Gene Duplications Drive the Evolution of Complex Traits and Regulation

Alexander Lalejini[1,2,3], Michael J. Wiser[2,3], and Charles Ofria [1,2,3]

[1]Department of Computer Science, Michigan State University, East Lansing, MI, USA
[2]BEACON Center for the Study of Evolution in Action
[3]Ecology, Evolutionary Biology, and Behavior Program, Michigan State University, East Lansing, MI, USA
alex@lalejini.com

Abstract

Gene duplications have been shown to promote evolvability in biology and in computational systems. We use digital evolution to explore *why*; that is, what characteristics of gene duplications increase evolutionary potential? Are duplications valuable because they inflate the effective mutation rate, generating increased amounts of genetic variation? Or is it that those mutations are clustered together? Or, is it that the mutations insert genetic material, providing evolution an easy technique to select for longer genomes? Does the value pertain to the information being duplicated in the genome? If so, is the full structure of duplicated code critical, or would the duplication of functional building blocks be valuable even if rearranged? Using the Avida Digital Evolution Platform, we experimentally tease apart these aspects in two qualitatively different environments: one where complex computational traits are directly selected, and another where those traits need to be regulated based on current environmental conditions. We confirm that gene duplications promote evolvability in both static and changing environments. Furthermore, we find that the primary value of gene duplications comes from their capacity to duplicate existing genetic information within a genome. Specifically, while duplications that randomize the order of genetic material are valuable, the most useful form of duplication also preserve the structure (and thus information content) of duplicated sequences.

Introduction

Gene duplication is a phenomenon by which additional copies of a region of genetic material are incorporated into a genome. There are numerous processes that can result in gene duplication, each with their own specific outcomes (as reviewed in (Zhang, 2003)), ranging from repeats of gene fragments to whole genome duplication. All of these processes result in an increase in genome size, but more importantly, genetic material that is likely more structured and meaningful than random insertions. As such, gene duplication is an important mechanism for generating genetic novelty, providing a source of new genetic material for evolutionary processes to act on and enabling new evolutionary opportunities (Zhang, 2003; Crow and Wagner, 2006; Magadum et al., 2013). Indeed, gene duplication has been shown to promote evolvability – the capacity of a system to generate adaptive phenotypic variation and to transmit that information via an evolutionary process (Hu and Banzhaf, 2010) – both in biology and in computational evolution.

Gene duplications have been shown to enhance evolvability in a number of biological systems. A striking example comes from the Long-Term Evolution Experiment in *Escherichia coli*, where a duplication in one population allowed a protein to be expressed in an environment where it would normally be inhibited (Blount et al., 2012). By expressing this protein in a different context, the cell gained access to a resource that it previously could not metabolize resulting in a 7-fold increase in population size. Beyond specific cases of how a gene duplication led to increased evolvability, large scale genomic studies have discovered cases where a strikingly high fraction of the genes in an organism show evidence of having arisen from gene duplications (Teichmann et al., 1998; Teichmann and Babu, 2004), emphasizing the role of duplication in evolutionary innovation. Comparative studies further suggest that many ancient duplication events were associated with increases in both genetic robustness and evolutionary innovation (reviewed in (Wagner, 2008)). Though comparative studies lack the definitive proof of laboratory manipulation, the accumulated weight of evidence clearly suggests that gene and genome duplication events can have long-term evolutionary consequences.

The prominence of gene duplications in biological evolution has inspired their use in artificial evolutionary systems. In genetic programming, the optimal program architecture for solving a particular problem is challenging to predict *a priori*. Inspired by Ohno's *Evolution by Gene Duplication* (Ohno, 1970), Koza used gene duplication and deletion operators to co-evolve genetic programs and their structure, finding that gene duplication and deletion increased program evolvability and produced simpler solutions (Koza, 1995). Calabretta *et al.* evolved modular neural network motor controllers for robots with and without module duplication operators, finding evidence that access to duplication operations resulted in increased functional specialization in network modules (Calabretta et al., 1998, 2000). These and other computational studies (Ryan et al., 1998; Sawai and

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

Adachi, 1999, 2000; Schmitt, 2005) corroborate some of the benefits of gene duplication seen in natural systems.

Given the evidence that gene duplication promotes evolvability in both computational and natural systems, we use a digital evolution approach to explore *why*. That is, what aspects of gene duplications promote evolvability?

Is the full structure of duplicated code critical, or would the duplication of the functional elements be valuable even if they are rearranged? Exact duplications can result in functionally redundant genes that can increase the mutational robustness of a genotype (Crow and Wagner, 2006) or allow the organism to produce additional gene product (Zhang, 2003). If a highly constrained genetic sequence is duplicated, one of the versions can mutate more freely, which may lead to new functionality (Zhang, 2003; Wagner et al., 2003) – a process known as neofunctionalization. Alternatively, subfunctionalization may occur where the two 'daughter' genes diverge from the ancestral gene state, specializing on different aspects of the ancestral gene's functionality (Zhang, 2003). Gene duplications are also thought to play an important role in the evolution of complex genetic regulation (Teichmann and Babu, 2004). If a regulatory gene is duplicated, the copy could continue to regulate the same target genes, but mutations may cause it to respond to new signals; or, the duplicated regulatory gene may continue to respond to the same signals, but it may begin regulating a different gene (Teichmann and Babu, 2004).

