Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

John Kirkham Date

Effects of Mutation and Recombination on the Rate of Evolution on a Time Varying Mt. Fuji Fitness Landscape

By

John Kirkham

Master of Science

Physics

Ilya Nemenman

Advisor

Stefan Boettcher

Committee Member

David Cutler

Committee Member

Fereydoon Family

Committee Member

Eric Weeks

Committee Member

Accepted:

Lisa A. Tedesco, Ph.D.

Dean of the James T. Laney School of Graduate Studies

Date

Effects of Mutation and Recombination on the Rate of Evolution on a Time varying Mt. Fuji Fitness Landscape

By

John Kirkham

B.Sc., Rhodes College, 2010

Advisor: Ilya Nemenman, Ph.D., Princeton University, 2000

An abstract of

A thesis submitted to the Faculty of the

James T. Laney School of Graduate Studies of Emory University

in partial fulfillment of the requirements for the degree of

Master of Science

in Physics

2013

Abstract

Effects of Mutation and Recombination on the Rate of Evolution on a Time varying Mt. Fuji Fitness Landscape

By John Kirkham

Recombination confers many positive attributes to populations of organisms, who implement it. The discovery of recombination in bacteria changed our understanding of them. Interactions of recombination, mutation and selection have been considered in diploid organisms, but this still leaves plenty of questions open about haploid organisms. Though it is commonly believed that recombination exists primarily to repair DNA and possibly increase variability, the interaction of recombination and mutation has not been well considered. In this thesis, a Wright-Fisher model of a population evolving on a Mt. Fuji Landscape will be explored with different values of selection, mutation, and recombination. Ultimately, the model will incorporate periodic fluctuations that completely randomize the peak of the Mt. Fuji Landscape. The goal of this thesis is to answer the following question: What are the optimal mutation and recombination rates to most rapidly improve a population's fitness given a fixed period and selection strength? It will be demonstrated that the recombination has no optimum (more recombination is always better). Further, it will be demonstrated that the mutation rate does have an optimum that is strongly affected by period and possibly weakly affected by selection.

Effects of Mutation and Recombination on the Rate of Evolution on a Time varying Mt. Fuji Fitness Landscape

By

John Kirkham

B.Sc., Rhodes College, 2010

Advisor: Ilya Nemenman, Ph.D., Princeton University, 2000

A thesis submitted to the Faculty of the

James T. Laney School of Graduate Studies of Emory University

in partial fulfillment of the requirements for the degree of

Master of Science

in Physics

2013

Contents

List of Figures

1 Introduction

Mutation and recombination play an intricate role in bacterial evolution. Both mutation and recombination have some similarities in specific changes they generate in a bacterium's genetic code. However, mutation and recombination differ in their manner of action and their effect on the population. To better understand these differences, both the causes of mutation and recombination as well as the manner they are carried out must be understood.

Traditionally, the argument has been that mutations are spontaneous.[59] However, there is some suggestion that mutation may be adaptively controlled.[8, 5, 84, 88, 24, 20] There are three main types of mutation: substitutions, insertions, and deletions.[101, 50] Substitutions happen when base pairs are changed from one type to another. Insertions add a new base. Deletions are, as one would expect, removal of base pairs. Both insertion and deletions can cause frameshift, which result in sections of DNA being offset due to the missing or added base pairs that precede them. Though most mutations are neutral and some are deleterious, advantageous mutations can and do occur as well.

There are three main types of recombination: transformation, conjugation, and transduction.[63, 23, 94, 58, 73] In transformation, a bacterium cell picks up DNA from its environment, in particular, from dead or secreting bacterium. In conjugation, a bacterium pulls another bacterial cell close, after which a bridge is built (or the existing apparatus is used) to transfer DNA from the connecting cell to the connected cell.[2, 14, 98] Lastly, in transduction, a bacteriophage (virus) infects the cell and attempts to insert its genetic code into the bacterium. Once the DNA is inside the bacterium, it must be integrated into the bacterial chromosome or it will be degraded.[9, 94] The main point is that recombination constructs new genetic material from different pieces already present in the population.

In the presence of selection, mutation and recombination constitute the known mechanisms to evolve populations of bacteria. The mechanisms and the effects of spontaneous mutation are largely understood.[24, 26, 25, 42, 36, 59, 66] The mechanisms of bacterial recombination are mostly understood as well as the features they are able to confer like antibiotic resistance.[10, 21, 94, 2, 9, 14, 23, 52, 58, 101, 98, 82] Though the raison d'être of recombination is not well known, there are several hypotheses as to why recombination would be beneficial.[30, 4, 96, 76, 64, 99, 41]

One such hypothesis is that recombination can combine several favorable alleles that otherwise occur independently and would be slowly fixed (a.k.a the Hill-Robertson effect).[43, 32, 67, 41] Another such hypothesis is that without recombination a well-adapted population will accumulate deleterious mutations until those with no deleterious mutations are eventually lost resulting in an ever increasing minimum number of deleterious mutation in the population.[68, 41] These two hypotheses have been shown to be very much related.[30, 41] Thus, a reformulated hypothesis would be that recombination is useful in its ability to remove linkage disequilibrium between different alleles. In other words, recombination would serve mainly to repair the existing DNA. However, removing linkage could have undesirable results like breaking up favorable alleles or possibly combining many deleterious mutations.[64, 76]

Another hypothesis is that recombination exists to enable faster adaptation in a changing environment (i.e. the Red Queen hypothesis).[40, 64] Normally, this hypothesis is posited in terms of a population resisting parasites, which may be a reasonable way to look at bacterial resistance of antibiotics amongst other abiotic threats. In general, the hypothesis is that recombination is a way to add variation to the population (like mutation). Having recombination present simply to add variation to the population doesn't seem likely due to concerns of breaking up favorable alleles.[64, 76] Though, it does not mean that recombination does not play a role in adding variation.[64] In fact, recombination could function as a repair mechanism and as a source of variation.[99]

Despite the argument that recombination exists primarily as a repair mechanism, little has been done to look at the interaction of recombination and mutation in the presence of selection. Though some models of recombination in the presence of a changing environment have been considered, mutation has not been seriously considered as well. The interaction of mutation and recombination will be explored on a Mt. Fuji fitness landscape. Initially, the landscape will be fixed. Later on, the landscape's peak will be move to a new random point in genome space after a fixed

period has elapsed.

In this thesis, the follow question will be explored. What are the optimal mutation and recombination rates to most rapidly improve a population's fitness given a fixed period and selection strength? It will be demonstrated that the recombination has no optimum (more recombination is always better). Further, it will be demonstrated that the mutation rate does have an optimum that is strongly affected by period and possibly weakly affected by selection.

2 Methods

The model used is similar to a Wright-Fisher process. A population of fixed size is taken as the first generation, which is formed using some generation rule. From it a new generation of the same size is formed in a synchronous fashion using a selection rule. In each case, the previous generation is completely replaced by the subsequent one. We choose to hold the generation size constant at 1000 individuals.

The selection rule used determines the relative likelihood of a parent to have offspring, which is proportional to its fitness. In the infinite population limit, a parent's fitness will determine the relative number of offspring it will have. However, the population size isn't quite taken to infinity as it would eliminate one of the key features of the model, extinction, which is preferable when the individual's fitness is relatively low compared to the rest of the population. Similarly, if an individual has a very high fitness relative to the rest of the population, it will likely become fixed in the population. In general, this is accomplished by choosing randomly the parent(s) favoring those with higher probability until all offspring needed to maintain the population size are created. Just using the generation and selection rule yields the trivial result.

In other words, the selection rule simply eliminates the population variance until ultimately one type of individual is left. If the selection rule is trivial, then this results in pure genetic drift. In our model, every individual has a sequence composed of bits (1 and 0). We choose to hold the sequence length constant at 100 bits. Simulations occur on a Mt. Fuji style landscape; where, the peak is represented as an ideal sequence. Differences from the ideal sequence are measured using the Hamming Distance[85]. Fitness (1) can easily be calculated knowing that s (S in the graphs) represents selection, d_i represents the Hamming Distance of the i-th individual from the ideal sequence, and L represents the length of all sequences, which, as mentioned, is proportional to the probability of selecting the i-th individual.

$$P_i \propto F_i = e^{s\cdot(L-2\cdot d_i)} \tag{1}$$

Determining the actual probability only requires a trivial normalization (2). Note that every term will have a factor of $e^{s\cdot L}$, which will cancel in the normalization, which means only the distance away from the ideal sequence and the strength of selection determines the probability of selection. In other words, the sequence length does not directly factor in to the probability; however, the distribution of sequences in the population will be affected by sequence length.

$$P_i = \frac{e^{-2\cdot s\cdot d_i}}{\sum_j e^{-2\cdot s\cdot d_j}} \tag{2}$$

As discussed in the introduction, mutation is spontaneous. So, mutation will be modeled as uncorrelated point mutations that occur with a per site probability of μ (M in the graphs). Instead of treating mutation with a Bernoulli distribution A.2, where each bit is tested for a mutation. It will be asked when the next mutation occurs, which will be done using a geometric distribution A.4. Algorithmically, mutation will be accomplished by treating all sequences of all individuals in the simulation as one long sequence and drawing an index offset from a geometric distribution to determine the next site to mutate. The geometric distribution is used to determine when mutations should occur by using the follow probability distribution (3). Note that this distribution finds when the next mutation will occur relative to the current position (i). To avoid re-mutating the same position, 1 will be added to the last mutated position making the current position (i) just after the last mutated one.

$$P_i = (1-\mu)^i \cdot \mu \tag{3}$$

This ensures there will be no extra random number generation in the cases of no mutation. Representing the spins as bits, the diagram below shows how one typical point mutation would be carried out on one individual from one generation to the next Figure 1 on page 4.

