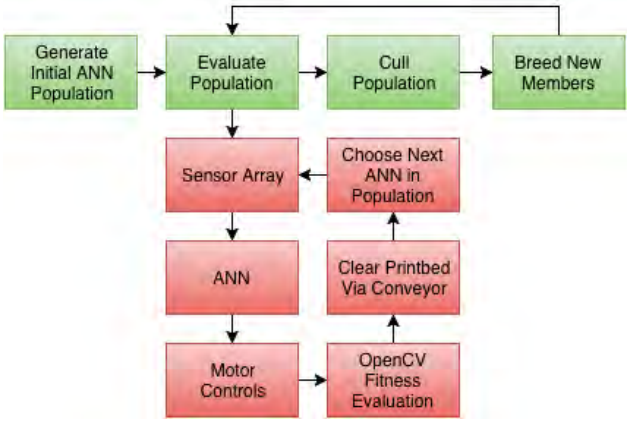


Figure 3: The EvoFab procedure for evolution and evaluation

3.2 EvoFab Simulation

Previous work developing EvoFab and testing the system design as a proof of concept have used a 2d simulation of the printer written in Python. This simulation describes the printer build platform as a raster grid where the print head draws in those cells.

For this work we have improved and extended the existing EvoFab simulation. The simulation used allows the print head to move in cell coordinates. The size of the print head is defined as some multiple of grid cells, allowing for an arbitrary-resolution raster drawing area. The photo resistor array is simulated with a 3×3 grid of cells where the dimension of each sensor cell is some multiple of grid cells. The simulated sensor response is the percentage of the cells in view of each sensor cell that are filled with material. Sample shapes produced in simulation by randomly-generated simple 3-layer feed-forward ANNs are shown in Figure 4.

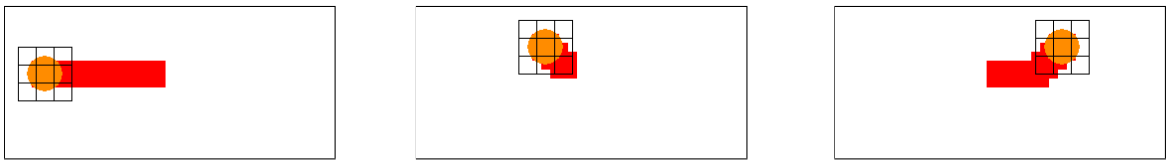


Figure 4: Sample outputs in the EvoFab simulation from three randomly-generated, simple ANNs. The orange circle shows the print head, with the sensor grid around it. Red shows grid cells that have been filled.

Label	Image	Compressed Size
A		51680
B		57707
C		6879
D		5036
E		11436

Table 1: Sample input images and calculated compression-based "complexity" measure

4 Describing the Complexity of 2d Shapes

In order to begin to understand the relationship between ANN complexity and geometric result produced by a 3d printer controlled by that ANN, we first have to approach the question of how to quantify *shape complexity*. While humans can intuitively assess the complexity of an object or shape relative to others, to use complexity as an objective function for optimization we need a quantitative measure of it. There are a few possible approaches to the problem. For input images of smooth curves (as would be used in the physical EvoFab system) there are two clear possibilities.

4.1 Kolmogorov Complexity

The Kolmogorov complexity of a string is defined as the shortest binary computer program for computing that string. By describing the information content of a string based on the smallest unit of computation that can produce that string, this approach describes a general understanding of "how much information" exists within a piece of data. [3]

If we are to understand an image as a string, it seems possible that the Kolmogorov complexity of an

image of a 2d-shape may give results consistent with an intuitive understanding of the geometric complexity of that shape. However, Kolmogorov complexity is uncomputable in general [3], so defining shape complexity in this way would depend on some measure that approximates Kolmogorov complexity.

In order to assess the suitability of a measure of shape complexity based on an approximation of Kolmogorov complexity, we implemented a measure based on the size of a compressed image and tested it on shape inputs with varying qualitative "interestingness". Results of this exploratory testing are shown in Table 1.

Although images A through D seem to be ranked in the expected order, image E, a relatively straight line has a significantly higher "complexity" as computed than either image D or C. Furthermore, this measure gives images A and B "complexities" a full order of magnitude above those given to images C and D. These results likely stem from the fact that although the compressed size of an image should be somewhat related to Kolmogorov complexity, it is a fairly poor proxy when relying on real-world implementations.

4.2 Shannon Entropy

Shannon entropy is a measure of the uncertainty of a random variable. For a random variable x and a probability density function (PDF) $p(x)$, the entropy $H(x)$ of that variable is given generally by

$$H(x) = -\int_{-\infty}^{\infty} p(x) \log p(x) dx$$

In the case of discrete data, entropy is given by

$$H(x) = \sum_{i} p_i \log p_i \tag{1}$$

Page et al. develop a measure of shape complexity for 2d and 3d shapes based on the Shannon entropy of curvature [9]. This measure has been shown to strongly correlate with human perceptions of shape complexity [6] and has been successfully used to describe the complexity of evolved robot morphologies [1]. We base our implementation of entropy of 2d-shape curvature on the model given by Page et al.

For smooth input images, we use openCV to identify the contour of a 2d curve in an input image. To reduce the level of noise in the placement of the points identified along the contour, we downsample the points by an order of magnitude. As computed by openCV, the points along the contour are listed in order, allowing us to compute line segments between each pair of points. A test image with marked points and line segments is shown in Figure 5.

Page et al. note that for a curve that is sampled uniformly along arc length, curvature is equivalent to

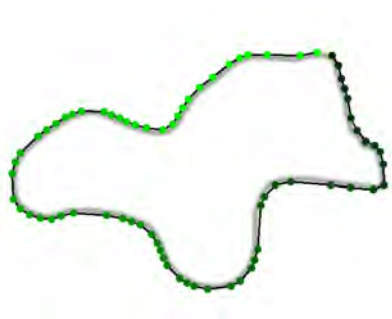


Figure 5: Sample input with downsampled points for exterior contour and line segments marked. Points are ordered lightest green to darkest green.

Lower	Upper	Probability
0.0	7.0	0.166666666667
7.0	14.0	0.222222222222
14.0	21.0	0.305555555556
21.0	28.0	0.138888888889
28.0	35.0	0.0277777777778
35.0	42.0	0.0277777777778
42.0	49.0	0.0
49.0	56.0	0.0
56.0	63.0	0.0
63.0	70.0	0.0277777777778

Table 2: Probability Density Function (discrete) calculated from the sample input

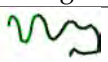
Label	Image	H
A		0.878998287142
B		0.68094959913
C		0.648913678978
D		0.583475152381
E		0.446103952925

Table 3: Sample input images and calculated entropy

Label	Image	H
A		0.515925693453
B		0.512699872124
C		0.446103952925
D		0.399160919416

Table 4: Sample input images and calculated entropy (simulation)

turning angle. We compute the turning angle for each pair of consecutive line segments in our image, and produce a discrete PDF for the curvature of the input over M bins with a constant bin width (so results for two different inputs are comparable). The PDF for the sample input shown in Figure 5 is shown in Table 2.