Aside from creating redundant copies of existing genetic code in a genome, there are other features of gene duplication that may result in increased evolvability. Are gene duplications valuable because they inflate the effective mutation rate of genomes, increasing genetic variation? Or is the important factor that those new mutations are clustered together? Or is it that the mutations are insertions that provide evolution with an easy technique to select for longer genomes and increased information storage capacity? These aspects are nearly impossible to disentangle in biological systems, but can be addressed in computational ones.

Using the Avida Digital Evolution Platform, we implemented a series of mutation operators to systematically isolate aspects of gene duplication and tease apart which factors promote evolvability. We tested these operators in two contexts: in a static environment that rewards the performance of nine basic Boolean logic operations, and in two dynamic environmental conditions that require the evolution of regulatory mechanisms capable of altering which operations are expressed as a function of current environmental conditions. There are several mechanisms that produce gene duplications in biological systems. Here, we use gene duplication mutation operators in Avida that resemble replication slippage (Bzymek and Lovett, 2001) (*slip mutations*) and allow for gene duplications or deletions at any scale.

We introduce five slip mutation operators to tease apart the specific components of a gene duplication. By observing how each of our slip mutation operators affect the evolution of digital organisms in different contexts, we are able to isolate which aspects of gene duplications are most important for promoting evolvability.

Methods

The Avida Digital Evolution Platform

We conducted all experiments with Avida, which provides a computational instance of evolution where researchers can empirically test hypotheses that would be difficult or impossible to test in natural systems (Ofria et al., 2009).

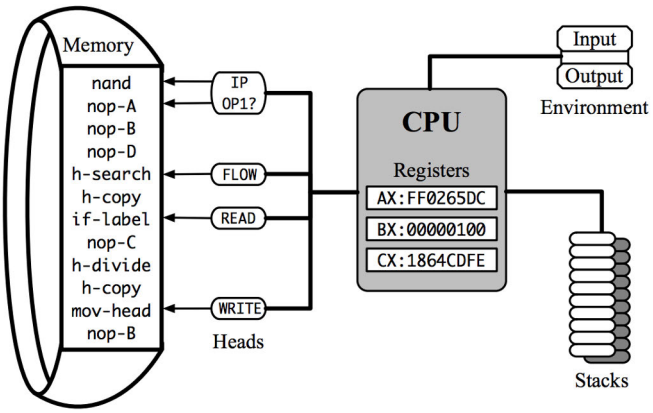

Figure 1: A visual representation of an organism's virtual hardware in Avida. The original figure is from (Ofria et al., 2009).

Digital Organisms in Avida are self-replicating computer programs that compete for space in a finite world. Each organism is defined by a linear sequence of program instructions (*i.e.* its genotype) and a set of virtual hardware. The instruction set in Avida is Turing Complete and syntactically robust – any ordering of instructions is syntactically valid, though not always useful. The instruction set includes operations for basic computations, execution flow control, input and output, and self-replication. Additionally, we incorporate sensory instructions into the instruction set, enabling organisms to sense the state of their current environment. An organism's virtual hardware (depicted in Figure 1) consists of components such as a central processing unit (CPU), registers for computation, buffers for input and output, and memory stacks.

Organisms in Avida replicate asexually by allocating memory for their offspring, copying their genome instruction-by-instruction, and then dividing. However, copy and divide operations are not perfect and can result in mutated offspring. Organisms can influence their replication speed by improving their metabolic rate. An organism's metabolic rate determines the speed at which it executes instructions in its genome; a higher metabolic rate allows an organism to execute its genome faster, and thus, allows the organism to copy itself faster. Initially, an organism's

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

metabolic rate is roughly proportional to its length. However, an organism's metabolic rate can be adjusted when an organism completes a particular task, such as a mathematical computation. In this way, we can differentially reward or punish the performance of different computational tasks.

When an organism finishes replicating, its offspring is placed in a random location anywhere in the population, replacing that location's previous occupant. Thus, improving the efficiency of self-replication or performing rewarded computational tasks are both advantageous in the competition for space in Avida. The combination of competition and heritable variation due to imperfect copy and divide operations results in evolution by natural selection in Avida.

Mutation events in Avida come in four types (substitution, insertion, deletion, or slip) and can be set to occur when an instruction is copied or when an offspring is being born. All mutation rates are independently configurable for any combination of type and timing.

Copy mutations occur when a copy instruction errs. If it is a *substitution* error, a random instruction is written to the copy location instead of the intended instruction. If it is an *insertion*, a random extra instruction is copied along with the intended instruction. If it is a *deletion*, the copy instruction fails to write any instruction at all.

