Facilitated Variation on Increasingly Complex Boolean Functions

Victor Goncalves[1] **and Emeka Ezike**[2]

[1]Harvard John A. Paulson SEAS, victorgoncalves@college.harvard.edu
[2]Harvard John A. Paulson SEAS, eezike@college.harvard.edu

ABSTRACT

This paper discusses the theory of facilitated variation (FV), which proposes that living organisms consist of highly conserved modules that perform core functions and that genetic mutations can affect the regulatory mechanisms that control these modules, leading to the emergence of new traits. In *Facilitated Variation: How Evolution Learns from Past Environments To Generalize to New Environments*, Parter et al. use computer simulations to investigate how FV can emerge during evolution, focusing on the effects of varying environments on FV. However, this paper argues that the limitations of their study, such as its reliance on simplistic boolean logic circuits, resulted in ideal results that would not persist as complexity increases. Thus, this paper aims to critically evaluate the results of Parter et al. by replicating their experiments for boolean logic gates on more complex goals to provide a more comprehensive understanding of the evolution of FV and the role of modularity in this process.

Keywords: facilitated variation, combinatorial logic circuits, evolution

1 Introduction

"How can small, random genetic changes be converted into complex useful innovations?[1]" This question has puzzled biologists since the publication of Charles Darwin's *On the Origin of Species* in 1859. To understand how novel, useful phenotypes arise in evolution, Kirschner and Gerhart integrated observations on molecular mechanisms to show how the current design of an organism helps to determine the nature and the degree of future variation. Their key observation is that the organism seems to be built in such a way that small genetic mutations have a high chance of yielding a large phenotypic payoff. Kirschner and Gerhart compiled these findings and their related conclusions and then published them in *The Theory of Facilitated Variation*.

The theory of facilitated variation (FV), then, suggests that complex biological systems can be generated by a few regulatory genetic changes and the differential reuse of pre-existing developmental components. Living organisms consist of a set of modules that are highly conserved, referred to as "core processes," which perform functions related to development and physiology and have remained mostly unchanged for millions years. When there are genetic mutations, they can affect the regulatory mechanisms that control the core components of an organism, causing changes in their combinations, amounts, and functional states. These altered combinations of core components work together to generate new traits, which can be acted upon by natural selection. A commonly provided example of this process is the evolution of a bird wing. A small number of genetic alterations trigger changes in the length and thickness of bones, and other aspects of limb development, such as the muscles, nerves, and vasculature, adapt to accommodate these changes without requiring independent regulatory changes (see Figure 1). Due to their modular structure, adaptability, and compartmentalization, developmental systems tend to create facilitated phenotypic variation that is functional and adaptive when faced with genetic mutation or new environmental conditions.

In *Facilitated Variation: How Evolution Learns from Past Environments To Generalize to New Environments*, Merav Parter, Nadav Kashtan, and Uri Alon investigate how FV spontaneously emerges during evolution by using computer simulations of two well-studied model systems: logic circuits and RNA secondary structure. They find that evolution of FV is enhanced in environments that change from time to time in a systematic way and are made of the same set of subgoals but in different combinations. Organisms that evolve under such varying goals not only remember their history but also generalize to future environments, exhibiting high adaptability to novel goals. The authors also found that elements of FV theory, such as weak regulatory linkage, modularity, and reduced pleiotropy of mutations, evolve spontaneously under these conditions. The study suggests that environments that change in a systematic, modular fashion promote FV and allow evolution to generalize to previously unseen conditions.

While the study by Parter et al. provides interesting insights into how FV can emerge during evolution, we believe that there are several limitations to the research. Firstly, the study relies solely on computer simulations of two model systems, which may not accurately reflect the complexities of biological systems. Additionally, the study only examines the effects of systematically varying environments on FV, and does not consider other factors that may influence the emergence of FV, such as behavioral traits and reproductive strategies. Nonetheless, the study's most significant drawback is the use of boolean logic circuits to support FV, which are too simplistic. Randomly guessing 0 and 1 would yield an expected accuracy of 50% for any arbitrarily-selected logic gate, undermining the reliability of the findings. Consequently, this paper aims to critically evaluate the results of Parter et al. by replicating their experiments for boolean logic gates but with more complex goals. By doing so, we hope to provide a more comprehensive understanding of the evolution of FV and the role of modularity in this process. Our hypothesis is that the modularity observed in the context of FV, as measured and observed in *Facilitated Variation: How Evolution Learns from Past Environments To Generalize to New Environments*, is unlikely to manifest in functions that are more complex. This is because boolean logic functions, being relatively simple, are vulnerable to common errors that arise in reinforcement learning, such as becoming trapped in local maxima. Therefore, while the findings of Parter et al. provide a useful starting point for investigating the emergence of FV in more complex systems, we believe our additional research is needed to determine the extent to which FV can be observed in these systems and the factors that may influence its emergence.

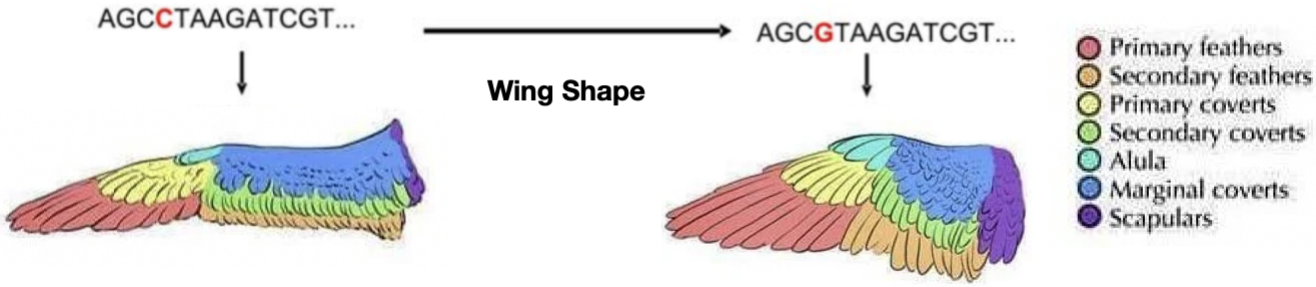


Figure 1. Weak Regulatory Linkage: Gerhart and Kirschner use the example of how a bird or bat wing evolved from a tetrapod forelimb to explain how relatively small genetic changes can lead to the development of new traits. They argue that changes in the length and thickness of bones can result from genetic mutations, and other aspects of limb development, such as the muscles, nerves, and vasculature, can accommodate these changes without requiring independent regulatory changes. The adaptability of muscle precursors enables them to receive signals from developing dermis and bone, and to take up positions relative to them. As the nerve cord extends axons into the developing limb bud, some of them contact muscle targets and stabilize, while others shrink back. Finally, vascular progenitors enter the developing limb, and wherever the limb cells are hypoxic, they secrete signals that trigger nearby blood vessels to grow into their vicinity. This adaptability means that the co-evolution of bones, muscles, nerves, and blood vessels is not required, and that large phenotypic changes can be favored by selection. The result is that viable phenotypes can easily be generated with little genetic change, and that genetic mutations are less likely to be lethal. Moreover, the phenotypic variation generated in this manner is functional and adaptive, which is why it is called 'facilitated' variation.