With the PDF computed, we simply calculate entropy for sampled data as given in Equation 1. Results for the standard set of smooth inputs are shown in Table 3.

5 Measures of Shape Complexity in Simulation

In order to be able to use any measurement of shape complexity as an objective function for optimization in the simulated EvoFab system, that measure has to appropriately evaluate the low-resolution raster shapes that are produced as expressed phenotypes in simulation.

Because it performed poorly in initial testing we did not implement the compression-based pseudo-Kolmogorov complexity measure for use with the EvoFab simulation. Instead, we focused on an adapted

implementation of entropy of curvature along with other simpler geometric descriptions.

5.1 Entropy of Curvature in Simulation

To compute turning angles on the exterior of a raster shape in simulation we first compute the exterior contour of the shape by identifying "boundary lines" between filled and unfilled cells and tracing a path composed of those lines. We then compute the turning angle of consecutive line segments spanning two points on the contour. This ensures that there is more variation in the computed angle than simply 0 and 90 degrees. Entropy is then computed as usual. Results of initial testing are shown in Table 4.

Interestingly, while shapes A and B are ranked as expected, the solid square, C, is ranked as having higher complexity than shape D. This discrepancy highlights the possibility that although entropy of curvature appeared to correspond well with shape complexity for smooth inputs, it does not seem particularly well-suited (or easy to define well) for use in a low-resolution raster-based simulation.

5.2 Alternative Measurements for Use in Simulation

Because of the computational expense and poor initial results of the compression-based complexity measure, alternative geometric objective functions were explored. The following measures were implemented:

- Ratio of perimeter of the shape to the units of filled area
- Ratio of perimeter of the shape to the perimeter of a "bounding rectangle"
- Ratio of perimeter squared to the units of filled area (denoted $\mathcal{P}$)
- Ratio of perimeter to the normalized "bounding rectangle" perimeter (the perimeter of the bound divided by the area of the bound) (denoted $\mathcal{B}$)

These measures were implemented with the understanding that they might describe the intuitive notion of "shape complexity" while using only simple geometric attributes. The first two measures: the ratio of the perimeter of the shape to the number of filled cells and to the perimeter of a "bounding rectangle" are the two most obvious geometric measures that would be likely to be related to perceived complexity. The other two measures can be understood as modifications of the first two that are "normalized" to the non-perimeter value. In the case of (perimeter/filled area), we note that area does not scale linearly with perimeter — rather, it scales linearly (or close to linearly) with the square of perimeter, and we square the perimeter value accordingly to get $\mathcal{P}$. In the case of (perimeter/perimeter of bounding rectangle) we simply incorporate the filled area of the rectangle into the measure (with some understanding that two shapes identical other than scale, should have identical complexity), resulting in $\mathcal{B}$.

5.3 Comparison of Implemented Measurements as Measures of Shape Complexity

In order to assess the applicability of the implemented measurements as measures of shape complexity, we compare computed rankings of a known set of inputs to a qualitative ranking. This comparison is shown in Table 5.

Rank	Std	Entropy	Perim/Area	Perim/Bounding Rect	$\mathcal{P}$	$\mathcal{B}$
0		0.2028	0.3102	 1.0952	 16.2	 70.875*
1		 0.4088	 0.45	 1.1111	 19.0588	 70.875*
2		 0.4146	 0.5294	 1.1176	 21.9661	 70.875*
3		 0.4155	 0.6101	1.1206	 30.0833	 73.7647
4		 0.4215	 0.6666	 1.125*	 33.2830	 82.6875
5		 0.4326	 0.7142	 1.125*	 35.2666	 90.5625
6		 0.4379	 0.7666	 1.125*	40.3341	120.4761
7		0.4452	0.7916	1.2962	42.32	135.0
8		0.4488	0.7924	1.3125	42.8571	172.8571
9		0.4550	0.92	1.4375	46.6666	188.8888
10		0.4830	0.9705	1.5714	61.0169	235.9259
11		0.5179	1.0169	1.6666	64.0588	537.9310

Table 5: Rankings of a standard set of inputs by all five candidate geometric measures (along with "std" — human-ranked inputs). Note that starred (*) values are not rounded.

Notably, the measure that performs most similarly to the qualitative ranking is $\mathcal{P}$, the "normalized" ratio between shape perimeter and filled area. Not only does this measure outperform the others in terms of the correspondence of rankings to human rankings, but the "complexity scores" given by the measure

vary proportionally to intuitive understandings of increases in complexity. We see that $\mathcal{B}$, the "normalized" bounding-box measure also performs quite well, and while it tends to rank shapes differently than both the given qualitative ranking and the perimeter measure, it clearly behaves better than any of the remaining two candidate measures.

6 ANN Designs and Features for Testing

With a sense that these two geometric "ranking" functions may function as a reasonable proxy for geometric complexity (or more generally, a qualitative sense of "interestingness"), the focus shifts to how those functions behave in a system like EvoFab (in simulation). In particular, we begin to be able to develop a context for how to best test the fitness of printer-directing ANNs.

In order to do this, we define fixed ANN topologies based on a set of significant network "features". With populations of ANNs restricted to those fixed network topologies, we then optimize those networks using the simulated EvoFab GA system to maximize the fitness of the networks for each of the two promising geometric measurements: $\mathcal{P}$ and $\mathcal{B}$.

6.1 Significant Structural Characteristics of ANNs for 3d-Printer Control

The fundamental goal of this work is to improve our understanding of what "kinds" of ANNs are able to produce complex geometric results when controlling an EvoFab-like 3d-printer system. In order to do this, we identify a number of "features" of ANNs which are likely to have influence on the fitness of those networks.

It is important to note that this work is focused on very simple neural networks for use as control systems. Much current work using neural networks for complex problems is focused on the use of many-layered networks with complex structure. However, as simple neural networks have been successfully used as control systems for interesting problems in Evolutionary Robotics [4][10], we hypothesize that simple, 3-layer, feed-forward, sigmoid-activated ANNs are a reasonable platform for investigative work on their use as control systems for Evolutionary Fabrication with 3d-printers.

Furthermore, these simple ANNs are trivial to encode as genotypes for use with Genetic Algorithms. Since we evolve populations of ANNs with a shared structural topology, we are able to fully describe a member of that population by an ordered list of weights corresponding to each of the (ordered) connections present between nodes in that network.

With these simple networks as a platform for our work, we identify the following as features that are

likely to impact the behavior of ANNs in our system.

1. The number of *hidden nodes* in an ANN

   Increasing the number of hidden nodes in an ANN increases the number of connections and paths between the input nodes of the network and the output nodes. In doing this, it increases the interpretive capabilities of that ANN.