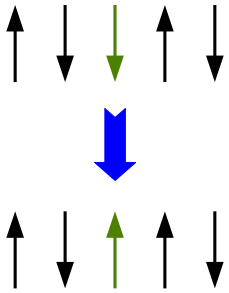

Figure 1: Mutation Example

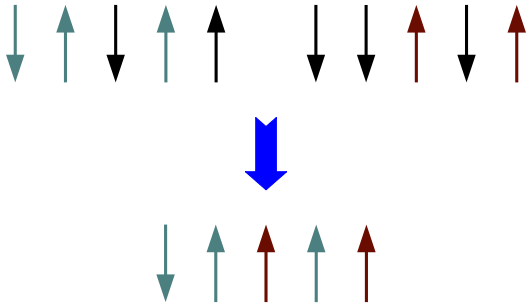

Figure 2: Recombination Example

Recombination (R in graphs), on the other hand, samples large pieces from the population. Thus, recombination will be treated as replacing small sequences in an existing individual's sequence with pieces from another individual's sequence. Recombination events will only occur between two distinct individuals when forming a new generation. The length of the small sequences will be geometrically distributed with a mean of 5, which will essentially use the geometric distribution A.4 determined by its (40); however, as the trivial sequence length will be explicitly disallowed, the mean will be treated as 4 and a 1 will be added to all values drawn from the distribution making the average actually 5. The two individuals will have their sequences intertwined as shown in the diagram Figure 2 on page 4. After recombination happens, mutation will also be allowed to happen on the recombined sequence.

One way to ensure that two distinct individuals are picked for recombination is by building a distribution of individuals weighted with the fitness of each and then rebuilding a distribution without the first picked individual. However, this turns out to be suboptimal. This is due to the amount of time spent on individuals with low probabilities particularly in terms of normalization. A far better strategy is to build a Huffman Tree using the weights to determine the tree. The details of the algorithm to build a Huffman Tree are widely available and so will not be explained here. The only relevant details about it are that the average weighted path length is minimized and that every internal node has two children. A sample Huffman Tree Figure 3 on page 5 shows a couple of letters and their weights. Picking between two branches can be done with a random Bernoulli (more accurately with a uniformly distributed real that is successively binned for as long as it still has enough precision to be binned). When an object is picked as demonstrated by picking D Figure 4 on page 6, then it's weight and the weights of its parents are adjusted by removing the weight of D from all of them. This ensures that it will not be drawn again unless the old weights are restored (they are kept in memory for such as case). This approach makes picking without redraw an $O(\lg(N))$ (due to re-weighting) as opposed to an $O(N)$ operation (due to insertion and normalization). Despite the $O(N \cdot \lg(N))$ initialization cost of the Huffman Tree Distribution (as opposed to $O(N)$ of the other method), the Huffman Tree Distribution pays off by dealing with multiple redraws efficiently.

In order to keep properly compartmentalized code, notification systems were used. In particular, this was used for updating fitness and time. As it is not a bacterium's job to know its

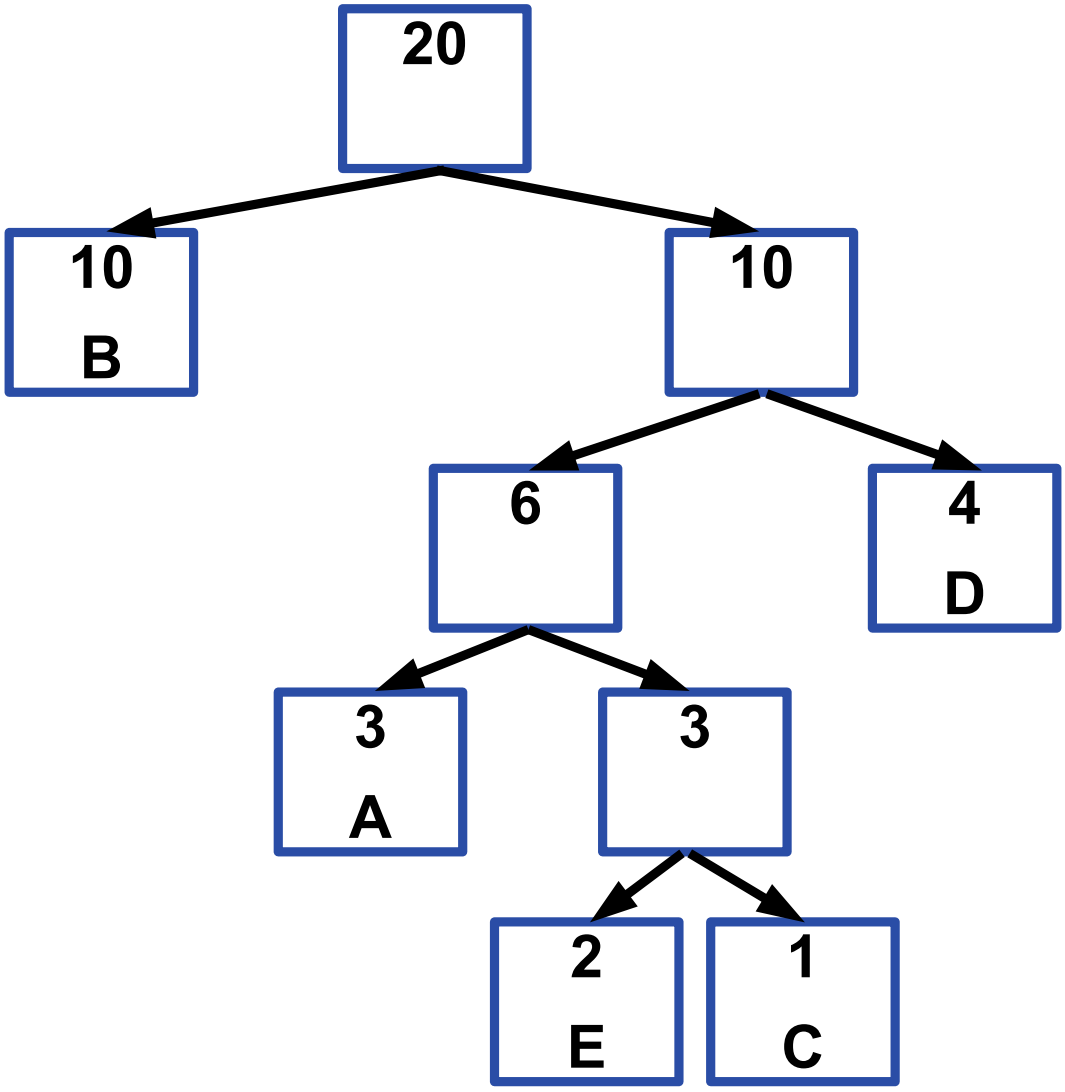

Figure 3: Initial Huffman Distribution

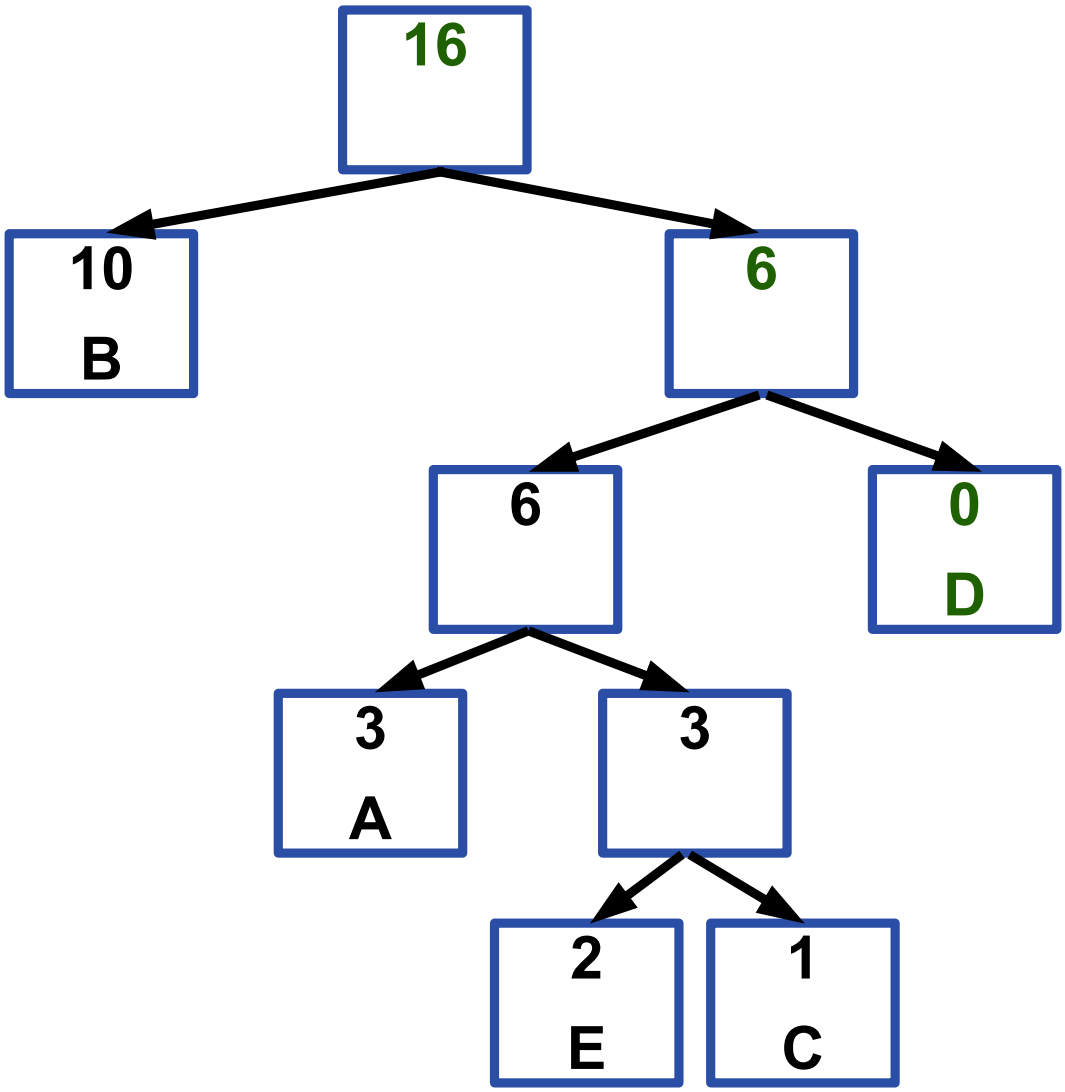

Figure 4: Huffman Distribution after 1 draw (D)

environment (certainly it may have some perceptions, but these aren't relevant in this problem), it should not be in charge of assigning its fitness, the environment should. However, the fitness of a bacterium should be stored in its object. So, to deal with this, a bacterium must submit their sequence and something to store their fitness in. Though, after it is updated, other components like the colony will want to perform operations to update their state after the change in fitness. As it is also not the colony's job to update this parameter, it needs to register itself and its bacteria for updates and then do something once notified of the fitnesses being updated. Similarly, the environment is in charge of time not the cell, colony, etc. However, everything depends on changes in time. Similarly, they all register for updates regarding the time and once those updates happen they respond. This includes the fitness function; though, part of the environment, it itself is not in charge of issuing time updates. Instead, the fitness function merely calculates the fitness of a bacterium at a point in time. This allows simple extensions to the model. For instance, mutation or recombination could be made time dependent easily, which could allow modeling of a mutagen being released in the environment or recombination increasing when the environment makes a large unfavorable change.