Divide mutations act on an organism's offspring during division. When a divide *insertion* mutation occurs, a random instruction is inserted into the offspring's genome at a random position, increasing the size of the offspring's genome by one. When a divide *deletion* mutation occurs, a random instruction in the offspring's genome is removed, decreasing the size of the offspring's genome by one.

A new type of divide mutations are *Slip mutations*, which enable gene duplications and deletions. When a slip mutation occurs, two sites in the offspring genome are randomly selected, defining the target segment for the operation. If the first site is upstream of the second, the slip mutation results in an insertion; this is as if the organism's replication machinery had slipped backward during replication and recopied a segment. If the second site is upstream of the first, the slip mutation results in a deletion; this is as if the organism's replication machinery slipped forward during replication, skipping over a genetic segment. Our slip mutation operators ensure that insertions and deletions due to slip mutations occur with equal probability; thus, absent selection, we do not create an inherent bias on genome length.

We use five variants of slip mutations: slip-duplicate, slip-scramble, slip-random, slip-NOP, and slip-scatter. When a slip mutation results in a deletion, in all but the slip-scatter variant, the target segment is deleted. Rather than deleting the target segment, the slip-scatter mutation operator distributes a number of single-instruction deletions equal to the length of the target segment uniformly throughout the genome. When a slip mutation results in an insertion, a number of instructions equal to the length of the target segment is inserted into the genome; however, depending on the particular slip mutation operator, which instructions are inserted and where the insertions take place may be different. Figure 2 provides description and examples of how each slip mutation operator handles insertions.

Sensing in Avida In Avida, organisms can use regulatory mechanisms that alter which operations are expressed as a function of environmental conditions. Such (*phenotypic plasticity*) has been shown to evolve in a temporally changing environment (Clune et al., 2007; Lalejini and Ofria, 2016). In our changing environment experiments, we include sensory instructions that, when executed, provide information about the current environmental conditions. Using a combination of these task-specific sensors, an organism can fully resolve the current state of its environment.

Experimental Design

To investigate how gene duplication affects evolvability, we evolved populations in three different environments: a static environment, a simple changing environment, and a complex changing environment.

In the **static environment**, we rewarded the performance of nine Boolean logic functions: NOT, NAND, OR-NOT, AND, OR, AND-NOT, NOR, XOR, and EQUALS (for more information on these functions in Avida see (Lenski et al., 2003)). Rewards for these functions were identical and consistent for the duration of the experiment.

To analyze an organism evolved in the static environment, we evaluated the number of unique computational tasks it could perform, resulting in a *phenotypic match score* that indicates how well the organism is adapted to the static environment. Phenotypic match scores in the static environment range from a minimum of 0 (*i.e.* the organism performs no tasks) to a maximum of 9 (*i.e.* the organism performs all 9 tasks). Equation 1 describes the static match score ($Score_s$), where P is a phenotype and P_s represents the set of tasks phenotype P performs in the static environment.

$$Score_s(P) = |P_s| \tag{1}$$

In the **simple changing environment**, we considered the performance of only four boolean logic tasks: NOT, NAND, OR-NOT, and AND-NOT. At any given time each of these tasks was either rewarded or punished. Thus, there were a total of 16 possible environmental conditions. Starting at the beginning of the experiment and every 50 updates[1] thereafter, the environment changed to a random one of the 16 possible conditions. These environmental fluctuations created a selective pressure for organisms to use sensory instructions to regulate which tasks they perform (*i.e.* phenotypic plasticity) to match the current condition. We chose an

[1] An update in Avida is equal to the amount of time it takes for the average organism to execute 30 instructions; see (Ofria et al., 2009) for further detail.