2. Whether or not the ANN has *recurrence*

   An ANN is *recurrent*, if there are any cycles between layers of the network. Intuitively, recurrence can be viewed as a certain amount of "memory", or "state" within a network. Where a network without recurrence produces an output based only on the current external inputs to the network, a network with recurrence produces outputs based both on current sensor inputs, and on some previous input or output to or from the network.

3. Whether or not the ANN has an understanding of the "global" position of the printer on the build platform

   In the EvoFab 0.3 system, the ANN directing printer motion has inputs based on local sensor data around the extruder head. That is, the ANN receives input based on the state of the world locally around the extruder, but has no information about the absolute position of the extruder relative to the build platform. Adding inputs to the ANN based on this global position could potentially aid in the production of interesting/complex geometric outputs.

6.2 Encoding of "Global Position" Information

Since our chosen ANNs have sigmoid activation functions, and few inputs, it is not immediately obvious how to introduce global position data as inputs to the neural network. This sigmoid activation function is well suited to the simulated sensor inputs to the network, which simply represent "percent fill" of the area under that portion of the sensor grid. As percent fill increases or decreases, the sigmoid function crosses a critical value and "activates". In contrast, global (x, y) position is not likely to be useful simply as raw numeric inputs to the ANN.

A reasonably obvious solution to this problem is to encode the x and y coordinate each as a $log_2(n)$ bit graycode value, where n is the dimension of the (square) build platform. Graycode is a binary encoding of integers where each increment from m to $m + 1$ is performed by changing a single bit. This is in contrast to standard binary encodings, where the difference in representation from m to $m+1$ may require a significant number of flipped bits (for example, from `01111` to `11111`). Each of the $log_2(n)$ bits would then be given

as a binary input to the ANN. The ANN has, therefore, a full-precision measure of its position on the build platform.

The primary concern with this implementation is the number of additional inputs that would be added to the network. While the most basic network in use has 9 inputs, that same network with the addition of graycode global coordinates would have $9 + 2log_2(n)$ inputs, where n would be on the order of 100. This would increase the number of inputs to the neural network in our case, but would likely preform fairly well since it would only add around 14 inputs to the network. However, this implementation is not likely to scale well to either a larger build plate, or a physical printer system. In the raster EvoFab simulation, a 100×100 "raster unit" build plate represents a large build space. However, since the movement of the physical printer is significantly more precise, the coordinate system used for global coordinate inputs to the ANN would likely have to be significantly more precise (and therefore would require many more inputs). This would cause a significant increase in the number of inputs to the network with the addition of global coordinate information.

To avoid this problem, we propose a minimal design for the introduction of a "smooth" representation of global coordinate data as input to an ANN. This implementation fixes the number of inputs added to the neural network. The values given to these m inputs range from 0 to n/m (where as above, n is the dimension of the build space). These inputs "accumulate" the value of the current x or y coordinate. So if there are $m = 4$ input nodes, and the current x coordinate of the print head is $\frac{3}{4}n + 2$, the inputs would have values $\frac{n}{4}, \frac{n}{4}, \frac{n}{4}, 2$ respectively.

This implementation for global coordinate information both avoids the addition of a very large number of inputs nodes, and should scale appropriately. In addition, our choice of a relatively small number of additional input nodes for each coordinate value represents how the system will likely behave at scale when operating in a higher-resolution physical system.

6.3 Simple Recurrence for Simple ANNs

Just as our approach to an implementation of global coordinate information for ANNs focuses on maintaining the structural simplicity of those networks, our implementation of recurrence represents the simplest possible introduction of "state" into our networks. Our implementation adds a given number of inputs to the network, where these inputs give the previous n inputs or outputs to or from the network. For testing, we focus on recurrence from outputs of the network (printer movement instructions) to printer inputs.

String	Instruction
000	None
001	None
011	None
010	None
110	North
111	South
101	East
100	West

Table 6: Initial encoding for ANN printer instructions

String	Instruction
000	None
001	North
011	None
010	South
110	None
111	East
101	None
100	West

Table 7: Encoding for ANN printer instructions (modified)

6.4 Printer Instruction Encoding

During exploratory testing, outputs of the ANN were encoded as graycode values corresponding to motor instructions (Table 6). However, early testing showed that many networks which were bred during evolution were likely to produce "No Motion" behavior. While not inherently bad for 3d-printer control, the frequency of this behavior significantly slowed the process of evolution, and served as a barrier to the discovery of "interesting" output shapes. This problem was first attributed to the clustering of "No Motion" instructions in the graycode table. However, it did not appear that distribution of the "No Motion" instructions as shown in Table 7 improved this behavior.

To minimize the impact of these "No Motion" networks on our testing, we prohibit networks from giving "No Motion" commands to the printer. For all discussed testing we use the following simple encoding for printer instructions. Each of 4 outputs from the ANN represents a cardinal direction. At each time step, the direction instruction given to the printer corresponds to the direction associated with the output node with the largest floating point value.

In addition to eliminating the potential for "No Motion" instructions from the ANN, this encoding also ensures that there are no "relationships" between the encodings of two cardinal directions. For example, with the encodings given in Table 6, the encoding for "South" and "East" differ only by a single bit. In contrast, the encoding for "North" and "East" differ by two bits. This suggests the possibility that it might be "easier" for ANNs to move towards the southwest corner of the build plate than the northeast corner. This problem is eliminated by encoding cardinal directions relative to single output nodes, rather than to binary strings generated based on all of the output nodes of the network.

7 Testing Configuration

In order to evaluate the effect of the number of hidden nodes, recurrence, and the presence of global coordinate information on the ability of ANNs to produce "interesting" shapes, we select 12 fixed ANN topologies, and perform evolutionary optimization using the simulated EvoFab system using each of the two candidate geometric complexity proxies as objective functions.

We use Genetic Algorithms to maximize the fitness of each fixed-topology population of ANNs relative to $\mathcal{P}$ and $\mathcal{B}$. Figure 6 diagrams the 24 GA runs performed.