2 Literature Review

2.1 Evolving Logic Circuits

Our research is built upon prior literature and established frameworks. While obviously heavily inspired by Parter et al.'s *Facilitated Variation: How Evolution Learns from Past Environments To Generalize to New Environments*, earlier research by Nadav Kashton and Uri Alon also contributed to our study. In their paper, *Spontaneous evolution of modularity and network motifs*, Kashton and Alon explored the idea that designs with higher modularity have higher adaptability and survival rates in changing environments. They conducted an experiment where the evolutionary goal changed with time in a modular fashion, switching between different combinations of subgoals. Their experiments involved two evolutionary algorithms: Electronic Circuit Evolution and Neural Network Evolution. The Electronic Circuit Evolution method used circuits represented by binary genomes and evolved towards a fixed goal (FG) or alternated between two goals every 20 generations. **Circuits with too many gates were penalized to prevent over-reliance on "perfect" circuits.** This idea of penalization was ultimately critical to our own research (see 3.4). The Neural Network Evolution method used a fixed size neural network genome with a penalty for adding too many neurons, encouraging the evolution of modular structures. A quantitative measure of modularity and

network motifs were detected using detailed network algorithms (see 3.5). The study compared the outcomes of using FGs versus MVGs in experiments with logic gates and neural networks. The results showed that while FGs produced high success rates, they lacked modularity, whereas MVGs had a 100% success rate with strong modularity and pronounced network motifs. The MVG approach also allowed for rapid adaptation to new goals. Similarly to their next paper, *Facilitated Variation: How Evolution Learns from Past Environments To Generalize to New Environments*, the findings suggest that using MVGs can lead to the development of modular network architectures with significant network motifs

In addition, **the study suggests that the use of MVGs can facilitate the movement of the population away from local fitness maxima and towards a region in network space that contains fitness peaks for each goal in close proximity.** This idea was also critical to our study, as it prompted further experiments to investigate the feasibility of this phenomenon. Evolved modular networks have a denser local structure, which increases the number of subgraphs and network motifs.

3 Methodology

We utilized circuits made up of NAND gates (NOT-AND function) encoded in a binary genome to evolve and satisfy a designated Boolean function G. Our experiments explored 4 and 6-input circuits with the purpose of observing the increasing complexity of 6-input circuits. We employed a standard genetic algorithm (see 3.2) to evolve our circuits. To evaluate FV, we compared the evolution of circuits under FG to circuits evolved under MVG, where goals changed in a modular fashion over time. We also used the Newman and Girvan algorithm (see 3.5) to calculate network modularity.

3.1 Coding a Binary Genome

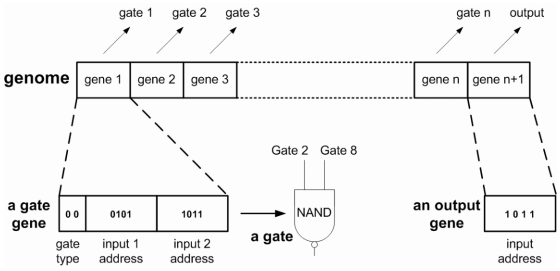


Figure 2. Visualization of the algorithm used to convert a genome into a circuit.

Our circuit encoding utilized a binary genome, which is was designed in a fashion similar to prior literature[1,2]. The 4-input binary encoding is shown in Figure 2, where each gate has a gate type and two input addresses, either from the actual circuit inputs or output of another gate. For clarification, binary interpretations of 0-3 represent input addresses for each of the four inputs. The binary representations of 4-15 represent the input address of the gate that may or may not exist at index $i+4$ in the genome sequence. The gate type is a 2-bit string that encodes a NAND or None typing, as we only allowed for NAND gates, which can construct any logical function or circuit. The None typing allowed circuits to add or drop gates through genomic changes. To extend to 6-input binary encodings, we increased the input gate address size from 4 to 5, allowing for $2^5 = 32$ possible input addresses instead of $2^4 = 16$. We also considered 6-input circuits with 1 or 3 outputs, which required appending the gate address of each output at the end of the genome. The binary genome size B adhered to this specification.

```
B = (GATE_TYPE_SZ + 2 * GATE_ADDR_SZ) * MAX_GATES + (GATE_ADDR_SZ * NUM_OUTPUTS)
```

Therefore, for the different circuits listed above, we have:

Circuit Type	Gate Address Size	Max Gates	B
4-input 1-output	4	12	124
6-input 1-output	5	26	317
6-input 3-output	5	26	327

Table 1. Binary genome sizes for different circuit types.

3.2 Genetic Algorithm

To facilitate the evolution of logic circuits, we followed Mitchell's framework in *An Introduction to Genetic Algorithms* for the implementation of the standard genetic algorithm[3]. For the context of this paper, we focus on their basic definitions for the components of a genetic algorithm including fitness functions, selection methods, and genetic operators.

Mitchell's standard genetic algorithm works as follows:

1. Selection: Parent chromosomes are selected from the current population based on their fitness, with a higher probability of selection for fitter chromosomes. Selection is done "with replacement," meaning that the same chromosome can be selected more than once to become a parent.
2. Crossover: With probability P_c (the "crossover rate"), the algorithm crosses over the pair of parent genomes at a randomly chosen point to form two offspring. If no crossover takes place, the offspring are exact copies of their respective parents. We note that Mitchell believes multi-point crossover versions can be used to allow multiple crossover points; however, we opt not to use this in our implementation.
3. Point Mutation: The algorithm applies the mutation operation to each locus of the two offspring, with probability P_m (the "mutation rate"). A simple point mutation alters one nucleotide in the genome.
4. The resulting chromosomes are then placed in the new population.

The fitness of a circuit is determined by the percentage of correct outputs from a truth table defined by a Boolean function G. For 6-input circuits with 3 outputs, there are $2^6 * 3 = 192$ output combinations (as each input variable can be either 1 or 0). To start, we created a population, N_{pop}, of 5000 individuals, each with a random binary sequence of size B. We experimented with different selection algorithms to evolve the next generation of circuits. One approach we tried was Elitism, where the best-performing L circuits are selected and passed down to the next generation unchanged. For the remaining individuals, we explored two options for choosing their parents. The first was to select the best a circuits (where $a = N_{pop}/2$) and disregard the rest (BEST). Second, we chose parents with replacement with a probability that increases with their fitness (EXP). Given a fitness F_i of an individual i, we chose a parent with a probability $e^{t*Fi}/\sum_i^{N_{\text{pop}}}$ (where t = 30). To generate offspring, we used a crossover with probability $P_{\text{c}} = 0.5$ and a point mutation with probability $P_{\text{m}} = 0.7/B$ (per locus per genome). Our best results came from using Elitism in combination with EXP.

Initially, we set the algorithm's settings (as shown in Table 2) according to the experimental configuration described by Parter et al.[1].

N_{pop}	B	P_{c}	P_{m}	L	G	t
5000	124	.5	.7 / B	N/A	20	30

Table 2. Initial simulation settings for genetic algorithm based on Parter and Kashtan.

Upon conducting our simulations, we found that the use of $N_{\text{pop}} = 5000$ resulted in considerable computational costs. Thus, we identified that a population size of $N_{\text{pop}} = 2000$ yielded improved outcomes while decreasing computation time by an approximate factor of 2.5. Additionally, we identified an appropriate Elitism sub-population size of $L = 500$ through prior literature and empirical analysis[2]. Lastly, we modified the binary genome to incorporate 6-input 3-output functions, resulting in a genome length of $B = 327$. The optimized genetic algorithm settings for the 6-input 3-output problem are summarized in Table 3.

N_{pop}	B	P_{c}	P_{m}	L	G	t
2000	327	.5	.7 / B	500	20	30

Table 3. Modified simulation settings for the genetic algorithm using 6-input 3-output circuits.