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

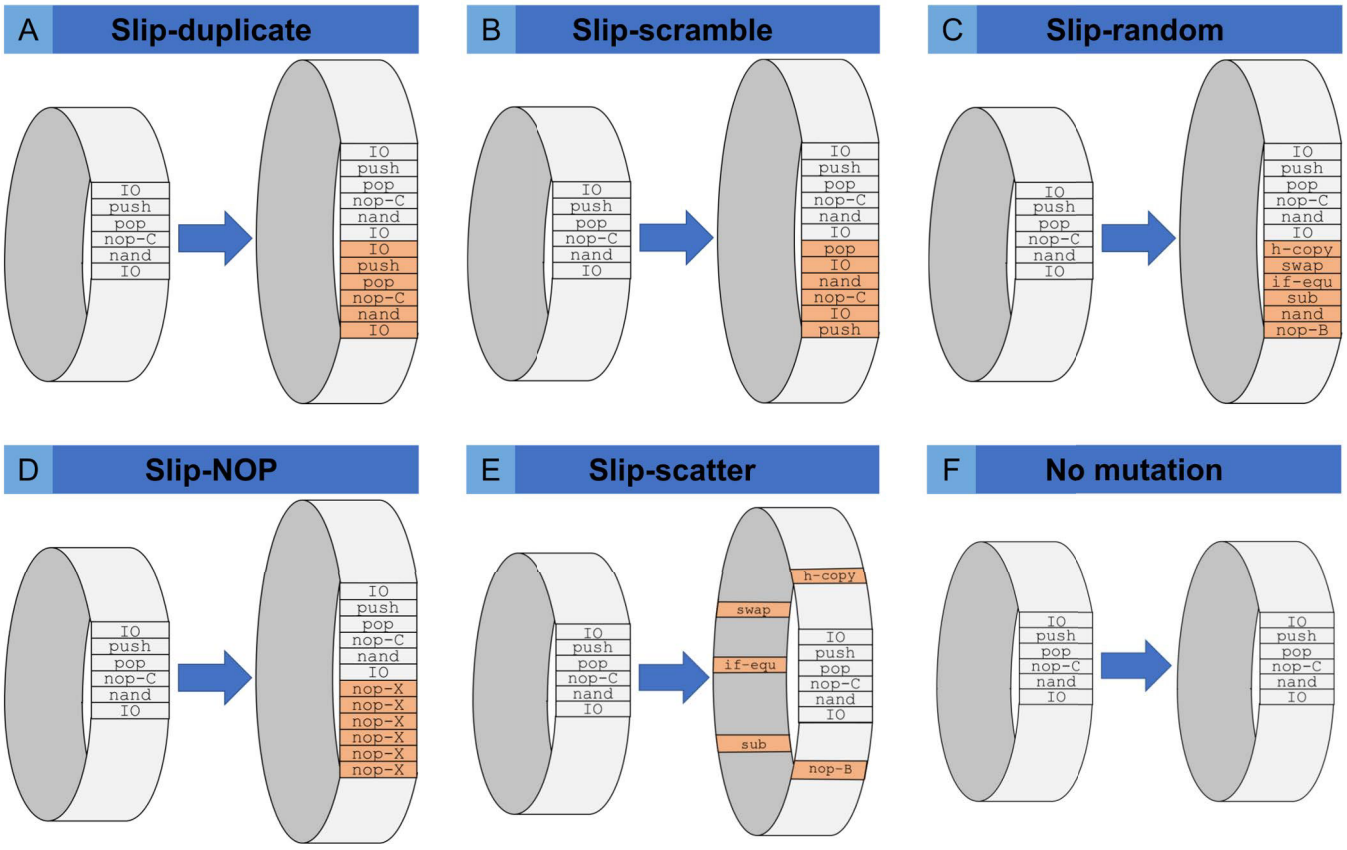

Figure 2: Descriptions and visual examples of how each of our slip mutation operator handles insertions. A) Slip-duplicate: The inserted segment is an exact duplicate of the target segment and is inserted directly after the target segment. B) Slip-scramble: The inserted segment is a shuffled duplicate of the target segment and is inserted directly after the target segment. C) Slip-random: The inserted segment consists of random instructions and is inserted directly after the target segment. D) Slip-NOP: The inserted segment consists of nop-X instructions (a no operation instruction in Avida) and is inserted directly after the target segment. E) Slip-scatter: The inserted segment consists of random instructions and is broken up and inserted into random locations in the genome. F) For comparison, we include an example with no mutation.

environmental change rate of once per 50 updates because previous work has shown that it facilitates the evolution of phenotypic plasticity (Lalejini and Ofria, 2016).

The **complex changing environment** was identical to the simple changing environment, except instead of considering only four boolean logic tasks, we considered nine: NOT, NAND, OR-NOT, AND, OR, AND-NOT, NOR, XOR, and EQUALS. Thus, there were a total of 512 possible environmental conditions, making it significantly more challenging than the simple changing environment.

To analyze an organism that evolved in a changing environment, we computed a phenotypic match score that indicates how well the organism can cope with all possible environmental conditions. Across each environmental condition (16 in the simple changing environment, 512 in the complex changing environment) and for each task we increased an organism's score by one if the task execution matched the condition (*i.e.* performed a rewarded task, or avoided a punished task), and we decreased an organism's score by one if the task execution did not match the condition (*i.e.* performed a punished task or did not perform a rewarded task). We describe phenotypic match scores for a changing environment, $Score_c$, in Equation 2 where P is a phenotype, E is the set of all environmental conditions, P_e is the set of tasks phenotype P performs in environmental condition e, $\neg P_e$ is the set of tasks phenotype P does not perform in environmental condition e, R_e is the set of tasks rewarded in environmental condition e, and $\neg R_e$ is the set of tasks punished in environmental condition e. Given this method of scoring, any non-plastic organisms always receive a phenotypic match score of 0. In the simple changing environment, scores range from a minimum of -64 (*i.e.* perfectly maladaptive plasticity) to a maximum of 64 (optimally adaptive plasticity), and in the complex changing environment, scores range from -4608 to +4608.

$$Score_c(P) = \sum_e^E |R_e \cap P_e| + |\neg R_e \cap \neg P_e| - |R_e \cap \neg P_e| - |\neg R_e \cap P_e| \quad (2)$$

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

Treatments Each of our three experiments consisted of seven treatments: one baseline treatment and six experimental treatments. Treatments differed only in the available mutation operators and the rates at which those operators were applied. Each treatment was designed to tease apart why gene duplications promote evolvability. Table 1 provides details about each treatment, and these details were shared across all three experiments.