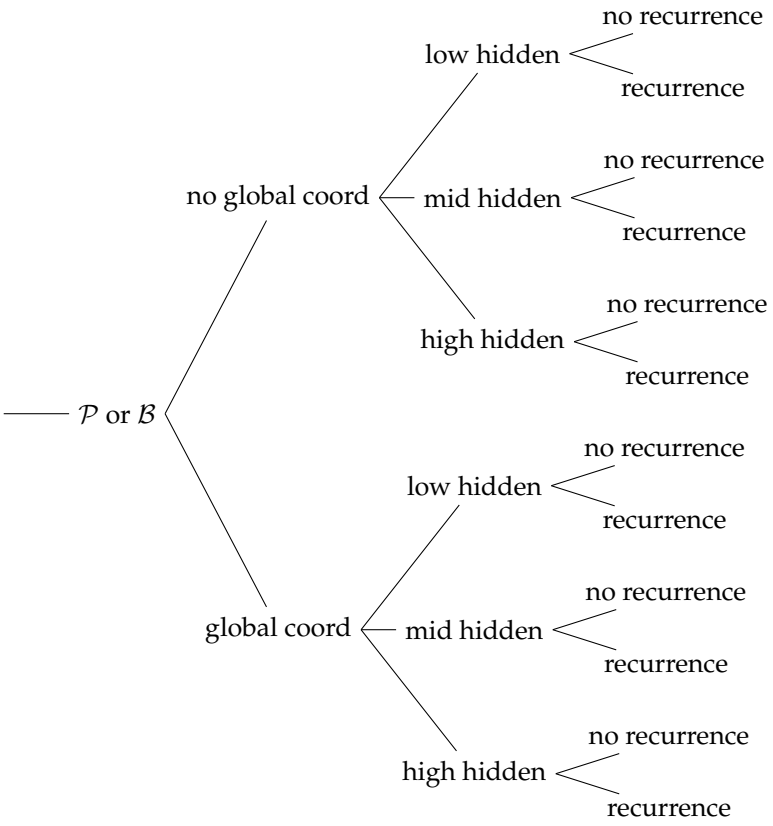


Figure 6: The 24 unique GA optimization runs performed for data collection (for 12 ANN topologies and 2 objective functions — totaling 24 GA runs)

For each unique GA configuration, 6 duplicate runs were performed with different random seeds. There are a number of parameters for the EvoFab optimization system, including parameters for the GA and parameters for the printer simulation itself. These parameters are held constant between runs. A few selected (significant) values are shown in Table 8.

Specific values for ANN topologies (including numbers of hidden nodes, and number of time-steps for recurrence) are given in Table 9.

Parameter	Value
population size	100
number culled	80
mutation rate	0.2
crossover : mutation	$1:1$

Table 8: Selected (significant) parameters for GA runs

Description	Value
High number of hidden nodes	50
Mid number of hidden nodes	25
Low number of hidden nodes	15
Number of inputs for global coords	4
Number of recurrence time steps	5

Table 9: Specific values for tested ANN topologies

8 Results

8.1 Results for Networks Optimized to $\mathcal{P}$

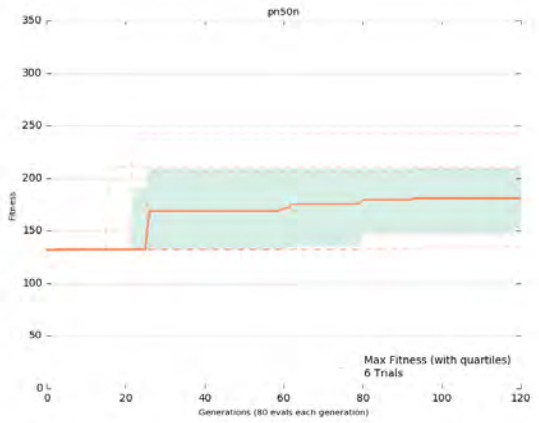
For almost all ANN topologies, GA results for optimization to maximize $\mathcal{P}$ are very similar in terms of maximum fitness. Figures 7a, 7b, and 7c show results for all replicated runs for three very different network topologies.

Notably, while there are noticeable differences in the median values, ranges, and locations of plateaus in fitness in each of these three cases, the highest fitness member of the best duplicate run for each only reaches a fitness of approximately $\mathcal{P}(x) = 250$. The two major exceptions to this are two network topologies with 15-hidden nodes. The first has no global coordinates, 15 hidden nodes, and no recurrence. This network topology only reached a fitness of $\mathcal{P} < 150$ (Figure 8). The second has global coordinates, 15 hidden nodes, and recurrence. This network topology plateaued around $\mathcal{P} = 250$ as expected, with the exception of two outlier runs with most fit members with fitness $\mathcal{P} > 300$ (Figure 9).

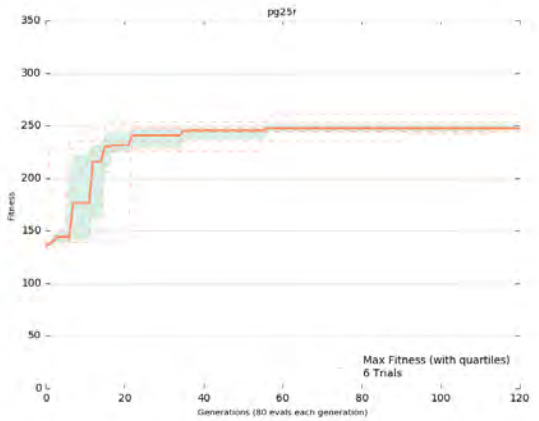
Beyond these major outliers, these results heavily suggest that for all network topologies that have either recurrence or access to global coordinate information, those are optimizable to a fitness of $\mathcal{P} \approx 250$. A fairly typical output from an ANN with fitness around 250 is shown in Figure 11.

Qualitatively, the shape produced by this network is not impermissible "simple". However, it is clearly *very much* less qualitatively interesting than those shapes shown in Figure 9. For this reason, it is likely significant that effectively all optimization runs with all ANN topologies plateaued around $\mathcal{P} = 250$ rather early on, and struggled to surpass that fitness.

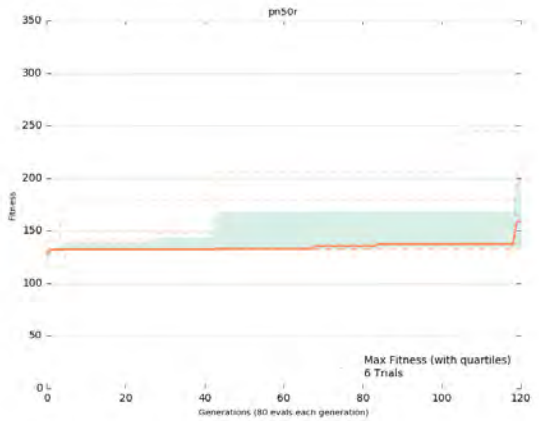
We also note that while the existence (or perhaps more noticeably, absence) of global coordinate information and/or recurrence impacted the ability of ANNs to be optimized to maximize $\mathcal{P}$, an increase in



(a) No global coordinates. 50 hidden nodes. No recurrence.



(b) Global coordinates. 25 hidden nodes. Recurrence.



(c) No global coordinates. 50 hidden nodes. Recurrence.