3.3 Goal Functions

Parter et al. proposed that modular changes in environments lead to modular changes in goal functions used to simulate the environment in logical circuits, which formed the basis of their contribution to the theory of FV[1]. They designed two circuits, the first using FG evolution to achieve G_1 and the second using MVG evolution with three different goals (G_1-G_3), consisting of two XOR modules that input into a third module implementing the OR function. Each goal had four inputs and one output, and during evolution, the goals would probabilistically switch, with each switch changing a single XOR module to EQ and vice versa. The second circuit showcased a more modular approach to circuit evolution compared to the first circuit. Although the

goals presented during MVG evolution shared the same subgoals, they were arranged in different combinations. For instance, to define G_2, an XOR in G_1 was replaced with an EQ.

Parter et al. based their results on genomes that encoded for 4-input 1-output circuits and were trained on the following goals for MVG[1]:

$$G_1 = (w \text{ XOR } x) \text{ OR } (y \text{ XOR } z)$$
$$G_2 = (w \text{ EQ } x) \text{ OR } (y \text{ XOR } z)$$
$$G_3 = (w \text{ XOR } x) \text{ OR } (y \text{ EQ } z)$$

For our study, we implemented their exact experiment and achieved comparable results, proving that our simulation setup was similar to that of Parter et al.'s. With this confirmation, we then constructed genomes capable of expressing more complex boolean functions with 6 inputs. These were then evolved on the following goals:

$$G_4 = (u \text{ XOR } v) \text{ OR } (w \text{ XOR } x) \text{ OR } (y \text{ XOR } z)$$
$$G_5 = (u \text{ EQ } v) \text{ OR } (w \text{ XOR } x) \text{ OR } (y \text{ XOR } z)$$
$$G_6 = (u \text{ XOR } v) \text{ OR } (w \text{ XOR } x) \text{ OR } (y \text{ EQ } z)$$

Lastly, during our experimentation, we determined that the 6 input circuits would need multiple outputs in order to aptly express the difficulty of increased complexity. Ergo, we yet again enhanced the complexity of the boolean functions, creating genomes that encoded for 6-input, 3 output circuits, following inspiration from Kashton and Alon's *Spontaneous evolution of modularity and network motifs*. These circuits were trained on the following goals:

$$G_7 = (x \text{ XOR } y) \text{ OR } (z \text{ XOR } w); (x \text{ XOR } y) \text{ AND } (t \text{ XOR } u); (z \text{ XOR } w) \text{ AND } (t \text{ XOR } u)$$
$$G_8 = (x \text{ XOR } y) \text{ AND } (z \text{ XOR } w); (x \text{ XOR } y) \text{ OR } (t \text{ XOR } u); (z \text{ XOR } w) \text{ AND } (t \text{ XOR } u)$$
$$G_9 = (x \text{ XOR } y) \text{ AND } (z \text{ XOR } w); (x \text{ XOR } y) \text{ AND } (t \text{ XOR } u); (z \text{ XOR } w) \text{ OR } (t \text{ XOR } u)$$
$$G_{10} = (x \text{ XOR } y) \text{ AND } (z \text{ XOR } w); (x \text{ XOR } y) \text{ AND } (t \text{ XOR } u); (z \text{ XOR } w) \text{ AND } (t \text{ XOR } u)$$

Despite this complexity increase, the majority of the settings were kept the same; however, the goal was changed every 20 generations in the order of $G_7, G_{10}, G_8, G_{10}, G_9, G_{10}, G_7, G_{10} \ldots$ This order enabled the changes to occur in a systemic fashion where the number of changes from one goal to the next was minimized.

3.4 Constraints

During our experimentation, we encountered several simulations (approximately 80% of trials) that converged to a local maximum. In our study, we define a local maximum as a network that lacks "critical design portions" and yet still achieves a fitness greater than 0.5. Local maximum circuits evolve along a non-optimal path, drastically reducing the probability of achieving a more optimal fitness in the future due to their genomic neighborhoods[1] having very poor fitness. To address this, we implemented a variety of constraints and tested their effect on improving fitness and evolution[2] by encouraging the development of these "critical design portions".

The following penalties were implemented to protect the following "critical design portions":

1. To increase the maximum possible fitness, we enforced the use of all three output nodes. Failure to do so resulted in a penalty of -1.
2. To allow for more complex representations, we enforced the differentiation between input and output nodes with a penalty of -1 for its absence.
3. We wanted to encourage the use of all six input nodes to allow for more complex representations, so we penalized circuits with a penalty of -.02 per input absence. However, we also wanted circuits to prioritize correct answers (+.0556 in fitness) over using an additional input.

[1] Note that all FG environment tests were run on the first goal defined for the experiment type

4. We enforced the use of 17-22 gates to add constant pressure for circuits to use more gates for more complex representations. Failure to comply resulted in a penalty of -.02 if under 17 gates and -.2 if over 22 gates. However, we also subtracted a .2 penalty to favor moves with higher fitness. We discovered that perfect solutions could be achieved in fewer than 22 gates, so we imposed harsher penalties to avoid this.

3.5 Evaluation

The Girvan-Newman algorithm is a hierarchical clustering algorithm that is commonly used for detecting community structure in complex networks. The algorithm works by iteratively removing edges from the network, with the aim of identifying the edges that connect communities of nodes. At each iteration, the edge with the highest betweenness centrality (i.e., the edge that lies on the most shortest paths between pairs of nodes) is removed, and the resulting network is partitioned into communities using a clustering algorithm. This process is repeated until the network is completely disconnected.

The Girvan-Newman algorithm identifies the underlying modularity structure of a network, which is a measure of how well a network is divided into communities or modules. Additionally, the algorithm can be used to identify key nodes or edges that play a critical role in maintaining the integrity of the network. Therefore, it plays a critical role in mathematically assessing the "modularity score" of a circuit. As explained by Parter et al. the resulting modularity of a circuit supports the existence of FV in a circuit's evolution. In this study, we use their equation[1] to quantify the modularity of the circuit's graph as a real number

$$Q = \sum_{s=1}^{K} \left[\frac{l_s}{L} - \left(\frac{d_s}{L} \right)^2 \right]$$

where K is the number of communities/modules in a network, l_s is the number of edges between nodes in module s, and d_s is the sum of the degrees of the nodes in module s. The reasoning behind this measure of modularity is that an effective division of a network into modules should consist of a substantial number of edges within each module and a minimal number of edges connecting different modules. As a catch for networks that cannot easily be divided or can be divided into one module, we set $Q = 0$. It is worth noting that a high score of modularity, denoted as Q, does not automatically indicate the occurrence of FV, as a network may display modularity without possessing modules that can be efficiently reconfigured to create beneficial phenotypes. Nonetheless, a network with a high modularity score is more likely to demonstrate FV in its evolution, and hence, it is the sole valuable measure to mathematically evaluate FV.

3.6 Experimentation

We conducted 12 experiments in total, with 8 of them being non-constrained tests, similar to Parter et al.'s study. In these tests, we wanted to experiment with different selection algorithms, including Elitism, BEST, and EXP, in order to see which best promotes the evolution of 6-input logic circuits. For each test, we ran experiments on FG and MVG to compare results. We found that Elitism and EXP worked best for these tests, and we used these settings for the next round of tests. For the next tests, we imposed constraints (as described in Section 3.4). In our study, we applied constraints that resulted in fitness=1.0 solutions similar to the ones obtained by Parter et al. These constraints were meant to encourage circuits to reach global maxima.