Environmental changes will be modeled by creating a totally random ideal sequence every period. In other words, there will be no inherent advantage to be well adapted to the previous ideal sequence. A couple of different period lengths will be sampled. The variable used to denote period length will be P.

The number of cells needed is quite large, it does not make sense to track individual properties of cells. Instead, just as in statistical mechanics, different moments of the distribution must be tracked. For instance, it is important to know the average distance from individuals to the ideal sequence (a.k.a. the peak of the fitness function). This will be known as the ideal sequence distance average. It will be calculated by finding the Hamming Distance of each sequence from the ideal sequence used in the fitness function and then will be summed over the population and divided by the population size. This can then be tracked over time to determine how close a population is to the target. It can also be used to determine how populations with different properties vary.

Another important quantity to determine is the spread of the population or its population variance. This population variance will also be known as the average sequence variance. In order to find the average sequence variance, an average sequence will be calculated by finding the probability that any particular position in the sequence will have a 1 based on the sequences in the population. In other words, the average sequence can be found by summing the sequences in the population and dividing by the population size. Once the average is found, the average sequence variance can be found in the standard way using this average.

Thus, both the average sequence variance and ideal sequence distance average are average scalar quantities that describe the population over time. In some cases, the average fitness can also be found, but this doesn't give much more information than the ideal sequence distance average as they respond in somewhat opposite fashions. In other words, as the ideal sequence distance average becomes small (close to the ideal sequence) the average fitness will become large (most individuals will be likely to have progeny). Similarly, if the ideal sequence distance average becomes large (far from the ideal sequence) the average fitness will become small (only a few individuals will have progeny). In addition to finding these moments, several trials will be averaged together (with the same parameters) in order to get a more accurate statistical representation. All of the graphs that follow are the product of averaging 50 trials per parameter. Several cases of different populations and their resulting average sequence variance and ideal sequence distance average are consider in the appendix Section B.

3 Results

We first explore a series of graphs with no recombination and a variety of different mutation rates for a given selection strength to verify consistency with Fisher's Fundamental Theorem and to set a basis for later discussion. The following graphs Figure 5 on page 9 shows constant selection (S) of 0.01 and recombination rate (R) of 0 with varying mutation rates (M). In particular, what is plotted is a trial average of the ideal sequence distance on the top and trial average of the average sequence variance on the bottom. It appears that in this graph Figure 5 on page 9 that the ideal sequence distance average is initially 50 and the average sequence variance is initially at 25, which corresponds to mathematical derivations of the initial population distributionB.1.

In all cases, there is a rapid drop of the sequences to around 35 (neglecting the extremely high mutation rates of $M = 0.5$ and $M = 0.05$ where this is not possible due to the large addition of variance from mutation). As this happens in all cases, including when there is no mutation, this cannot be a consequence of the mutation rate. Instead, this must be a consequence of the variance in the initial population. As a result, it falls under the case mentioned before (76). This means that one best individual (or perhaps a couple with the same or similarly high fitness) is (or are) being selected. Furthermore the average sequence variance Figure 5 on page 9 is also dropping off quite rapidly, which demonstrates the population is very similar to the best individual picked verifying the earlier calculation (76) accurately predicts how well any colony can initially improve. It is unclear at this point what role, if any, selection has to play in this rapid drop. One could speculate that it simply effects how rapid the drop is.

After the initial drop, the higher the mutation rates are more rapidly is approaching the ideal sequence. Though, this is hard to see in the extremely high mutation rates of $M = 0.5$ and $M = 0.05$. This is possible because higher mutation rates cause higher variance, which gives selection better options in terms of fit individuals for the next generation to spawn from. Though, it is worth mentioning that the higher mutation rates get stuck farther from the ideal sequence, which is again due to the variance created by the higher mutation rate. Clonal interference may be playing a role here as well.

Though, the time scales of different mutation rates are clear in the previous graph Figure 5 on page 9. It is difficult to determine, which mutation rates are still reducing their ideal sequence distance average near the end of the time series. By looking at a log-log plot of the same data, it can be seen that even some of the lowest mutation rates like $M = 5 \cdot 10^{-6}$ are still improving albeit less rapidly Figure 6 on page 10. Though, this lower rate of improvement for $M = 5 \cdot 10^{-6}$ can be linked to its lower average sequence variance Figure 6 on page 10. It is still unclear whether $M = 5 \cdot 10^{-7}$ may improve further. Though, a mutation rate of that magnitude will create a new mutation on average every 20 time steps.

It is also worth noting that divergence in the ideal sequence distance average has begun to occur around ~ 10 time steps. Afterwards, the average sequence variance begins to diverge around ~ 20 time steps or a little later. As mutation adds variance to the population, this opposes selection, which tries to remove variance. So, it isn't surprising that the variance doesn't drop immediately.

When looking at a higher selection for the same parameters, one would expect that the population variance would drop off more quickly to a lower variance. This would correspond with rapid adaptation followed by a slower approach to the ideal sequence. This is shown with $S = 0.05$, for the same parameters, in the ideal sequence distance average and average sequence variance Figure 7 on page 11. These graphs show are intuition is correct in terms of the differences in selection. So, it can be understood that higher selection forces the population through tighter bottlenecks (fewer individuals picked for reproduction that are closer to the ideal sequence), which initially improve the population rapidly; however, as this process continues, it results in a population that has less variance, which makes it difficult for selection to make more improvements to the population. Instead, selection must wait for mutation to create variance for it to act on. The magnitude of this selection is quite apparent on the mutation rate of $M = 5 \cdot 10^{-2}$. Previously this higher mutation rate, remained about half way from the ideal sequence. Due to the higher selection, it now also drops to the best individual (or a couple very good individuals) in its population. This drop in ideal sequence distance average is also matched by a noticeable drop in average sequence variance. Similarly, the

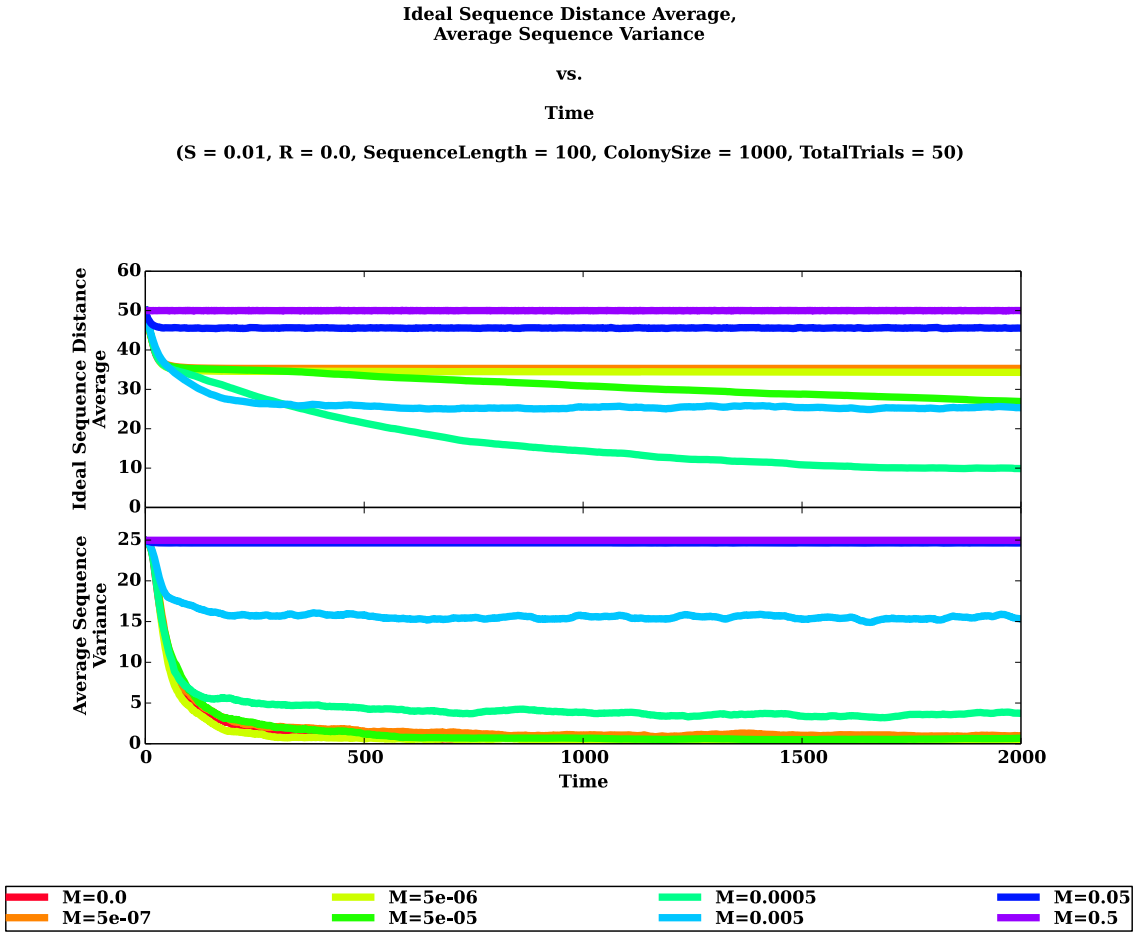

Figure 5: Ideal Sequence Distance Average and Average Sequence Variance vs. Time (Selection (S) held at $S = 0.01$, Recombination rate (R) held at 0, Mutation (M) ranging from $M = 5 \cdot 10^{-7}$ to $M = 5 \cdot 10^{-1}$ by orders of magnitude, includes 0 also)