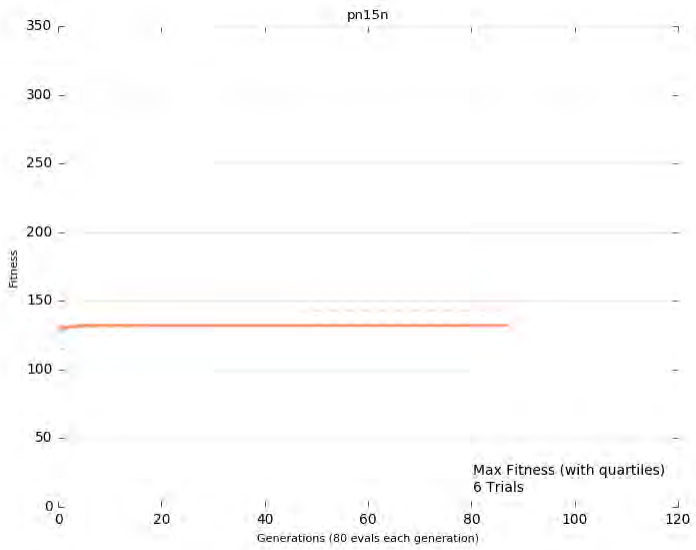
Figure 7: Fitness of maximum-fitness ($\mathcal{P}$) individual in population over time (generations). Median for all duplicate runs given in bold orange. Individual maximums for each run given in dotted orange. Green shading shows the upper and lower quartile spread across duplicate runs for maximum fitness.

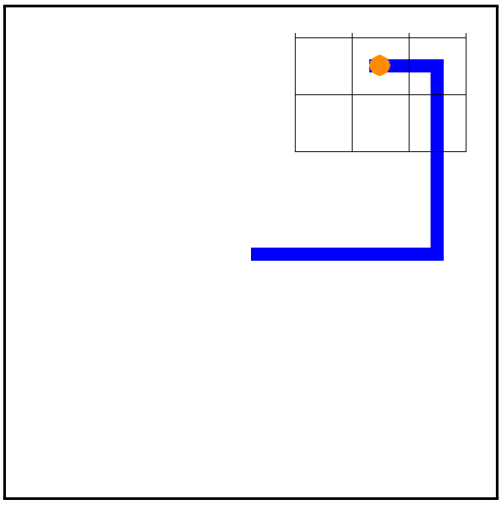
hidden nodes in the network did not have as clear of an impact. We can see this by comparing fitnesses over time with recurrent networks with global coordinate information with 15, 25, and 50 hidden nodes. These networks are shown in Figures 9, 7b, and 10, respectively.

8.2 Results for Networks Optimized to $\mathcal{B}$

While results of of the optimization of fixed ANN topologies to maximize $\mathcal{P}$ do provide some illumination as to which network characteristics contribute to the "interestingness" of the shapes that those networks can produce, results from optimization to maximize $\mathcal{B}$ provide significantly more information.

Table 10 shows maximum fitnesses of evolved members for each fixed ANN topology (optimized to





(a) Highest fitness member of all duplicate runs. $\mathcal{P} = 142.785$

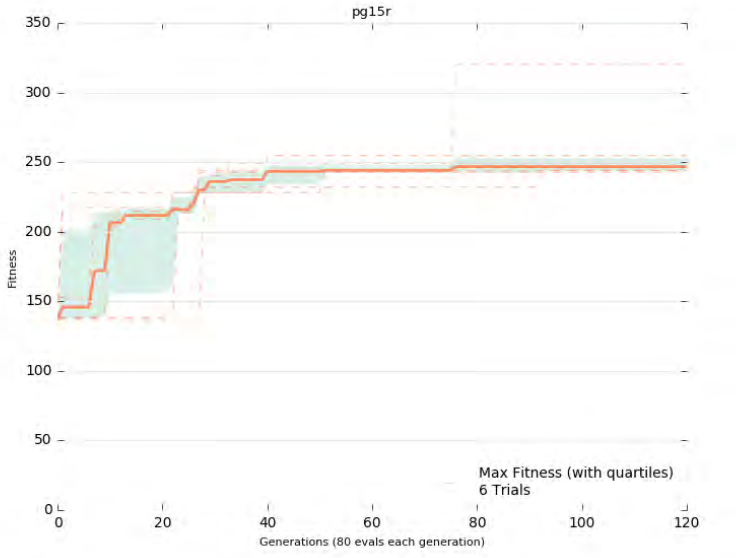
Figure 8: No global coordinates. 15 hidden nodes. No recurrence.

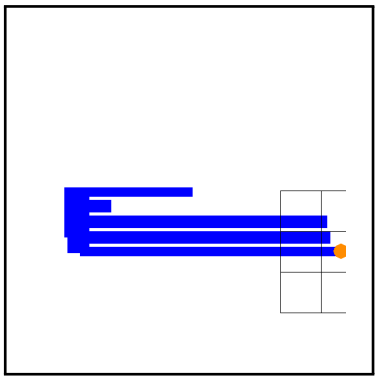
maximize $\mathcal{B}$). There is a noticeable difference in the maximum fitnesses for ANNs with access to global coordinates (which is significantly more noticeable than when the ANNs were optimized to maximize $\mathcal{P}$). This distinction between networks with and without global coordinates is demonstrated in Figures 12a and 12b.

Network Topology	Maximum Fitness
No global, 15 hidden, no recur	4415.081
No global, 15 hidden, recur	4449.599
No global, 25 hidden, no recur	5747.080
No global, 25 hidden, recur	4449.6
No global, 50 hidden, no recur	5327.882
No global, 50 hidden, recur	5357.647
Global, 15 hidden, no recur	6064.0
Global, 15 hidden, recur	11270.769
Global, 25 hidden, no recur	9917.157
Global, 25 hidden, recur	8360.232
Global, 50 hidden, no recur	9446.068
Global, 50 hidden, recur	9282.987

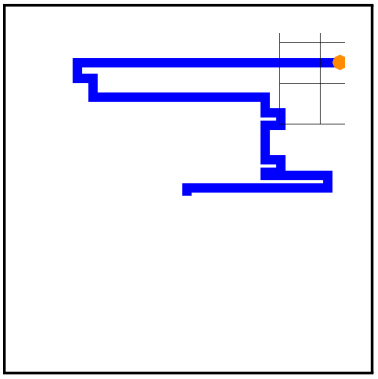
Table 10: Maximum fitness $\mathcal{B}$ (for all duplicate runs) for each network topology.

These figures show behavior typical to network topologies with and without access to global coordinate information when optimized to maximize $\mathcal{B}$. In particular, runs with network topologies that do not have access to global coordinates tend to give median and quartile boundaries that plateau with fitness significantly less than $\mathcal{B} = 6000$. In contrast, network topologies with access to global coordinates tend to plateau at fitness $\mathcal{B} \approx 6000$. In addition, this plateau tends to occur very early (in the first few generations).





(a) Second highest fitness member of all duplicate runs. $\mathcal{P} = 320.112$



(b) Highest fitness member of all duplicate runs. $\mathcal{P} = 359.111$

Figure 9: Global coordinates. 15 hidden nodes. Recurrence.

In order to understand the behavior of $\mathcal{B}$ as a fitness function, we can look at typical geometric outputs with values in these two ranges. Figure 13 shows typical outputs from ANNs with fitnesses near 6000 and below 5000.

While these two typical outputs are not drastically different in terms of their physical shape, they do appear to involve different mechanisms of printer control. In Figure 13b, the printer appears to start in a given direction, and then trace along the outside of the build platform along the first corner it encounters. In contrast, Figure 13a appears to show an ANN which controls the printer to create a shape beyond this simple edge-tracing. Furthermore, this ANN is able to produce multiple complex corners, and resulting has a shape that does appear more qualitatively "interesting".