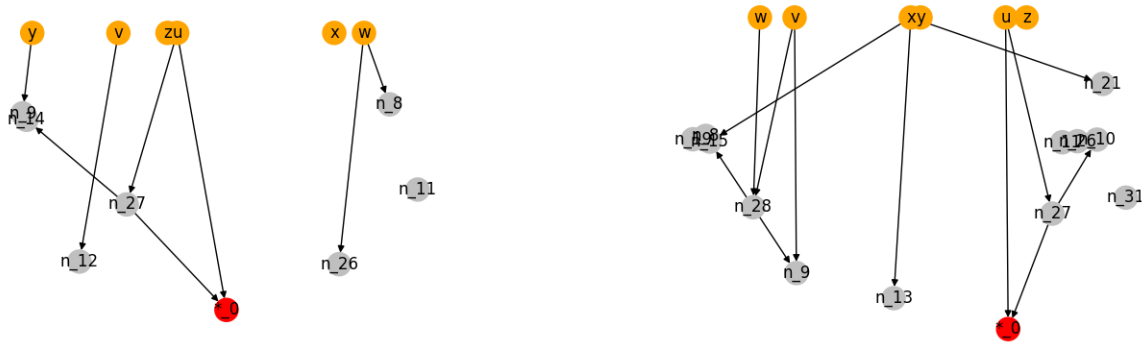
To evaluate if incorporating these constraints could improve circuit performance, we conducted experiments that varied by running simulations with penalties and without penalties. We compared the performance differences between these tests by running FG versus MVG tests. Our aim was to observe differences in modularity, network phenotype, and max/average fitness between the simulations.

4 Results

PROBLEM: Papers did not give specifics for their genetic algorithms or offer. Our research show that this form of experiment is very susceptible to local maxima.

4.1 6 input / 1 output

At first, we attempted to extend our simulation from 4-input 1-output circuits to 6-input 1-output circuits using the same settings. While these circuits achieved high fitness levels initially, we later realized that they quickly converged to a local maxima with a maximum fitness of .875 and mean fitness of approximately .84. After reviewing our experiments, we discovered that certain objectives, such as $G_4 = (u \text{ XOR } v) \text{ OR } (w \text{ XOR } x) \text{ OR } (y \text{ XOR } z)$, were vulnerable to inspiring less sophisticated circuits (ie. lacking critical design portions) that could attain high fitness levels with random initialization.For instance, as depicted in Figures 3a and 3b, both circuits utilized only one input and one NAND gate to achieve a fitness level of .875, despite the stark difference in the number of gates.



(a) Generation 5000.

(b) Generation 6000.

Figure 3. FG-evolved circuit

4.2 6 input / 3 output (Non-Constrained)

To prevent less sophisticated circuits from obtaining high fitness levels, we introduced additional complexity by allowing circuits to have three outputs instead of one. To accommodate this change, we permitted a goal function to include separate subgoals divided by semicolons. A circuit's fitness was calculated by taking its input and computing the output for each subgoal. For instance, $G_7 = (x \text{ XOR } y) \text{ OR } (z \text{ XOR } w); (x \text{ XOR } y) \text{ AND } (t \text{ XOR } u); (z \text{ XOR } w) \text{ AND } (t \text{ XOR } u)$ consists of three distinct goal functions, each potentially serving as one of the circuit's outputs.

To evolve this type of circuit, we experimented with various selection algorithms in both FG and MVG environments (see Section 3). We assessed the quality of each algorithm by evaluating the network phenotype (including the number of utilized input and output nodes in the best circuits), maximum and average fitness, and the modularity score, Q, across generations. Additionally, we considered the non-Elitism case with the EXP parent selection as the control method, following the selection algorithm outlined by Parter et al. for their 4-input and 1-output circuits[1].

Overall, the Elitism tests generated logic circuits with desirable traits, yielding results comparable to the control. In some instances, the Elitism/BEST selection algorithm produced circuits that utilized the highest number of input and output nodes, which we understand as beneficial for achieving greater robustness and expressivity in circuits through more fit genomic and phenotypic neighborhoods. The Elitism/EXP tests displayed Q modularity scores that aligned with the baseline set by Parter et al., where MVG represents higher modularity and FG indicates lower modularity. However, we noticed that several input and output nodes went unused in a substantial number of non-constrained outcomes. Additionally, the selection algorithm yielding the most favorable qualities encompassed all Elitism tests with both EXP and BEST selection.

In all cases, circuits evolved to fit a local maxima instead of achieving a global maxima with more intricate circuits. To guide solutions toward the global maxima, we intend to constrain the evolution by penalizing less sophisticated solutions.

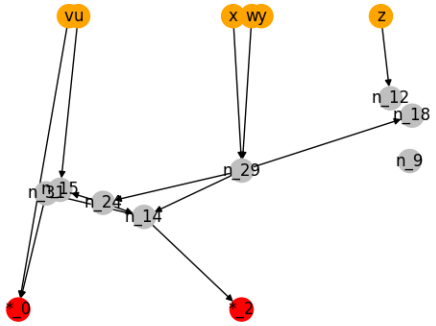
4.2.1 Control

In Figures 4a and 4b, it is observed that neither of the circuits utilized all of the available six inputs or the three outputs. A comparative analysis of the fitness levels achieved by both circuits reveals a similar performance, with the MVG environment exhibiting a marginally higher average fitness than the FG, as illustrated in Figures 4c and 4d. Intriguingly, the graphical representation of the results demonstrates that the circuits evolved within the FG environment were, on average, more modular than those in the MVG environment.

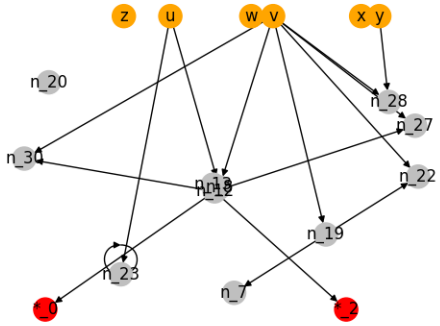
In conclusion, this observation suggests that the relationship between modularity and complexity may not be consistently maintained in more intricate environments. However, it may also suggest that FG evolved towards modules (with no guarantee that those modules are particularly useful), while MVG was still struggling to adapt to the rapidly changing environment.

4.2.2 Elitism Tests

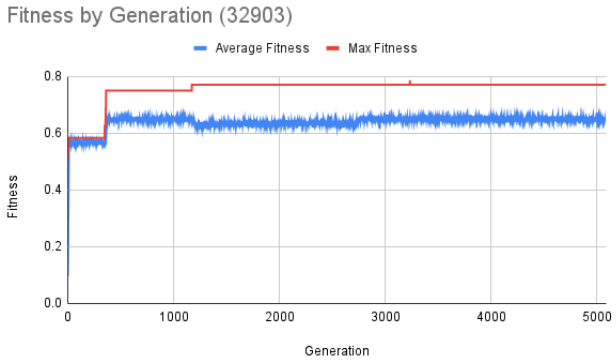
We conducted experiments to evaluate the effectiveness of combining Elitism with other genetic selection algorithms, such as selecting parents based on exponential probabilities (EXP) or choosing the best 25% of the population as parents and reproducing only with them (BEST). Despite the variations in these experiments, the highest-fitness circuits achieved comparable average



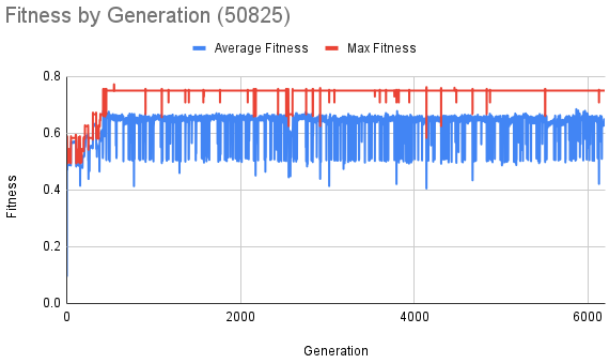
(a) Circuit in FG environment.