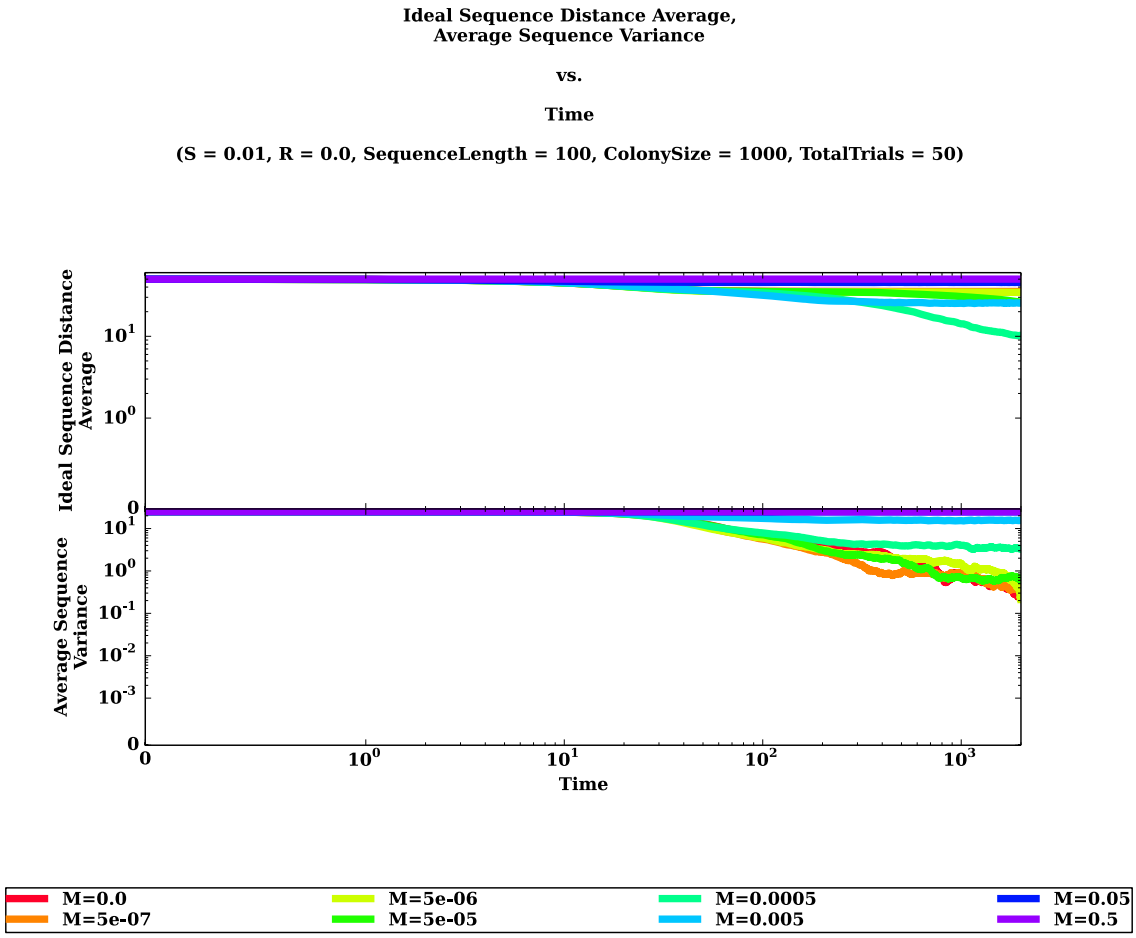

Figure 6: Log(Ideal Sequence Distance Average) and Log(Average Sequence Variance) vs. Log(Time) (Selection (S) held at $S = 0.01$, Recombination rate (R) held at 0, Mutation (M) ranging from $M = 5 \cdot 10^{-7}$ to $M = 5 \cdot 10^{-1}$ by orders of magnitude, includes 0 also, Log(ideal sequence distance average) linearized between 10^0 and 0 and Log(average sequence variance) linearized between 10^{-3} and 0))

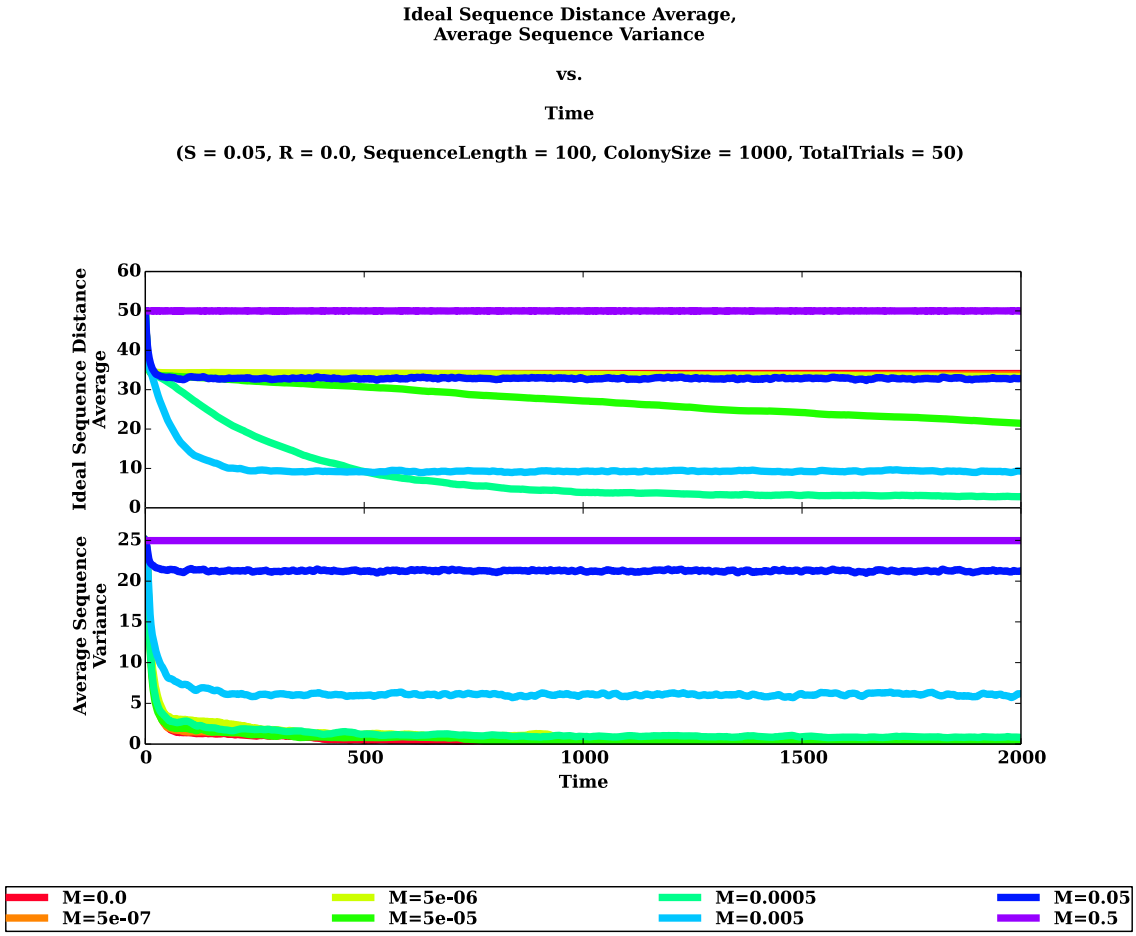

Figure 7: Ideal Sequence Distance Average and Average Sequence Variance vs. Time (Selection (S) held at $S = 0.05$, Recombination rate (R) held at 0, Mutation (M) ranging from $M = 5 \cdot 10^{-7}$ to $M = 5 \cdot 10^{-1}$ by orders of magnitude, includes 0 also)

mutation rate of $M = 5 \cdot 10^{-3}$ is affected. However, the magnitude of the drop in variance is not completely clear in the lower mutation rates.

Looking at log-log graphs of the same higher selection data Figure 8 on page 12, it is clear that mutation rates of $M = 5 \cdot 10^{-4}$ and lower end with average sequence variances of less than 1 (most are closer to 0.1) as opposed to close to or greater than 1 in the lower selection case. It is also worth noting that divergence in the ideal sequence distance average has begun to occur around ~ 2 time steps. Afterwards, the average sequence variance begins to diverge around ~ 5 time steps if not a little earlier. Both of these divergences occur significantly earlier for higher selection and mutation than lower selection and mutation by about a factor of 4 or 5. The concepts of how variance and the rate of improvement of a population are very important not just with mutation, but also with recombination.

In order to understand how recombination effects the population over time, it is important to look at it in isolation. So, taking $S = 0.01$ and $M = 0$, the ideal sequence distance average and average sequence variance Figure 9 on page 13 can be seen. In all case, the ideal sequence distance average flattens out quickly. It seems that the average sequence variance is dropping to zero; however, that cannot be clearly determined in the linear plots.