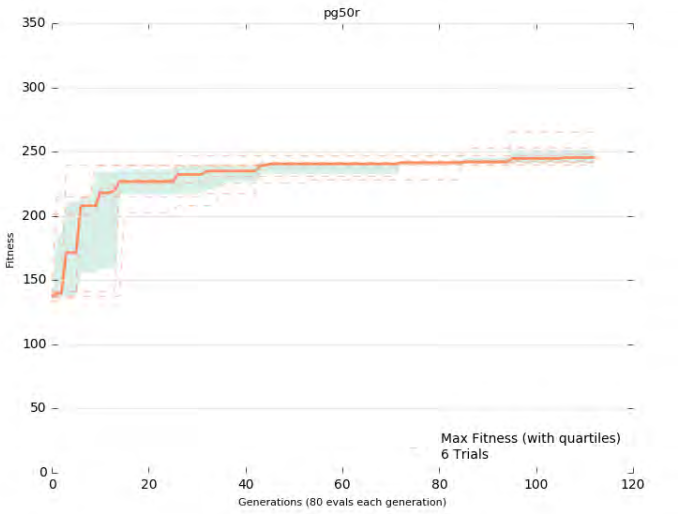


Figure 10: Global coordinates. 50 hidden nodes. Recurrence.

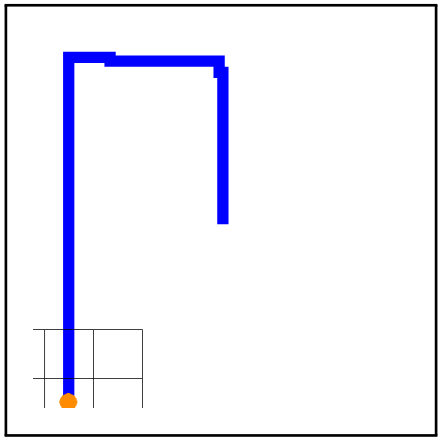
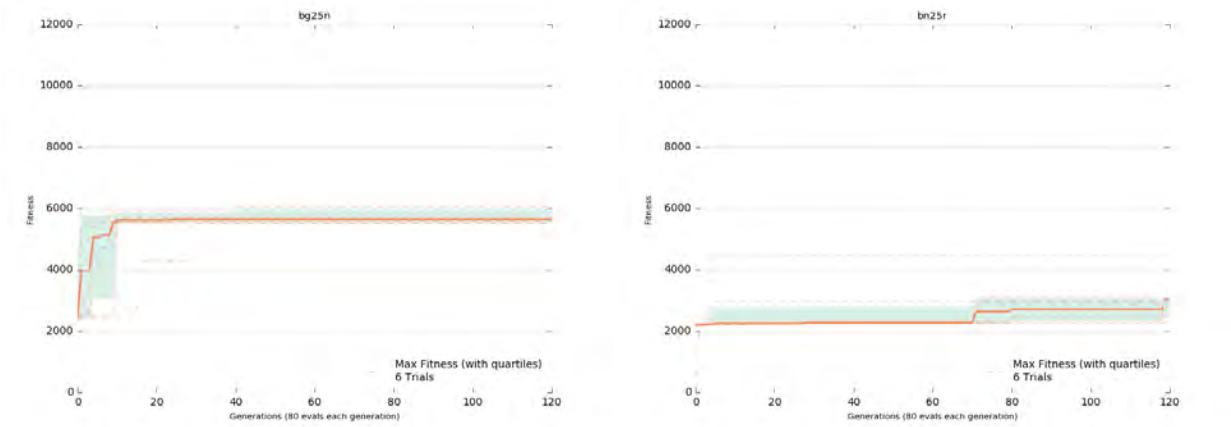


Figure 11: Output from an ANN with fitness of $\mathcal{P} = 244.0168$

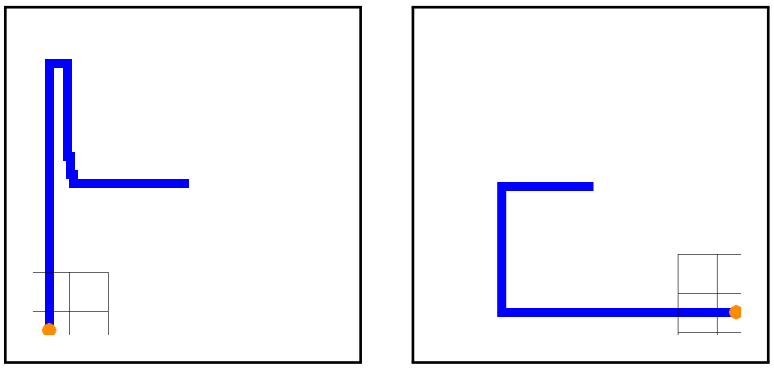
Perhaps even more significant than these general trends between network topologies with and without access to global coordinate information is the existence of individual, very high fitness outliers discovered when optimizing ANNs to maximize $\mathcal{B}$. These outliers were present in 5 of the 6 network topologies with access to global coordinate information. Figure 14 shows the outputs produced by these high fitness individuals.

In particular, we note that these designs are much more qualitatively "interesting" when compared with many of the outputs produced under optimization. Furthermore, we note that these members also have high fitness when evaluated under $\mathcal{P}$, despite the fact that they were evolved under $\mathcal{B}$. Additionally, the outputs shown in Figures 14a and 14b have both the highest fitnesses relative to $\mathcal{B}$ and the highest fitnesses relative to $\mathcal{P}$ of all members, all ANN topologies, in all runs optimized to either $\mathcal{P}$ or $\mathcal{B}$.



(a) Global coordinates. 25 hidden nodes. No recurrence. (b) No global coordinates. 25 hidden nodes. Recurrence.

Figure 12: Fitness of maximum-fitness ($\mathcal{B}$) individual in population over time (generations). Median for all duplicate runs given in bold orange. Individual maximums for each run given in dotted orange. Green shading shows the upper and lower quartile spread across duplicate runs for maximum fitness.



(a) Typical output from ANN with fitness $\mathcal{B} = 5694.928$

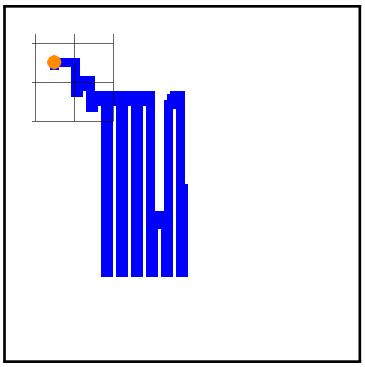
(b) Typical output from ANN with fitness $\mathcal{B} = 4357.41935484$

Figure 13

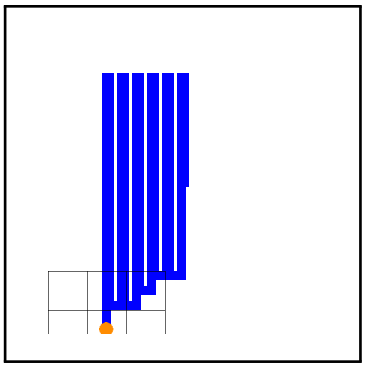
Notably, these high fitness outliers were not present in *any* duplicate runs for any ANN topologies that did not have global coordinate access.