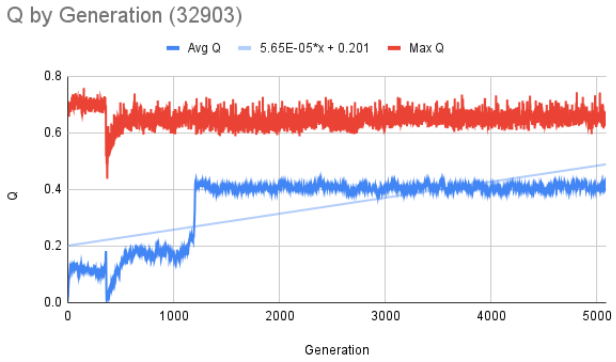
(b) Circuit in MVG environment.



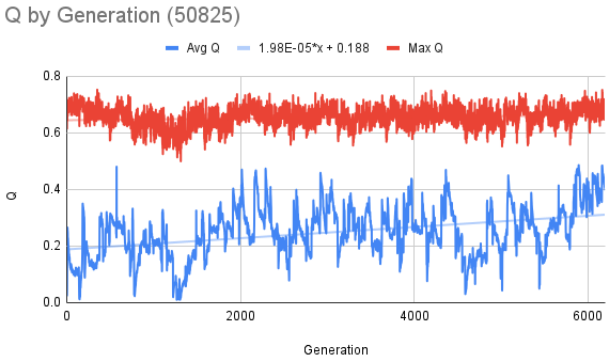
(c) Fitness in FG environment.



(d) Fitness in MVG environment.



(e) Modularity Q score in FG environment.



(f) Modularity Q score in MVG environment.

Figure 4. Control No-Elitism EXP.

fitness levels to those of the control population. However, the network phenotypes and modularity Q scores differed significantly with each experiment.

Our experiments using the Elitism/EXP genetic algorithm have shown that in a FG environment, there was low modularity, while in a changing genotype (MVG) environment, modularity increased. This behavior is in line with the theory proposed by Parter et al.[1], which suggests that modularity tends to increase under MVG environments. However, it is important to note that both solutions in our experiments, as seen in Figures 5 and 6, utilized only two inputs within the circuits and two outputs. Assuming the other inputs are necessary for creating the correct output and cannot be abstracted away, this solution is a local maxima, even though it is achieving fitness rates close to the control. Additionally, in the experiments with the Elitism/BEST

circuits, it is valuable to note that these resulted in circuits that utilized the most input and output nodes in this round of testing, meaning that these evolutionary settings likely inspire genotypes with more fit phenotypic neighborhoods and are more ideal for running future simulations.

In conclusion, the Elitism/EXP genetic algorithm produced low modularity in a FG environment but increased modularity in an MVG environment, consistent with Parter et al.'s theory. However, the resulting circuits only utilized two inputs and outputs, which may indicate a local maximum. In contrast, the Elitism/BEST algorithm resulted in circuits with the most input and output nodes, suggesting they may be more suitable for future simulations with more fit phenotypic neighborhoods.

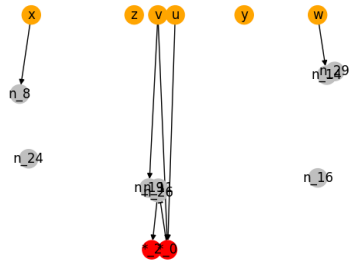


Figure 5. Elitism/EXP circuit in FG environment

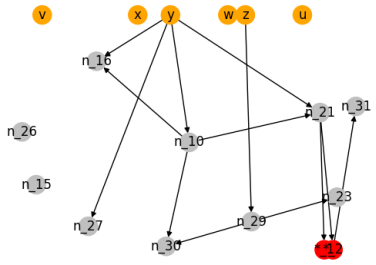


Figure 6. Elitism/EXP circuit in MVG environment

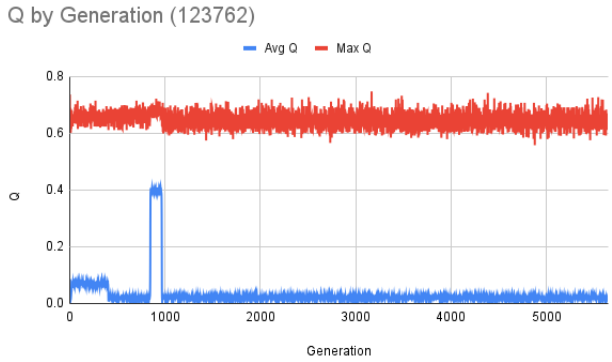


Figure 7. Elitism/EXP Q in FG environment

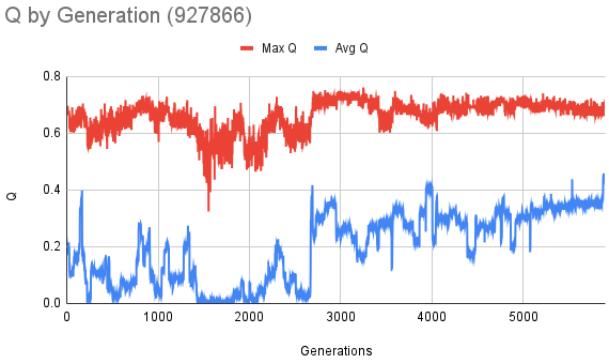


Figure 8. Elitism/EXP Q in MVG environment

4.2.3 Non-Elitism and BEST

The experiments utilizing the Non-Elitism/BEST genetic algorithm yielded suboptimal results with regards to the proximity of solutions to the ideal fitness=1 solutions, as well as the modularity of solutions. Specifically, the circuits generated by the algorithm had a phenotype where only one input corresponded to the one output for the entire circuit, which is not ideal. These shortcomings are attributed to the BEST algorithm's characteristics. In particular, as the algorithm selects the 500 best-performing circuits to recreate the entire population, the best circuits are rewritten each generation, limiting diversity and increasing the chances of local minima. Furthermore, the experiments had low variability from one environment to the next, further exacerbating the susceptibility to local minima.

In conclusion, the Non-Elitism/BEST genetic algorithm failed to reach a fitness near the control, likely due to the algorithm's tendency to select and rewrite only the best circuits in each generation. This leads to limited diversity and susceptibility to local minima. Low variability from one environment to the next further compounded these issues.

4.3 6 input / 3 output (Fully Constrained)

As seen in the non-constrained experiments, nearly all of the evolved circuits do not utilize all 6 inputs and use all 3 outputs. This causes circuits to evolve and remain in local maxima for thousands of iterations (see Figure 4c and 4d). To improve the performance of these circuits, we implement constraints that penalize evolutionary paths that stray from the global maxima in