Looking at the log-log plot Figure 10 on page 15, it can be seen that the ideal sequence distance for all recombination rates has flattened out. The average sequence variance has dropped

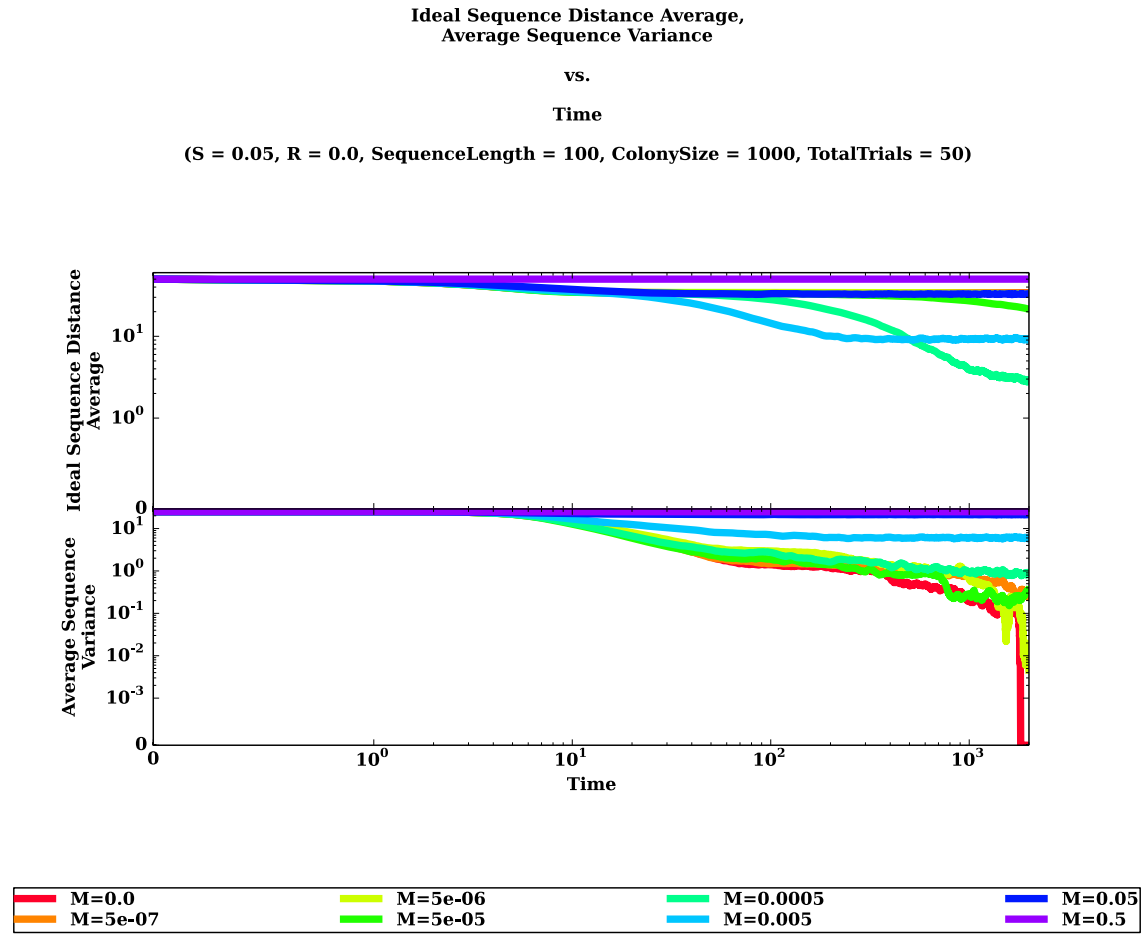

Figure 8: Log(Ideal Sequence Distance Average) and Log(Average Sequence Variance) vs. Log(Time) (Selection (S) held at $S = 0.05$, Recombination rate (R) held at 0, Mutation (M) ranging from $M = 5 \cdot 10^{-7}$ to $M = 5 \cdot 10^{-1}$ by orders of magnitude, includes 0 also, Log(ideal sequence distance average) linearized between 10^0 and 0 and Log(average sequence variance) linearized between 10^{-3} and 0))

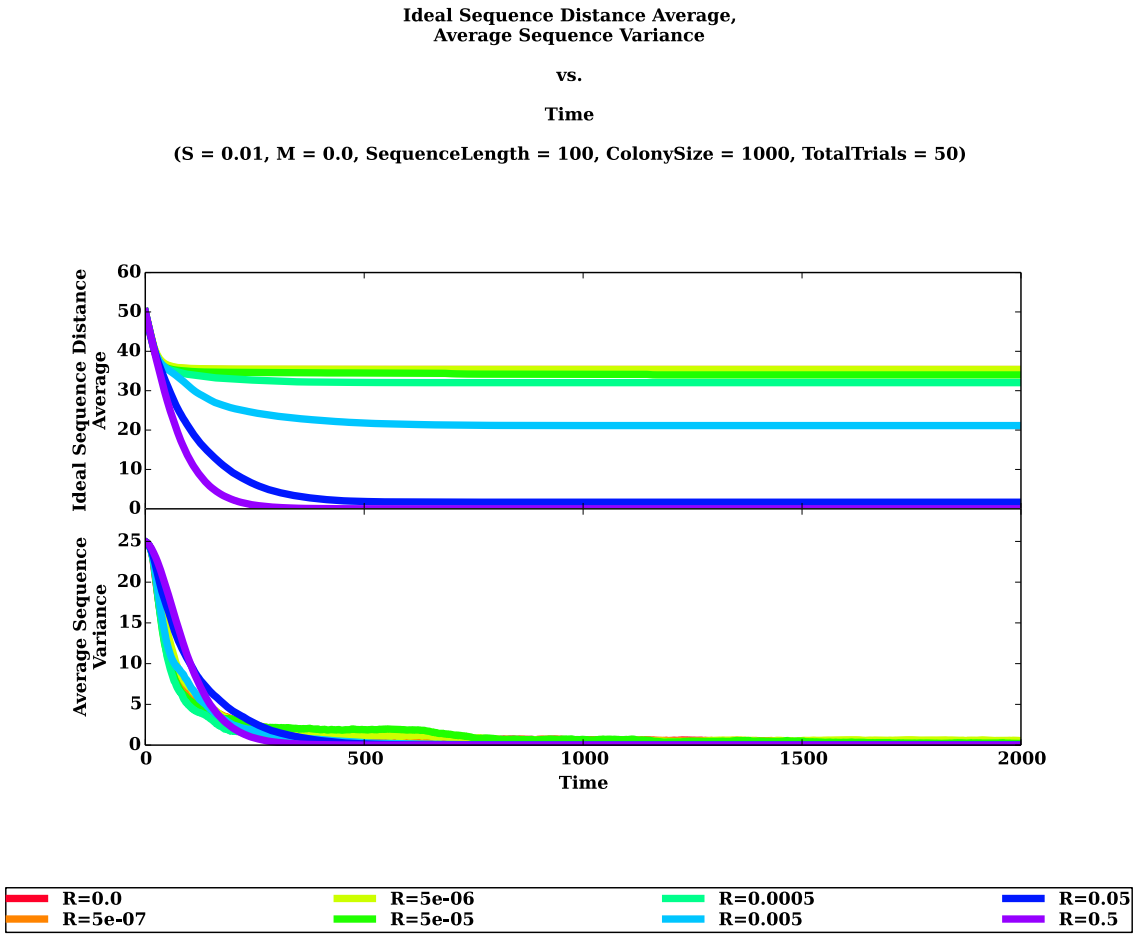

Figure 9: Ideal Sequence Distance Average and Average Sequence Variance vs. Time (Selection (S) held at $S = 0.01$, Recombination rate (R) ranging from $R = 5 \cdot 10^{-7}$ to $R = 5 \cdot 10^{-1}$ by orders of magnitude, includes 0 also, Mutation (M) held at 0)

to 0 for recombination rate $R = 5 \cdot 10^{-5}$ and higher. For the lower recombination rates, the average sequence variance is dropping, but has not reached 0, but is closer to ~ 0.25. Having a variance of 0.25 would be equivalent to having one bit that is uniformly random in the population B.2. As a result, it seems likely that most of the variance in the low recombination rates has been driven out. However, the fact that the ideal sequence distance average hasn't dropped in these lower recombination rates suggests that really there is competition amongst clones that selection no longer can act on. Thus, genetic drift is the main thing still acting on these populations.

Also, it is worth noting that the ideal sequence distance average and average sequence variance have begun to diverge for different recombination rates by ~ 30 time steps. Not only is this divergence ~ 20 time steps later for the ideal sequence distance average and ~ 10 time steps later for the average sequence variance than for mutation at the same selection strength Figure 6 on page 10, but both divergences are occurring around the same time for recombination.

Since the average sequence variance has yet to drop, this suggests that recombination is responsible for assembling a variety of sequences from the pieces it has in the beginning. As most of the sequences in the population are far from the ideal sequence, the sequences have many errors. The number of sequences at any distance goes as a combinatorial. For a distance of 30 (76), there are $3 \cdot 10^{25}$ possible sequences and will be larger for distances closer to 50. Even though not all of these sequences will be present, there will still be a large variety. Thus, when recombination acts on a couple of the high fitness sequences, its results are fairly random and can even result in worse sequences. With selection acting on this assembly process, bottlenecks are created through which the population passes. This allows recombination to act on a better set of sequences and possibly construct ones closer to the ideal sequence. After a few iterations of this process, recombination and selection have mixed and refined the population and there is no remaining variance in the population for recombination to act. Finally, genetic drift acts on any clones that remain until only one sequence is left. For high recombination rate, $R = 0.0005$ and higher, all population variance is removed by ~ 1000. This will eventually occur for lower recombination rates too.

In this case, variance starts at a maximum value. So, recombination cannot be seen to increase variance here, but it can be seen to slow the decrease in variance and it can be seen reduce variance by mixing the best pieces of the sequences available in the population. In all these cases, recombination happens so frequently that it results in a well-mixed or homogenous system. To better understand the cases where the population average remains static with high variance it is worth looking at higher selection without mutation briefly.