Finally, we see that ANN topologies with many hidden nodes do not seem to have an advantage in the discovery of high fitness individuals. Figures 15a and 15b show optimization of network topologies with global coordinates and recurrence with 15 and 50 hidden nodes, respectively.

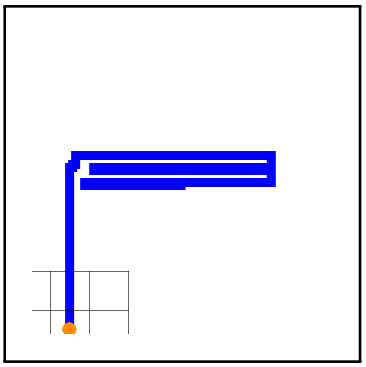
We see that both network topologies show plateaus between $\mathcal{B} = 4000$ and $\mathcal{B} = 6000$. However, while a high-fitness outlier is discovered around generation 45 in the 15 hidden node runs, discovery of a high fitness outlier in the 50 hidden node runs occurs around generation 75. Furthermore, with 50 hidden nodes, it takes until around generation 25 for a particularly low-fitness outlier (around $\mathcal{B} = 2700$) to converge with the rest of the runs around $\mathcal{B} = 5000$.



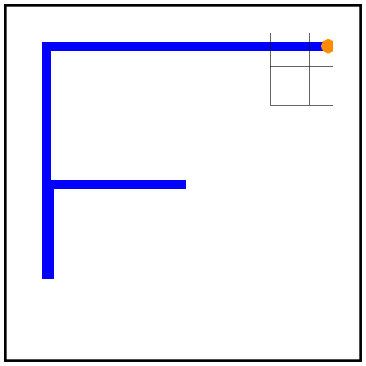
(a) 15 hidden nodes. Recurrence. $\mathcal{B} = 11270.769$, $\mathcal{P} = 435.059$



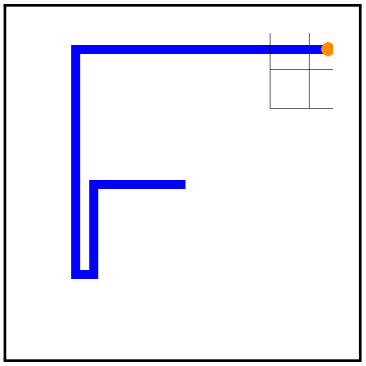
(b) 25 hidden nodes. No recurrence. $\mathcal{B} = 9917.157$, $\mathcal{P} = 491.483$



(c) 25 hidden nodes. Recurrence. $\mathcal{B} = 8360.232$, $\mathcal{P} = 312.720$



(d) 50 hidden nodes. No recurrence. $\mathcal{B} = 9446.068$, $\mathcal{P} = 281.756$



(e) 50 hidden nodes. Recurrence. $\mathcal{B} = 9282.988$, $\mathcal{P} = 306.680$

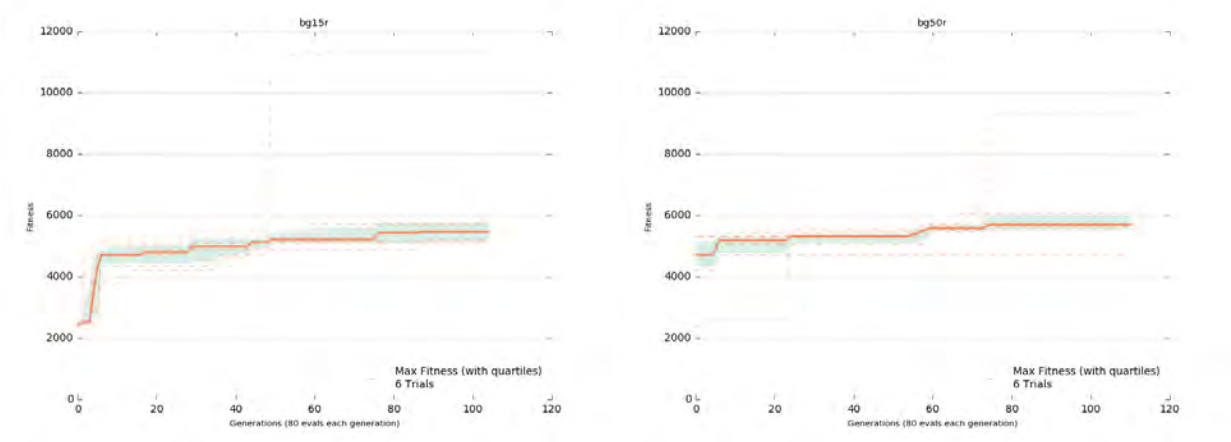
Figure 14: Outputs from high-fitness outliers with global coordinates optimized to maximize $\mathcal{B}$

9 Discussion

9.1 On "Shape Complexity"

Both $\mathcal{P}$ and $\mathcal{B}$ were designed to serve as proxies for "shape complexity" as described by the entropy of curvature of a 2d-shape. These two measures behaved well in testing when compared to a canonical qualitative ranking of shapes, and seemed to correspond to intuitive notions of the behavior of "shape complexity". However, using these functions as fitness functions for optimization with Genetic Algorithms revealed major edge cases that did not perform as well as expected. In particular, these measures appear to heavily favor large open shapes, where the length of the path of that shape is valued heavily, and complex internal geometry is sometimes left behind. This is illustrated well by the behavior of $\mathcal{P}$ on angled trajectories (Figure 16). While these fitness functions do, then, value "large" shapes with complex geometries higher than simple "large" shapes, this discovery of "largeness" may account for some of the major plateauing of fitnesses among competent ANNs for both $\mathcal{P}$ and $\mathcal{B}$.

Fundamentally, this means that neither $\mathcal{P}$ nor $\mathcal{B}$ should be viewed as perfect proxies for "shape complex-



(a) 15 hidden nodes

(b) 50 hidden nodes

Figure 15: Optimization of ANN topologies with global coordinates and recurrence to maximize $\mathcal{B}$

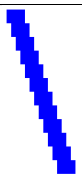
Input	Image	$\mathcal{P}$
0		30.083333333333332)
1		64.05882352941177
2		72.25

Figure 16: Normalized perim/area measurement on long angled input relative to others

ity". That being said, the fact that there were a number of outlying individuals in both runs performed to maximize $\mathcal{P}$ and to maximize $\mathcal{B}$ with both high fitness and high qualitative "interestingness", suggests that these measurements are still somewhat useful for understanding the ability of ANN topologies to produce complex geometric outputs.