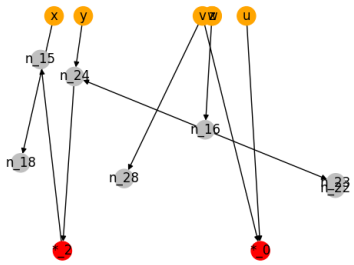


Figure 9. Elitism/BEST circuit in FG environment

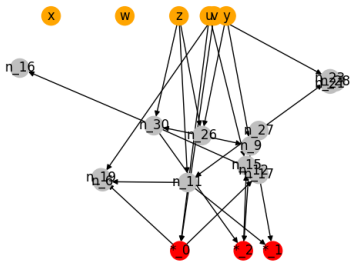


Figure 10. Elitism/BEST circuit in MVG environment

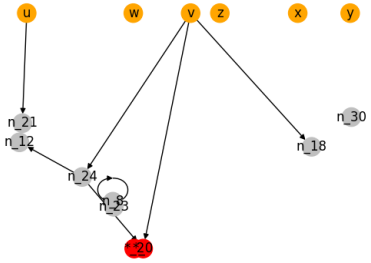


Figure 11. Non-Elitism/BEST circuit in FG environment

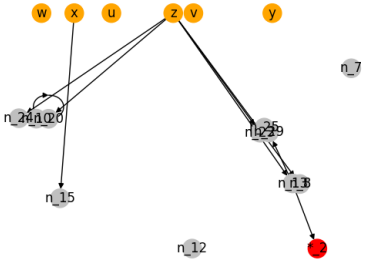


Figure 12. Non-Elitism/BEST circuit in MVG environment

specific ways (see 3.4). From the no constraint 6-input 3-output experiments, we determined that Elitism paired with EXP selection achieved results consistent with Parter et al. (MVG = more modularity, FG = less modularity), so we applied this selection algorithm in our constrained experiments. We achieved the best fitness results with the penalized circuits. Applying this approach to our genomic evolution for 6 input, 3 output circuits resulted in an MVG simulation that used all six inputs, three outputs, with some experiments achieving a maximum fitness of 1.

In conclusion, only through using full constraints and Elitism/EXP were we able reach a maximum fitness of 1, which involved using 6 inputs, 3 outputs, and 17-22 gate for an MVG evolution experiment.

4.3.1 No Additional Penalties

In both FG and MVG scenarios, the circuits evolved without any additional penalties (except for requiring all outputs to be present and not allowing input nodes in the output). These circuits converged around an average fitness of 0.7. Notably, they didn't utilize all the inputs but used three outputs, which is reasonable since they were constrained to use all outputs without penalties for using all inputs. Moreover, only a few (3-5) NAND gates were used outside of the output gates.

Interestingly, the circuits without penalties achieved the best results in terms of facilitated variation and modularity. It was easy to distinguish between an FG and an MVG environment. In the FG scenario, the Q modularity score decreased and converged to a value of 0.05 over time. In contrast, in the MVG environment, circuits steadily increased in modularity as time passed. These findings align with the experiments conducted by Parter et al.[1].

In conclusion, the circuits that evolved with only output constraints produced results that seem to align with the principles of facilitated variation in terms of modularity scores. However, the fitness on these circuits were on average lower than circuits evolved without any constraints.

4.3.2 With Input and Gate Count Penalties

In this study, penalties were applied to ensure all inputs were used and that the number of gates were within the 17-22 range. This led to similar findings as those described by Parter et al. regarding network phenotypes and maximum fitness by generation. By the 1500th iteration, the maximum fitness reached 0.88 in both FG and MVG environments, as demonstrated in Figures 18c

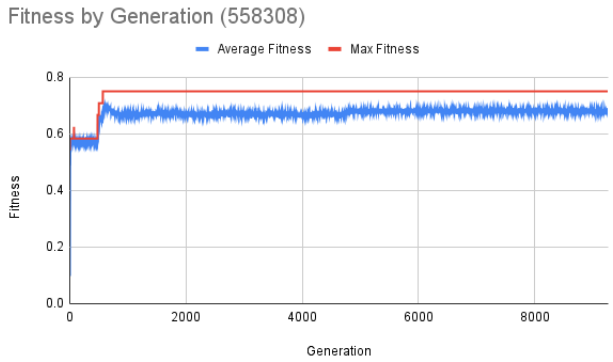


Figure 13. Non-Elitism/BEST Fitness in MVG environment

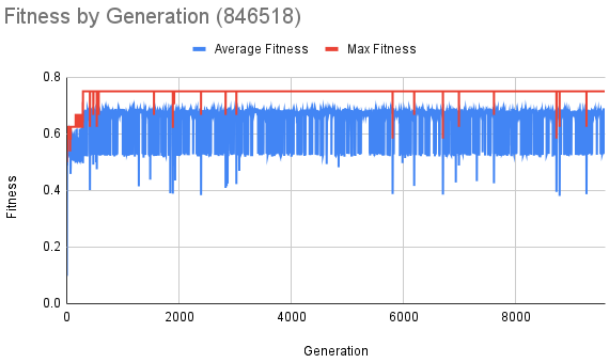


Figure 14. Non-Elitism/BEST Fitness in MVG environment

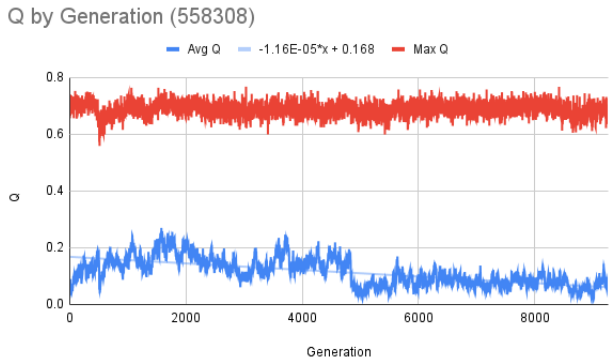


Figure 15. Non-Elitism/BEST Q in MVG environment

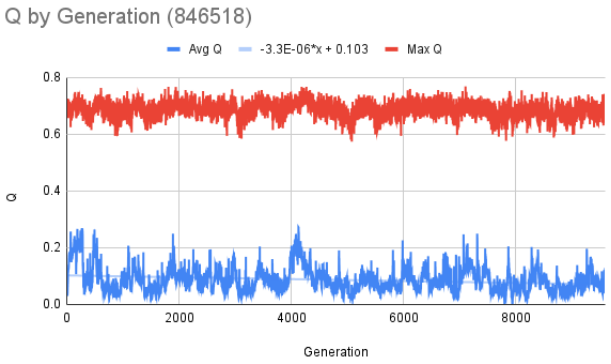


Figure 16. Non-Elitism/BEST Q in MVG environment

and 18d.

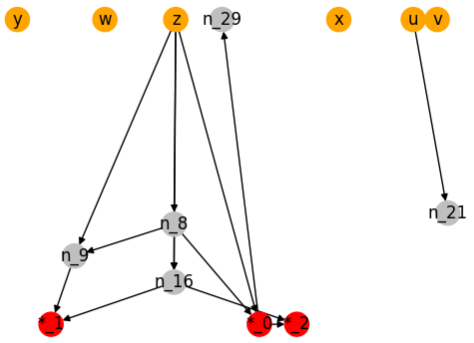
Both circuits had high Q scores in comparison to each other, with the FG circuit scoring higher than the MVG circuit. This could be due to the high number of potentially unnecessary gates. The circuits might appear modular, but this could be a result of the large number of gates constraining the graph, as shown in Figure 18a and 18b.

Using these input and gate count penalties, we achieved our first perfect solution circuits evolving in the MVG environment, with a fitness score of 1. The circuit, shown in Figure 19, reached a perfect fitness=1 solution in roughly 12,000 iterations and used 17 gates within the circuit itself. Although these constraints can lead to similar solutions, perfect solutions won't be reached every run due to the randomness in evolution. For instance, none of the identical simulation runs achieved the same results.