The **baseline treatment** was used as a control; we used the results from this treatment as a baseline for evolvability with which we compared the results from all other experimental treatments. Mutation rates in the baseline treatment have been shown to facilitate both the evolution of complex boolean logic tasks, such as EQUALS, and the evolution of task regulation in Avida (Lenski et al., 2003; Lalejini and Ofria, 2016)

The **slip-duplicate treatment** allowed for mutation events that resulted in full code duplications via the slip-duplicate mutation operator. Duplications in this treatment preserved both the content and the structure of duplicated code. This allowed us to answer the following question: how important is it that gene duplications can exactly duplicate sequences in a genome in both content *and* structure?

The **slip-scramble treatment** allowed for mutation events (via the slip-scramble operator) that resulted in code duplications where the content of the duplicated code was preserved, but the structure of the duplicated code was not preserved. This, when compared with the slip-duplicate treatment, allowed us to answer the following question: is it that the duplication of particular instructions is important, regardless of their arrangement?

The **slip-random treatment** allowed us to answer the following question: is it the case that gene duplications promote evolvability because they result in the insertion of large, clustered mutations, regardless of what those mutations may be? The slip-random treatment (via the slip-random mutation operator) allowed for mutation events that could insert large, contiguous clusters of random instructions; one could also think of these mutations as maximally noisy duplications where neither the content or structure of the duplicated code is preserved.

The **slip-NOP treatment** allowed for mutations capable of inserting contiguous segments of blank 'genetic tape' (via the slip-NOP mutation operator) in the form of no operation instructions. This allowed us to answer the following question: how important is it that gene duplications provide evolution with an easy technique for increasing genome size?

The **slip-scatter treatment** helped us to tease apart whether or not gene duplications promote evolvability because they inflate the effective mutation rate, generating increased amounts of genetic variation. This treatment allowed for mutations that, when triggered, could insert many random instructions into random locations in a genome (via the slip-scatter mutation operator).

The **high mutation rate treatment** served a similar purpose to the slip-scatter treatment. However, instead of single mutation events (slip-scatter mutations) causing the insertion of many random instructions, we elevated the rates at which the copy instruction makes a random insertion or random deletion to result in approximately the same number of mutations per divide as in any treatment that includes a slip mutation operator. This allowed us to evaluate the importance of having an increased mutation rate due to large-scale, single-event mutations versus having a higher copy mutation rate.

In all experiments, organisms were limited to a minimum genome size (*i.e.* instruction sequence length) of 100[2]. We limited the population size to 3600 and seeded each experiment with an ancestral genotype capable only of self-replication. In both the static environment and simple changing environments, populations evolved for 200000 updates. In the complex changing environment, populations evolved for 400000 updates. We ran 100 trials of each treatment in each of our three environments.

Statistical Methods

In each of three environments (static environment, simple changing environment, complex changing environment), we performed experiments with six experimental treatments to be compared against a control. To determine if any of the treatments was significant within a set, we performed a Kruskal-Wallis test, applying a Bonferroni correction for the three different environments to keep the experiment-wise $\alpha = 0.05$. For an environment in which the Kruskal-Wallis test was significant, we performed Mann-Whitney U tests for each experimental treatment against the control, and applied a Bonferroni correction for the six such tests within each environment. All statistical analysis was conducted in R 3.2.3 (R Core Team, 2015).

Digital Organism Evaluation At the end of each trial, we extracted the final dominant (most abundant) genotype in the population, and we traced back that genotype's full ancestral lineage. We calculated the phenotypic match score for each final dominant organism and all of their ancestors. In both the simple and complex changing environments, however, the final dominant organism at the end of a trial was biased by the final environmental condition. The final dominant organism may have been well-adapted and abundant in the final environmental conditions, but it may not have been capable of surviving if the environmental conditions were allowed to change again. To avoid any edge effects associated

[2]In exploratory experiments, we found that, without enforcing a minimum genome size, slip mutations caused many lineages to quickly shrink in genome size because of inherent selection pressure for smaller genome size. Organisms became fast replicators but were then trapped on a local fitness optima, unable to evolve to perform complex computational tasks.