Below are the graphs with $S = 0.05$ for the ideal sequence distance average and the average sequence variance Figure 11 on page 16. It can be noted that higher selection has the same sort of behavior on recombination as it did in mutation, which is higher selection forces the population through tighter bottlenecks. This process drives variation out of the population faster as has been seen for high selection and for recombination acting on a completely random population. The low recombination rates don't show noticeably different behavior in high selection than they did in low selection. The higher recombination rates of $R = 5 \cdot 10^{-2}$ and $R = 5 \cdot 10^{-3}$ are trapped farther from the ideal sequence in high selection than in low selection. The lower recombination rates differ somewhat, but is not clearly significant and not clearly different than pure effects of selection (76). However, to get a time scale for selection and better determine the order of magnitude of the average sequence variances, it is worth looking at a log-log plot.

From the log-log graph Figure 12 on page 17, it can be seen that in about 10 or less time steps the different recombination rates begin to diverge in both ideal sequence distance average and average sequence variance. This is a factor of 3 less than the low selection recombination. However, this divergence point is still later than the mutation case with the same selection Figure 8 on page 12. Due to the stronger selection, recombination rates of $R = 0.0005$ and higher are seen to drop in a few hundred time steps, which is almost an order of magnitude less than the lower selection case for recombination Figure 10 on page 15. However, for the recombination rate $R = 5 \cdot 10^{-5}$, the average sequence variance still goes to zero around ~ 1000 time steps, which is, interestingly enough, close to where the no recombination case loses its average sequence variance as well. As the non-recombining case has no way to add variance to the population, it likely has been to a handful of clones. If the population did not consist of clones, selection would have acted swiftly (given it is

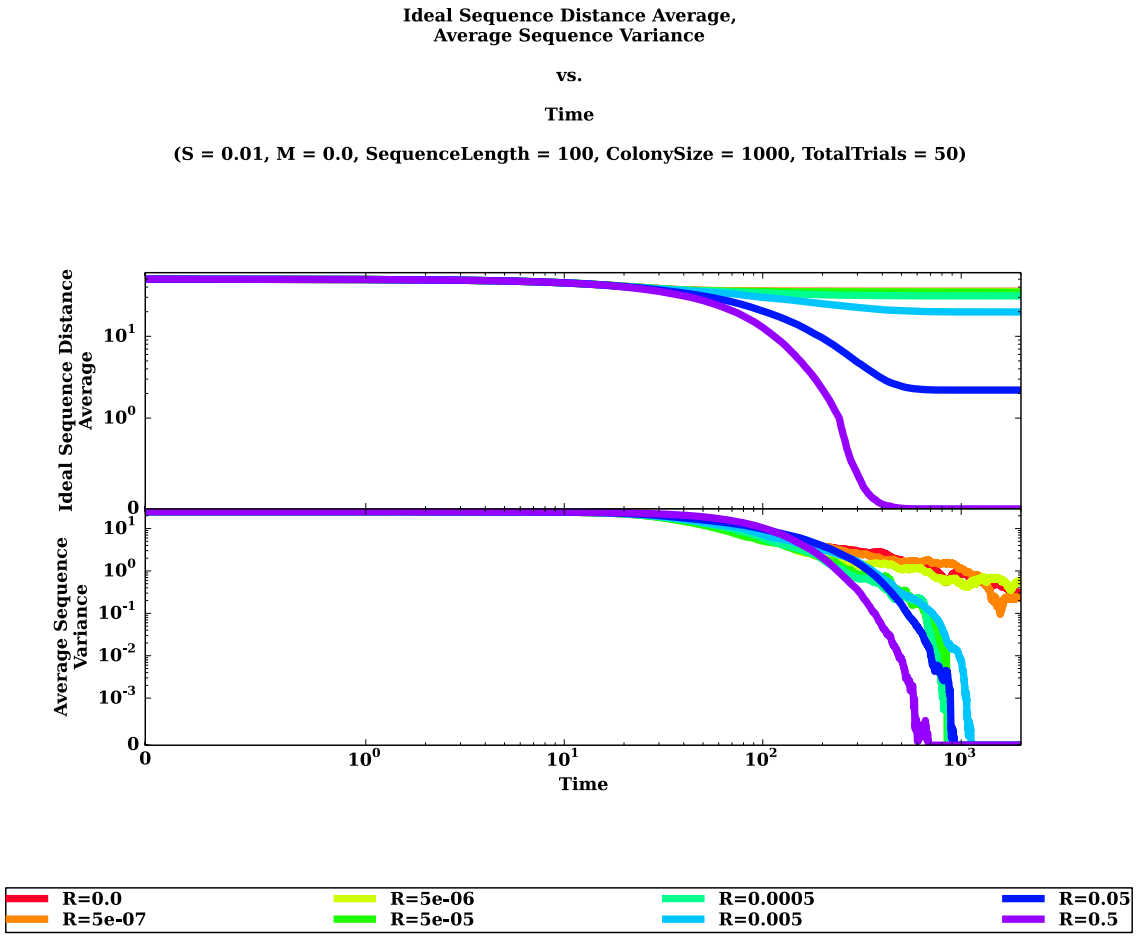

Figure 10: Log(Ideal Sequence Distance Average) and Log(Average Sequence Variance) vs. Log(Time) (Selection (S) held at $S = 0.01$, Recombination rate (R) ranging from $R = 5 \cdot 10^{-7}$ to $R = 5 \cdot 10^{-1}$ by orders of magnitude, includes 0 also, Mutation (M) held at 0, Log(ideal sequence distance average) linearized between 10^0 and 0 and Log(average sequence variance) linearized between 10^{-3} and 0))

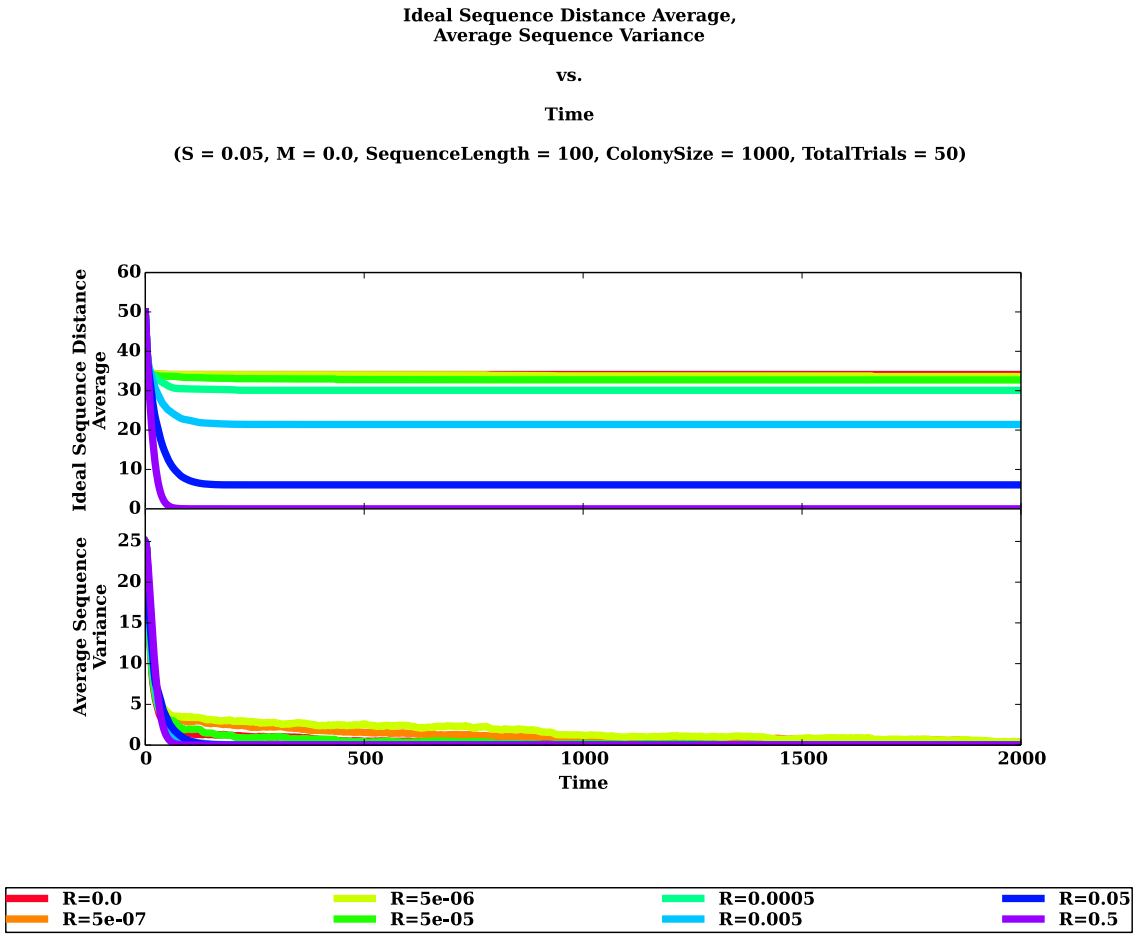

Figure 11: Ideal Sequence Distance Average and Average Sequence Variance vs. Time (Selection (S) held at $S = 0.05$, Recombination rate (R) ranging from $R = 5 \cdot 10^{-7}$ to $R = 5 \cdot 10^{-1}$ by orders of magnitude, includes 0 also, Mutation (M) held at 0)