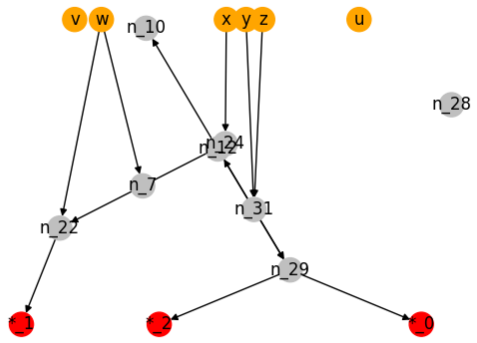
In conclusion, by heavily directing the evolution towards the global minimum, we achieved perfect fitness results. Furthermore, we observed similar modularity scores in both FG and MVG environments. This similarity could be attributed to the pressure placed on the circuits to have a larger number of gates (17-22).

4.4 Takeaways

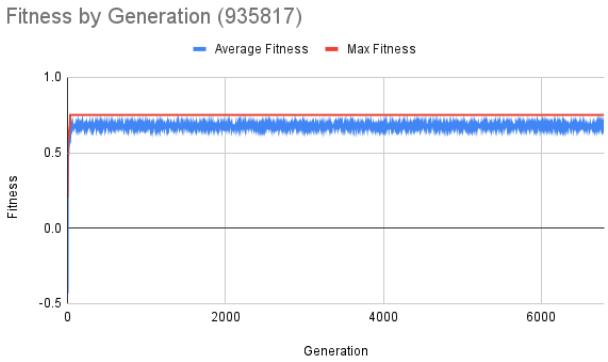
In summary, the experiment suggests that the relationship between modularity and complexity is not consistent in more complex environments. The Elitism/EXP genetic algorithm produced low modularity in an FG environment but increased modularity in an MVG environment, which is consistent with Parter et al.'s theory. The Elitism/BEST algorithm resulted in circuits with the most input and output nodes, which may be more suitable for future simulations with fit phenotypic neighborhoods. The Non-Elitism/BEST genetic algorithm failed to reach a fitness near the control, likely due to limited diversity and susceptibility to local minima. Only by using full constraints and Elitism/EXP, were the we able to achieve a maximum fitness of 1, involving six inputs, three outputs, and 17-22 gates for an MVG evolution experiment. Overall, the results suggest that different genetic algorithms and environments can lead to varying levels of modularity and complexity in evolved increasingly complex circuits,



(a) Best evolved FG circuit in 6,500 generations.

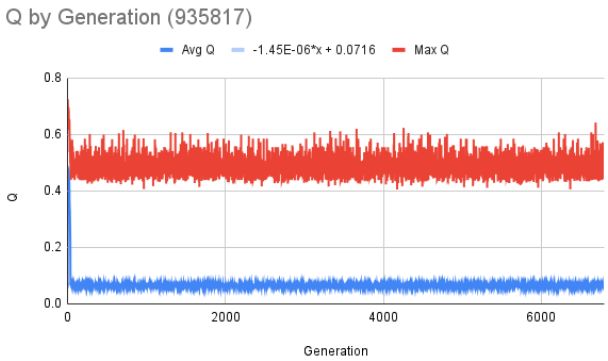


(b) Best evolved MVG circuit in 3,800 generations.

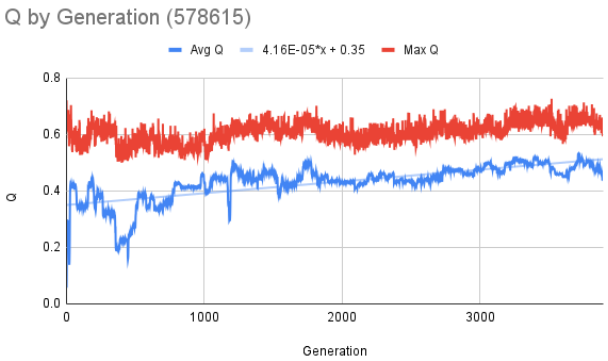


(c) FG fitness over 6,500 generations.

(d) MVG fitness over 3,800 generations.



(e) FG modularity Q score over 6,500 generations.



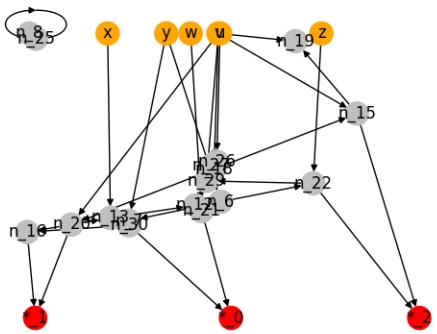
(f) MVG modularity Q score over 3,800 generations.

Figure 17. Forced 3-output and Elitism/EXP in FG/MVG environments (no other penalties)

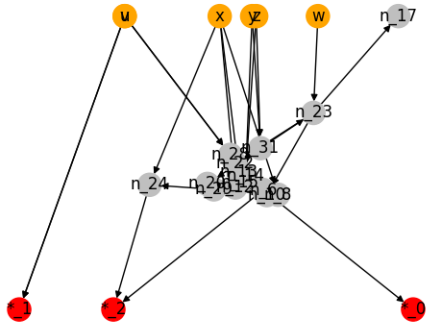
which falls in line with our initial hypothesis.

5 Next Steps

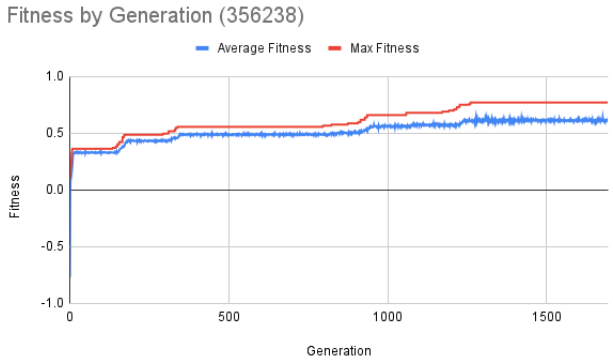
One of the next steps we plan to take is to run the simulations for a longer time, specifically a time comparable to the original study. Parter et al. ran the simulations for 100,000 generations, which is computationally infeasible for this paper. For this study, we were able to run the simulations long enough to observe a significant diversion from our base paper's conclusions, but we would like to run the simulations for 100,000+ generations to observe changes in the circuit's performance and modularity



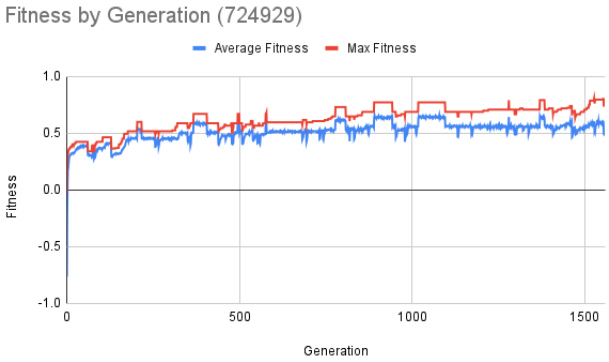
(a) Best evolved FG circuit in 1,800 generations.



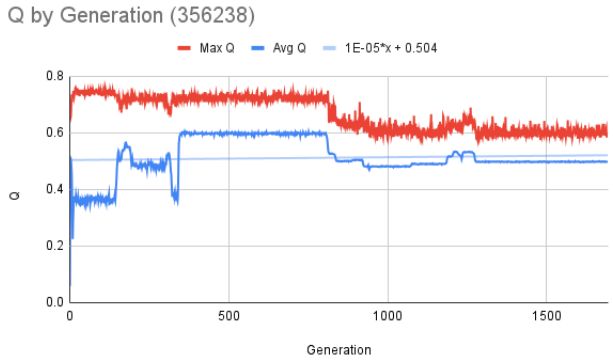
(b) Best evolved MVG circuit in 1,550 generations.