Treatment	Slip Mutation		Substitution rate per copy	Insertion and deletion rate per copy	Insertion and deletion rate per divide
	Operator	Rate per divide			
Baseline			0.0025		0.05
Slip-duplicate	Slip-duplicate	0.05	0.0025		0.05
Slip-scramble	Slip-scramble	0.05	0.0025		0.05
Slip-random	Slip-random	0.05	0.0025		0.05
Slip-NOP	Slip-NOP	0.05	0.0025		0.05
Slip-scatter	Slip-scatter	0.05	0.0025		0.05
High mutation rate			0.0025	0.0075	0.05

Table 1: Differences in mutation operators and rates among the seven treatments.

with this bias, instead of comparing final dominant organisms in the simple and complex changing environments, we compared the ancestors of these final dominant organisms that existed 1000 updates prior to the end of the experiment.

Results and Discussion

Does gene duplication promote evolvability?

We found that gene duplication promotes evolvability in all three environments. As shown in Figure 3, organisms evolved in the slip-duplicate treatment had significantly higher phenotypic match scores than organisms evolved in the baseline treatment across the static, simple changing and complex changing environments (two-tailed Mann-Whitney U tests, W = 562.5, 2702.5, 1540 respectively, Bonferroni-adjusted p values all $<< 0.0001$). This result suggests that the slip-duplicate operator promotes the evolution of complexity both in static environments that require the evolution of unconditional task performance (Boolean logic operations), and in dynamic environmental conditions that require the evolution of regulatory mechanisms that alter expression based on current environmental conditions. These results, as expected, support the existing literature on the capacity of gene duplications to promote evolvability (Koza, 1995; Zhang, 2003; Teichmann and Babu, 2004).

What aspects of gene duplication promote evolvability?

Across all three environments we found that the most important aspect of gene duplications is the capacity to duplicate meaningful information in the genome. As shown in Figure 3, only organisms that evolved in the slip-duplicate and slip-scramble treatments had significantly higher phenotypic match scores than organisms evolved in the baseline treatment. Organisms evolved in all other experimental treatments were either not significantly different from the baseline treatment or significantly worse than the baseline treatment.

Of the five types of slip mutation operators used (see Figure 2), only the slip-duplicate and slip-scramble operators were capable of duplicating meaningful information in a genome. Slip-scramble mutations maintain the content but not the ordering of duplicated instruction sequences; thus, they can duplicate information about what instructions make-up already successful genetic sequences but do not maintain the particular ordering of those instructions. Slip-duplicate mutations are the full form of gene duplication, able to exactly duplicate the ordered instruction make-up of existing genetic sequences. In contrast to the slip-scramble and slip-duplicate operators, all other slip mutation operators do not duplicate information about instruction sequences already present in the genome.

Because the slip-scramble treatment was significantly better at promoting evolvability than the baseline treatment, we can conclude that the duplication of functional building blocks is valuable no matter how they are arranged when duplicated. However, is it important to preserve both the contents and ordering of duplicated sequences?

Is more information better? From Figure 3, organisms evolved in the slip-duplicate treatment had higher phenotypic match scores than organisms evolved in the slip-scramble treatment, implying that the additional information in a gene duplication event does promote evolvability. To confirm this relationship and reduce multiple comparison issues with statistical tests, we re-ran 100 new trials of both the slip-duplicate and slip-scramble treatments and compared the phenotypic match scores of the evolved organisms. In all three environments, organisms that evolved in the slip-duplicate treatment had significantly higher phenotypic match scores than organisms evolved in the slip-scramble treatment (two-tailed Mann-Whitney U tests, W = 4305, 4028.5, 3621.5 respectively, Bonferroni-adjusted p values 0.0109, 0.0222, 0.0017, respectively). This result supports that duplicating both the content and structure of existing genetic code is an important factor in their ability to promote evolvability.