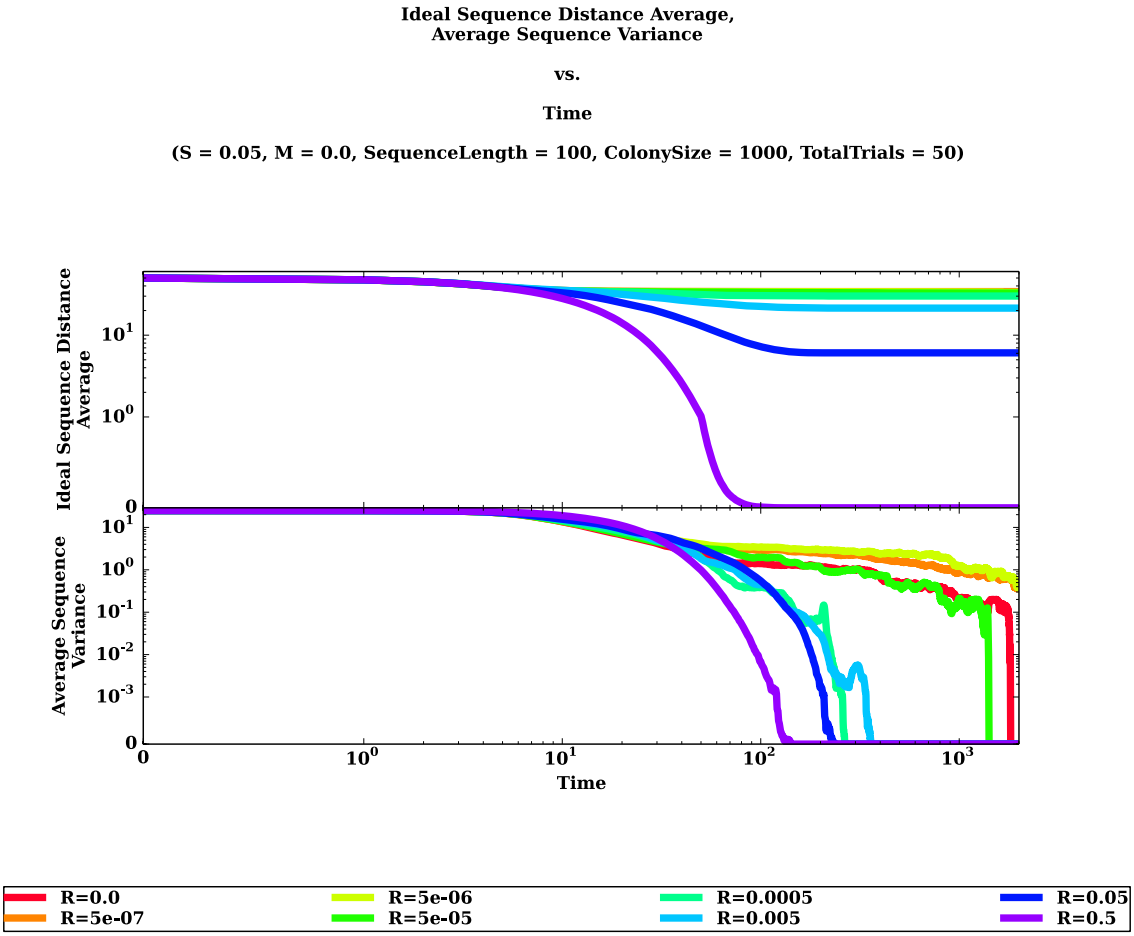

Figure 12: Log(Ideal Sequence Distance Average) and Log(Average Sequence Variance) vs. Log(Time) (Selection (S) held at $S = 0.05$, Recombination rate (R) ranging from $R = 5 \cdot 10^{-7}$ to $R = 5 \cdot 10^{-1}$ by orders of magnitude, includes 0 also, Mutation (M) held at 0, Log(ideal sequence distance average) linearized between 10^0 and 0 and Log(average sequence variance) linearized between 10^{-3} and 0))

so high) and removed everything but the best sequence. Thus, it is likely all the lower recombination rates also contain a fair number of clones, which would explain their similar path in average sequence variance reduction.

Taking the case of mutation and recombination, the ideal sequence distance average and average sequence variance can be seen Figure 13 on page 18. In all cases, the populations approach close to the ideal sequence. It can be seen that when the recombination rate is very low the populations behave much as if there was no recombination. The effects of recombination are more noticeable when the recombination rate is $R = 5 \cdot 10^{-4}$ and higher. It seems that recombination helps the population reach a lower ideal sequence distance average for all recombination rates. Also, all recombination rates appear to be getting closer to the ideal sequence. Though, it is not clear if they are all going at the same rate or not, which a log-log plot should help clarify.

Not too much can be garnered by looking at the linear plot of the average sequence variance. Clearly, it drops close to zero in all cases. However, it is worth noting that higher recombination rates appear to remain at a higher average sequence variance. After sometime, these higher recombination rates have average sequence variances that drop more rapidly to what appears to be lower average sequence variances. The time scales and magnitudes can be clarified by a log-log plot.

The log-log ideal sequence distance average and average sequence variance Figure 14 on

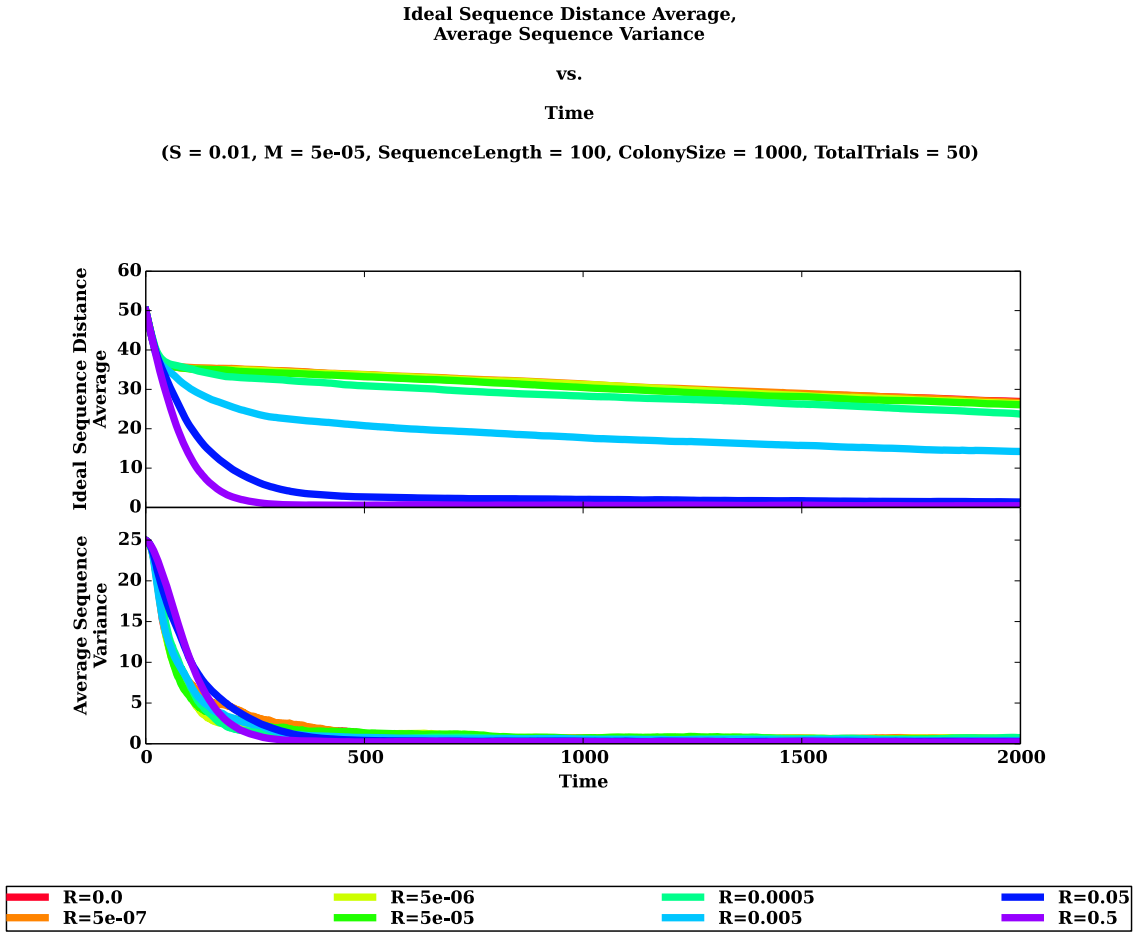

Figure 13: Ideal Sequence Distance Average and Average Sequence Variance vs. Time (Selection (S) held at $S = 0.01$, Recombination rate (R) ranging from $R = 5 \cdot 10^{-7}$ to $R = 5 \cdot 10^{-1}$ by orders of magnitude, includes 0 also, Mutation (M) held at $M = 5 \cdot 10^{-5}$)

page 20 is shown. The ideal sequence distance average seems to be improving more quickly at higher recombination rates even near the end. Though, it is a little difficult with $R = 0.5$ to determine if it is still improving due to the number of trials required to smooth the small values that its ideal sequence distance average has reached. However, it is also possible that mutation is keeping the recombination rate of $R = 0.5$ from improving further. This can be seen by comparing this graph to the no mutation case Figure 10 on page 15 seen before.

The time scale for divergence is about the same as for recombination itself. Also, divergence occurs in both the ideal sequence distance average and average sequence variance near the same time. These features were characteristic of divergences seen with recombination before. So, it seems recombination is a stronger effect than mutation in the beginning (when looking at this mutation rate).

Also, in all cases, the populations' average sequence variances are less than 1 and they seem to be stabilizing. There are differences in where the average sequence variances end for the different recombination rates, but these differences are not large. As a result, where the average sequence variance ends is dependent more on mutation and selection than recombination.

In short, recombination and selection without mutation ultimately removes variance from the population. This resulted in stagnation of the population before. However, when mutation is present, variation is continuously added to the population independent of its existing make-up. Thus, recombination can continue to build good sequences from the mutations added to the population. However, after some time, recombination ceases to be very effective and mutation dominates the behavior.

After seeing how the mutation and recombination interact with each other, especially in terms of the time scales of this interaction, it is important to look at periodic fluctuations in fitness. Just to summarize a few trends, mutation adds variance to the population. Recombination initially adds variance to population and then removes it. Selection is able to more quickly improve populations with larger variance because the tail of the distribution is farther out. However, there is only one ideal sequence; so, maintaining higher variance ultimately means that individuals remain farther from the ideal sequence. On the other hand, having lower variance makes selection less effective.

Thus, it is desirable to have high variance initially that decreases over time, which can be thought of as some sort of tempering. By having population variance start higher and decrease over time, selection acts on the population early on rapidly improving the population and bring it closer to the ideal sequence. Also, as the variance is removed from the population, more individuals in its population would be similar and thus closer to the ideal sequence. There are many possible parameters that would work to attain this behavior.