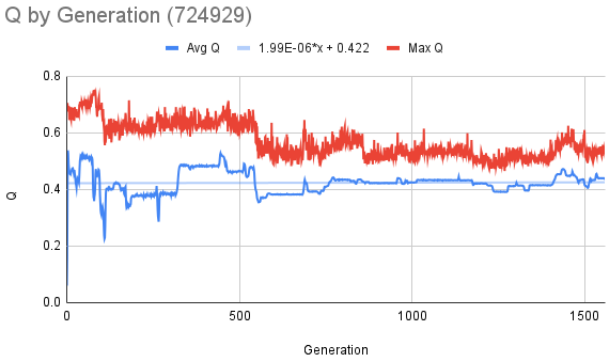
(c) FG fitness over 1,800 generations.



(d) MVG fitness over 1,550 generations.



(e) FG modularity Q score over 1,800 generations.



(f) MVG modularity Q score over 1,550 generations.

Figure 18. Forced 3-output and Elitism/EXP in FG/MVG environments (Not using input penalty, and gate within 17/22 penalty).

over time. This will allow us to determine what else might spontaneously occur over time if not FV.

In addition to running longer simulations, we plan to perform more complicated tests on RNA secondary structure. The original study by Parter et al. used RNA secondary structure as one of the model systems to investigate the emergence of FV. However, their study was limited to simple RNA structures. With additional time, we plan to use more complex RNA structures to explore the extent to which FV can be observed in these structures.

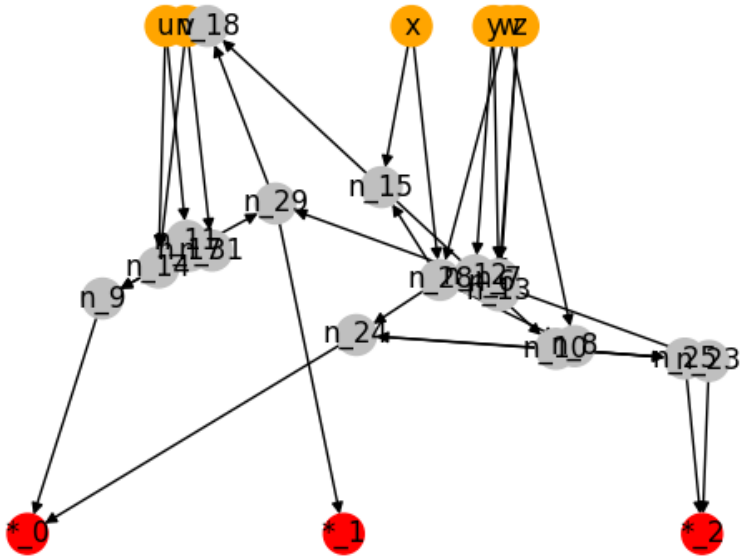


Figure 19. Perfect Solution Circuit to MVG Environment

6 Conclusion

In conclusion, our research indicates that modularity does not necessarily uphold as complexity increases. Although modularity can be maintained when constraints are added to penalize improper networks, the sheer quantity of constraints needed renders this approach unrealistic. Our findings suggest that the theory of facilitated variation spontaneously occurring through modular environments alone may need to be reevaluated in light of these results.

This paper's research challenges the assumptions underlying FV and sheds light on the limitations of previous studies on this topic. FV proposes that genetic mutations can affect the regulatory mechanisms that control core functional modules in living organisms, leading to the emergence of new traits. However, the effectiveness of FV is dependent on the maintenance of modularity, and our research suggests that modularity may not be maintained as complexity increases.

In computer science, understanding how to promote facilitated variation and modular design can have major performance benefits. One of the reasons why computational evolution has difficulty in scaling up to higher complexity is the lack of modularity in designs. Ergo, the benefits of promoting modularity in computational evolution are significant. These approaches enable the design of complex systems that are more efficient, reliable, and easier to maintain. Furthermore, modular design facilitates the reuse of components, allowing for faster development, generalizability, and reduced costs.

Overall, our findings have implications for the study of evolutionary biology and suggest that the role of modularity in the evolution of new traits may be more complex than previously thought. Further research is needed to better understand FV and the factors that influence its maintenance of modularity.

7 Methods

While attempting to reference and emulate previous research, a notable obstacle we encountered was the absence of a precise methodology. As a result, we sought to make our approach as transparent and comprehensive as possible (refer to Section 3). Despite our efforts, we still also opted to disclose the simulations and software we used in our experimentation, which can be accessed via this Github repository.

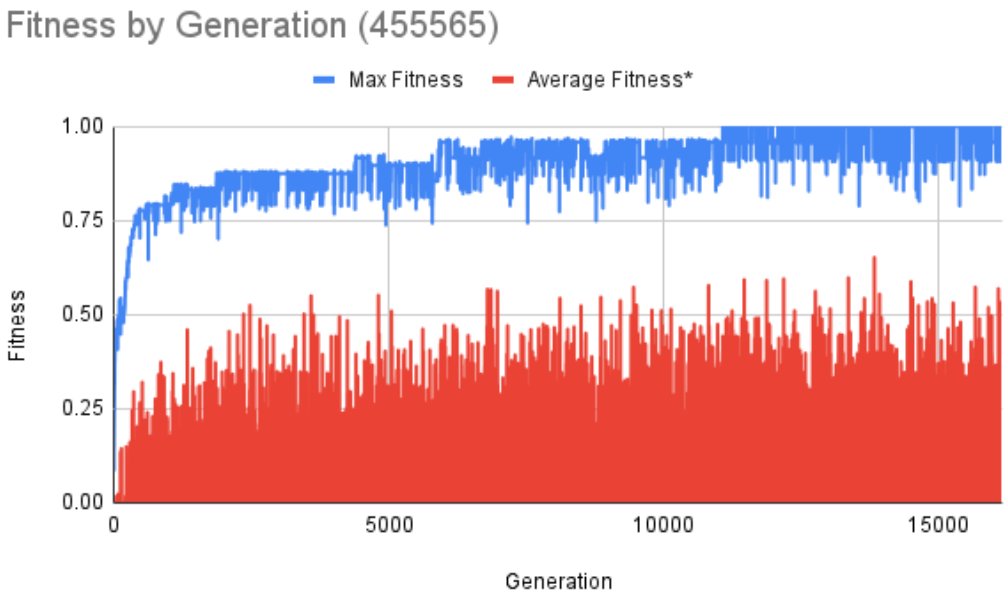


Figure 20. Perfect Solution Fitness Graph to MVG Environment; Note: the highly oscillating average fitness is the result of overly harsh penalties that drag the average below 0

Acknowledgements

On a more personal note, we really want to thank the COMPSCI 229r team, including Professor Valiant, John Wang, and Aayush Karan, for an amazing class and for so much support throughout this whole project. We learned so much through not just the lectures but also through the many struggles and challenges we overcame for this long project: brainstorming, reading research papers, implementing our code, running the experiments, creating visualizations, and more were all certainly challenging, but we feel like each challenge we overcame taught us value concepts and lessons.

References

1. Parter, M., Kashtan, N. & Alon, U. Facilitated variation: How evolution learns from past environments to generalize to new environments. *PLOS Comput. Biol.* **4**, 1–15, DOI: 10.1371/journal.pcbi.1000206 (2008).

2. Kashtan, N. & Alon, U. Spontaneous evolution of modularity and network motifs. *Proc. Natl. Acad. Sci.* **102**, 13773–13778, DOI: 10.1073/pnas.0503610102 (2005). https://www.pnas.org/doi/pdf/10.1073/pnas.0503610102.

3. Mitchell, M. *An Introduction to Genetic Algorithms* (MIT Press, Cambridge, MA, USA, 1998).