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

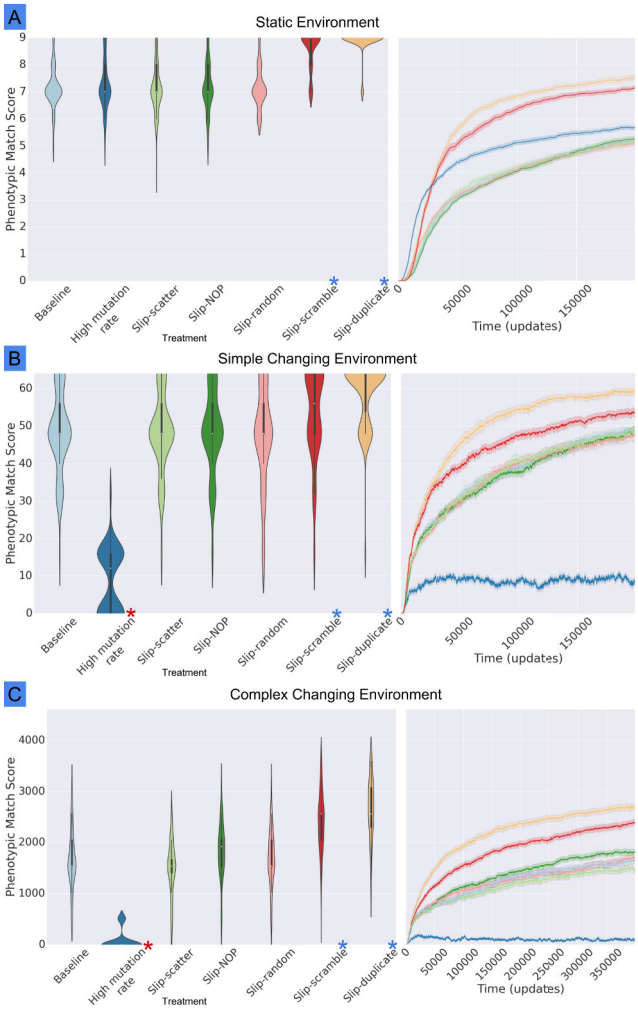

Figure 3: Experimental results from all three environments: A) static environment, B) simple changing environment, and C) complex changing environment. The violin plots for each environment indicate the phenotypic match scores for final dominants in the static environment and for the ancestors of final dominants that existed 1000 updates prior to the end of a trial in the simple and complex changing environments. Each time series shows the phenotypic match scores for the lineages of final dominant organisms over time. The colors in each time series correspond to the colors in the violin plots. Treatments that have significantly higher phenotypic match scores than the baseline treatment are marked with blue *, and treatments that have significantly lower phenotypic match scores than the baseline treatment are marked with red *.

High mutation rate inhibits the evolution of regulation in changing environments

In addition to our results on why gene duplications promote evolvability, our experiments indicate that high mutation rates inhibit the evolution of regulation in changing environments. In both the simple and complex changing environments, organisms evolved in the high mutation rate treatment had significantly lower phenotypic match scores than organisms evolved in the baseline treatment, which can be seen in Figure 3 (two-tailed Mann-Whitney U tests, W = 9958 and 9944, respectively, Bonferroni-adjusted p values both $<< 0.0001$). A similar result was found in (Lalejini and Ofria, 2016). Although higher mutation rates increase genetic variation, most mutations have deleterious effects (Schlichting and Smith, 2002). Thus, higher mutation rates may increase the difficulty of maintaining the genetic machinery necessary for complex regulation.

If elevated mutation rates inhibit the evolution of regulation and all experimental treatments with slip mutation operators have an effectively higher mutation rate than the baseline treatment, why do we see this effect only in the high mutation rate treatment? The rates of insertions and deletions per instruction copied in the high mutation rate treatment were selected to, on average, result in approximately the same number of mutations per divide as in our treatments with slip mutations. Yet, only organisms evolved in the high mutation rate treatment had significantly lower phenotypic match scores than organisms evolved in the baseline treatment in the changing environments; all other experimental treatments were either significantly better or not significantly different than the baseline treatment.

Why is it that *only* the high mutation rate treatment inhibited evolvability in the changing environments? One possibility stems from the higher mutation load imposed on populations by the high mutation rate treatment. Slip mutations result in large mutational events that affect few offspring, whereas the elevated rate of copy mutations in the high mutation rate treatment results in an increased number of smaller mutational events spread across many offspring. Thus, many offspring in the high mutation rate treatment are subject to the increased rate of deleterious mutations that results from the higher rates of copy insertions and copy deletions. This is in contrast to treatments with slip mutations where relatively fewer offspring are subjected to large mutational events, which results in the concentration of deleterious mutations in fewer offspring relative to the high mutation rate treatment. Further analysis is needed to confirm that this is, indeed, the case.

Conclusion

In this work, we investigated both if and how gene duplications promote evolvability in two qualitatively different contexts: 1) in a static environment that requires the evolution of unconditionally expressed complex traits, and 2) in changing environmental conditions that require the evolution of regulatory mechanisms capable of altering which operations are expressed as a function of current environmental conditions. Our results show that, indeed, gene duplications promote evolvability in both static and changing environments. We found evidence that the most important aspect of gene duplications for promoting evolvability is their capacity to duplicate existing genetic information within a

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

genome. Additionally, we found that the capacity to duplicate multiple types of information is beneficial; mutation operators that duplicated both the content and structure of genetic sequences promoted evolvability more than operators that duplicated content but not structure.

Broadly speaking, gene duplications can facilitate significant jumps in a fitness landscape in many different ways. For example, when genes are duplicated, processes like subfunctionalization or neofunctionalization may occur, allowing populations to more easily cross fitness valleys and escape local optima. In the domain of evolutionary computation, the more we understand about why gene duplication is so important in promoting evolvability in different scenarios, the more we can customize our mutation operators to emphasize those factors. These results suggest that when designing gene duplication operators for evolutionary computation systems, we should try to maximize the amount of information the operators are capable of duplicating.