9.2 On "Usefulness" of ANNs as Genotypes for Evolutionary Design with 3d-Printers

Despite the fact that $\mathcal{P}$ and $\mathcal{B}$ are imperfect as fitness functions to maximize "interestingness", the existence of high "interestingness" individuals as optimized outputs suggests that we can still make some conclusions about the fitness of various ANN topologies as genotypes for evolutionary design with 3d-printers.

Based on results from testing to optimize $\mathcal{P}$, we cannot make many definitive conclusions. However, it is clear that networks that have neither global coordinate information nor recurrence did not perform as

well as networks that had either. In addition, the presence of additional hidden nodes did not seem to make an impact on the ability of a network to produce outputs with high $\mathcal{P}$.

Results from testing to optimize $\mathcal{B}$ are more illuminating. It is clear that access to global coordinate information is a significant factor in the ability of a network topology to produce high $\mathcal{B}$ outputs. No network topology without global coordinates produced a network with fitness $\mathcal{B} > 6000$. In contrast, all but one of the network topologies with global coordinates showed high-fitness outliers with fitness $\mathcal{B} > 8000$.

Interestingly, the presence of large numbers of hidden nodes in network topologies optimized to maximize $\mathcal{B}$ did not seem to aid in the discovery of high fitness individuals. This is not surprising if we remember that the process of optimization is a search over an n-dimensional space of solutions, where each additional node in the neural network introduces many additional weights, and significantly increases the size of that search space. In this context, it is logical that simply adding hidden nodes to a network topology does not make discovery of "interesting" shapes easier. This is particularly true given that low-hidden node ANN topologies with global coordinates and recurrence have been shown to be able to produce high-fitness individuals.

From these results, we conclude that global coordinate information is likely essential for an ANN to be able to produce useful shapes in the EvoFab system. We further conclude that the addition of recurrence to the network may have a positive impact on the usefulness of that network, although it may slightly increase the dimensionality of the solution space. Finally, we conclude that relatively low numbers of hidden nodes are sufficient to produce fairly complex geometric outputs in this system, and that future work to use EvoFab for evolutionary design should (at least begin) by working with these low-hidden-node networks.

10 Future Work

Because of the scope of this project, there are a number of questions that remain unanswered and work that could be easily expanded on. Because of difficulty adapting understandings of the entropy of curvature to the EvoFab simulation, we rely on (imperfect) proxies for shape complexity as objective functions for optimization. These objective functions were shown to both be imperfect as proxies for shape complexity and imperfect as fitness functions for use with Genetic Algorithms (because they encouraged major plateaus at overvalued, relatively uninteresting shapes). It is possible that there is an effective adaption of entropy to the EvoFab simulation which would serve as a better objective function for optimization.

Furthermore, our set of fixed network topologies is relatively limited. Additional ANN "features" (including more sophisticated activation functions) could potentially lead to interesting results. Continued work could also investigate to what extent increasing sensor resolution, increasing the resolution of global

coordinate information, increasing the number of recurrent time steps, and changing the type of recurrence implemented impacts geometric outputs from those ANNs.

A much larger number of duplicate GA runs for each fixed network topology, along with a higher number of generations, would allow for statistical analysis comparing the results of optimization with different network topologies. In addition, similar analysis on the physical EvoFab system would serve to confirm whether results in simulation generalize to results in the physical hardware.

More generally, these results suggests a set of reasonable ANN topologies for use as evolved fabrication procedures for evolutionary fabrication and design using EvoFab. Future work will involve work to further validate the usefulness of ANNs as control systems for fabrication and design.

References

[1] Joshua E. Auerbach and Josh C. Bongard. "Environmental Influence on the Evolution of Morphological Complexity in Machines". In: *PLoS Comput Biol* 10.1 (Jan. 2014), pp. 1–17. DOI: 10.1371/journal.pcbi.1003399. URL: http://dx.doi.org/10.1371%2Fjournal.pcbi.1003399.

[2] James Cohoon, John Karro, and Jens Lienig. "Advances in Evolutionary Computing". In: ed. by Ashish Ghosh and Shigeyoshi Tsutsui. New York, NY, USA: Springer-Verlag New York, Inc., 2003. Chap. Evolutionary Algorithms for the Physical Design of VLSI Circuits, pp. 683–711. ISBN: 3-540-43330-9. URL: http://dl.acm.org/citation.cfm?id=903758.903786.

[3] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. A Wiley-Interscience publication. Wiley, 2006. ISBN: 9780471748816. URL: https://books.google.com/books?id=EuhBluW31hsC.

[4] D. Floreano and F. Mondada. "Evolution of homing navigation in a real mobile robot". In: (). URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.2068.

[5] Pablo Jose Funes. "Buildable Evolution". In: *SIGEVOlution* 2.3 (Sept. 2007), pp. 6–19. ISSN: 1931-8499. DOI: 10.1145/1366914.1366916. URL: http://doi.acm.org/10.1145/1366914.1366916.

[6] Andreas F. Koschan et al. "Towards understanding what makes 3D objects appear simple or complex". In: *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* 00.undefined (2008), pp. 1–8. DOI: doi.ieeecomputersociety.org/10.1109/CVPRW.2008.4562975.

[7] Tim Kuehn and John Rieffel. "Automatically Designing and Printing 3-D Objects with EvoFab 0.2". In: *Proceedings of the 13th International Conference on the Synthesis and Simulation of Living Systems (ALife XIII)*. 2012, pp. 372–378.

[8] J. D. Lohn, G. S. Hornby, and D. S. Linden. "An Evolved Antenna for Deployment on NASA's Space Technology 5 Mission". In: *Genetic Programming Theory and Practice II*. Ed. by U.-M. O'Reilly et al. Kluwer, 2005. Chap. 18.

[9] D. L. Page et al. "Shape analysis algorithm based on information theory". In: *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*. Vol. 1. Sept. 2003, I-229-32 vol.1. DOI: 10.1109/ICIP.2003.1246940.

[10] Dean A. Pomerleau. "Advances in Neural Information Processing Systems 1". In: ed. by David S. Touretzky. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989. Chap. ALVINN: An Autonomous Land Vehicle in a Neural Network, pp. 305–313. ISBN: 1-558-60015-9. URL: http://dl.acm.org/citation.cfm?id=89851.89891.

[11] John Rieffel. “Evolutionary Fabrication: the co-evolution of form and formation”. PhD thesis. Brandeis University, 2006.

[12] John Rieffel and Dave Sayles. “EvoFab: a fully embodied evolutionary fabricator”. In: *Proceedings of the 9th international conference on Evolvable systems: from biology to hardware*. ICES'10. York, UK: Springer-Verlag, 2010, pp. 372–380. ISBN: 3-642-15322-4, 978-3-642-15322-8. URL: http://dl.acm.org/citation.cfm?id=1885332.1885373.

[13] *Velleman k8200 Printer*. http://www.k8200.eu/. accessed: 2016-05-02.