The results presented here motivate several future studies. Performing deeper analyses of lineages evolved with gene duplications would allow us to more specifically identify how much different types of high-level processes are contributing to evolvability post-duplication event. Can we identify examples of subfunctionalization or neofunctionalization, and if so, can we identify their contributions to promoting the evolution of complex features? Or, would we find that duplications that are most likely to successfully sweep a population are those that have the highest information content? Answers to these questions will allow us to better understand the role that gene duplication plays in natural evolution and could play in computational systems.

Acknowledgements

We would like to acknowledge Jeffrey E. Barrick and David M. Bryson for laying the groundwork for this research. We also thank Rosangela Canino-Koning and Emily Dolson for their comments on this manuscript and the Digital Evolution Laboratory for thoughtful discussions, ideas and support. This research was supported in part by the BEACON Center for the Study of Evolution in Action and by Michigan State University through the computational resources provided by the Institute for Cyber-Enabled Research.

References

Blount, Z. D., Barrick, J. E., Davidson, C. J., and Lenski, R. E. (2012). Genomic analysis of a key innovation in an experimental Escherichia coli population. *Nature*, 489(7417):513–518.

Bzymek, M. and Lovett, S. T. (2001). Instability of repetitive DNA sequences: The role of replication in multiple mechanisms. *Proceedings of the National Academy of Sciences*, 98(15):8319–8325.

Calabretta, R., Nolfi, S., Parisi, D., and Wagner, G. P. (1998). Emergence of functional modularity in robots. *From animals to animats*.

Calabretta, R., Nolfi, S., Parisi, D., and Wagner, G. P. (2000). Duplication of modules facilitates the evolution of functional specialization. *Artificial life*, 6(1):69–84.

Clune, J., Ofria, C., and Pennock, R. T. (2007). Investigating the emergence of phenotypic plasticity in evolving digital organisms. In *Advances in Artificial Life*, pages 74–83. Springer.

Crow, K. D. and Wagner, G. P. (2006). What is the role of genome duplication in the evolution of complexity and diversity? *Molecular biology and evolution*, 23(5):887–892.

Hu, T. and Banzhaf, W. (2010). Evolvability and Speed of Evolutionary Algorithms in Light of Recent Developments in Biology. *Journal of Artificial Evolution and Applications*, 2010(4):1–28.

Koza, J. R. (1995). Gene duplication to enable genetic programming to concurrently evolve both the architecture and work-performing steps of a computer program. *IJCAI (1)*.

Lalejini, A. and Ofria, C. (2016). The evolutionary origins of phenotypic plasticity. In *Proceedings of the 15th International Conference for Artificial Life*, pages 372–379.

Lenski, R. E., Ofria, C., Pennock, R. T., and Adami, C. (2003). The evolutionary origin of complex features. *Nature*, 423(6936):139–144.

Magadum, S., Banerjee, U., Murugan, P., Gangapur, D., and Ravikesavan, R. (2013). Gene duplication as a major force in evolution. *Journal of genetics*, 92(1):155–161.

Ofria, C., Bryson, D. M., and Wilke, C. O. (2009). Avida: A software platform for research in computational evolutionary biology. In *Artificial Life Models in Software*, pages 3–35. Springer.

Ohno, S. (1970). *Evolution by Gene Duplication*. Springer.

R Core Team (2015). R: A language and environment for statistical computing.

Ryan, C., Collins, J. J., and Neill, M. O. (1998). Grammatical evolution: Evolving programs for an arbitrary language. In *Genetic Programming*, pages 83–96. Springer, Berlin, Heidelberg, Berlin, Heidelberg.

Sawai, H. and Adachi, S. (1999). Genetic algorithm inspired by gene duplication. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1, pages 480–487. IEEE.

Sawai, H. and Adachi, S. (2000). A comparative study of gene-duplicated gas based on pfga and ssga. In *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*, pages 74–81. Morgan Kaufmann Publishers Inc.

Schlichting, C. D. and Smith, H. (2002). Phenotypic plasticity: linking molecular mechanisms with evolutionary outcomes. *Evolutionary Ecology*, 16(3):189–211.

Schmitt, K. (2005). *Using gene deletion and gene duplication in evolution strategies*. ACM, New York, New York, USA.

Teichmann, S. A. and Babu, M. M. (2004). Gene regulatory network growth by duplication. *Nat Genet*, 36(5):492–496.

Teichmann, S. A., Park, J., and Chothia, C. (1998). Structural assignments to the Mycoplasma genitalium proteins show extensive gene duplications and domain rearrangements. *Proceedings of the National Academy of Sciences*, 95(25):14658–14663.

Wagner, A. (2008). Gene duplications, robustness and evolutionary innovations. *BioEssays*, 30(4):367–373.

Wagner, G. P., Amemiya, C., and Ruddle, F. (2003). Hox cluster duplications and the opportunity for evolutionary novelties. *Proceedings of the National Academy of Sciences*, 100(25):14603–14606.

Zhang, J. (2003). Evolution by gene duplication: an update. *Trends in Ecology & Evolution*, 18(6):292–298.