Ideally, all of these parameters would drive out as much population variance as possible so as to be very close to the ideal sequence as the environment is stationary. Once the environment changes, this may not be an ideal situation. The important question becomes how well the past environment predicts the future environment. Already, the case of an environment with perfect prediction of the future (a stationary environment) has been considered. If, on the other hand, periodic fluctuations are added to the environment, which completely randomize the ideal sequence, then the situation has changed dramatically.

Since the ideal sequence has no correlation to its state in the previous period (or any period for that matter), there is no benefit to being well adapted to the previous ideal sequence in the next period. In other words, all populations, regardless of their properties, will be on average just as far from the ideal sequence at the beginning of a new period. Thus, the only relevant properties involved in determining how well any population will do are its population variance and its parameters. However, it's clear that the parameters will dictate how the population variance will behave over time. Even though, in the first period, the population variance will start higher than it will in every other period (minus a few obvious exceptions), this will have no impact in any later period. As all periods start with a random ideal sequence, every period (after the first) will behave the same.

An example of this behavior can be seen in the following graph of Ideal Sequence Distance Average and Average Sequence Variance vs. Time with fixed mutation, recombination (non-existent in this case), selection, and period Figure 15 on page 23. Even though, recombination is neglected

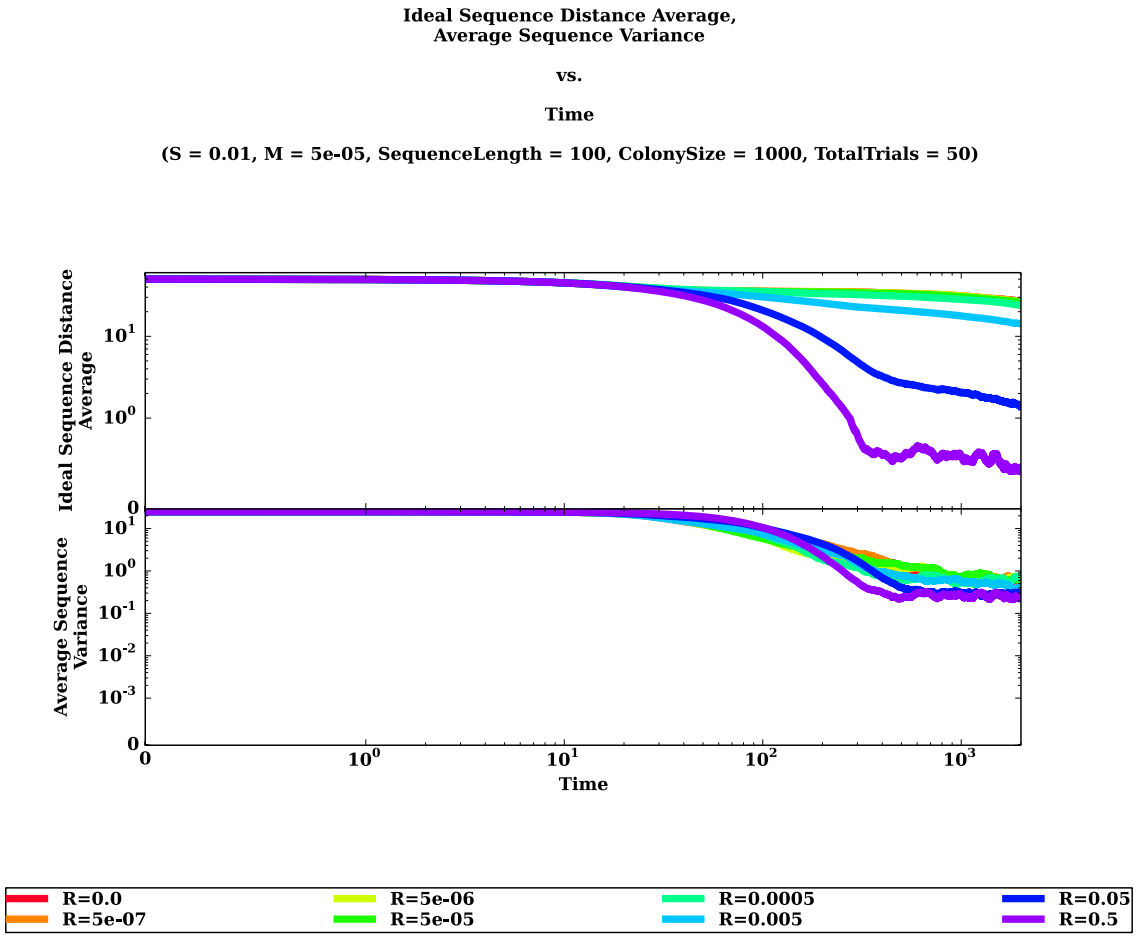

Figure 14: Log(Ideal Sequence Distance Average) and Log(Average Sequence Variance) vs. Log(Time) (Selection (S) held at $S = 0.01$, Recombination rate (R) ranging from $R = 5 \cdot 10^{-7}$ to $R = 5 \cdot 10^{-1}$ by orders of magnitude, includes 0 also, Mutation (M) held at $M = 5 \cdot 10^{-5}$, Log(ideal sequence distance average) linearized between 10^0 and 0 and Log(average sequence variance) linearized between 10^{-3} and 0))

in this particular case, adding it does not drastically change the patterns seen. Similarly, changing the other parameters results in roughly the same behavior with some changes in certain features.

In this graph Figure 15 on page 23, it can be seen that both the highest mutation ($M = 0.5$) and no mutation get stuck around half way from the ideal sequence in every period. This makes sense for both cases. To clarify why, consider the high mutation case first, an individual is selected for the next generation and afterwards half of its bits are flipped completely randomly. Thus, these mutations could just as likely destroy matches with the ideal sequence as create them. Even if one individual were substantially improved, it is clear this will be a unique case, statistically. In fact, it is possible that in such a population an individual exists that is very close to the ideal sequence. Even when taking the case of having an individual that exactly matches the ideal sequence, it can easily be seen that it's offspring will be scattered roughly halfway from the ideal sequence. This scattering is seen in the population variance Figure 15 on page 23 by the simple fact that the population variance stays roughly what it started at indicating the population remains random despite the forces of selection.

In the case of no mutation, the population rapidly becomes homogenous (many individuals with the same sequence). This homogeneity is evident by the population variance dropping to 0 in 2 periods Figure 15 on page 23. The removal of individuality from the population also makes sense in terms of the fact that there is nothing acting on the population to create diversity (mutation or recombination) while selection is removing it. Every period a new uniformly random ideal sequence is drawn; thus, it follows that another arbitrary sequence (i.e. from the population) will on average match half the ideal sequence. As the population does not improve, it remains the same distance from the ideal sequence for the period. Certainly, in each instance, the ideal sequence and the population may be closer or farther; thus, there are fluctuations seen with how close the sequence is in the trial average Figure 15 on page 23.

Why does the population not more quickly eliminate all, but one sequence in the case of no mutation. The answer is provided by three simple observations. First, the average distance of the population from the ideal sequence remains constant throughout each period (neglecting the very first couple time steps of any period, keep this in mind). Second, the population variance drops off at the beginning of each period. Third, the population variance does not remain constant throughout each period.

If there is population variance, it means there must be differences in the individuals in the population and if it varies it means that the population is changing how diversified it is over time. However, if the average distance remains the same while the variance is fluctuating, it means the individuals must have the same average distance. This is possible due to a symmetry of the problem. The fitness of an individual is invariant to which differences it has with the ideal sequence and only depends how many differences it has. In short, the hamming distance between the two sequences is all that is relevant. Thus, there is some sort of operation that leaves a sequence's distance from the ideal sequence invariant. This operation is really a reordering of the differences and the similarities shared with the ideal sequence, which means it goes as a combinatorial of the number of mismatches. These sequences that have the same fitness, but not the same sequence are known as clones.

As a result, it is clear that clones are randomly being picked in the no mutation case, but none are favored until the end of the period. Within each period, genetic drift operates on the clones. However, at the end of the period, a new ideal sequence is picked, to which the old clones may not be equally close. Thus, there is something akin to spontaneous symmetry breaking that is occurring at the end of each period, which results in removing a set of old clones each time until all individuals are alike in the no mutation case. Clones also can and like do occur in the mutation cases. However, it is difficult to determine to what degree clones are present. Though, some idea can be ascertained about the presence of clones by comparing mutation cases to the non-mutating case. More details about clonal competition can be found in the appendixB.3.

Regardless, the rest of mutation rates fall along this spectrum between high mutation and no mutation. As the two extremes of mutation rates result in the same unfavorable distance from the ideal sequence, it would seem that a desirable mutation rate would be somewhere in the middle of this spectrum; where, the population reaches the lowest mean distance from the ideal sequence. Even under high selection or different periods, the high and no mutation cases will behave the same.

Thus, there should be some desirable mutation rate that is in-between the two extremes for a given period and selection.

In this case, the best mutation rate is $M = 5 \cdot 10^{-4}$ as can be seen in the graph (blue-green or spring-green) Figure 15 on page 23. As previously mentioned, the population variance starts high and decreases throughout the period, but still remains high enough to recover after the period change Figure 15 on page 23. It is very important to emphasize that the population variance for the best mutation rate ($M = 5 \cdot 10^{-4}$) decreases throughout the period and does not appear able to decrease much farther. This decrease in population variance corresponds to a decrease in ideal sequence distance that appears to be flattening out at the end of the period.

This flattening out of the ideal sequence distance indicates that there are not better individuals that can be selected. It is possible this is due to clones as was the case before with the no mutation case; thus, there is fluctuation in the population variance without a corresponding improvement in the ideal sequence distance. Also, it is possible that a large number of mutations being made are deleterious; thus, a population variance remains mostly stable and the ideal sequence distance remains largely fixed as better individuals aren't being introduced.

The next two best are $M = 5 \cdot 10^{-3}$ and $M = 5 \cdot 10^{-5}$. In both of these mutation rates ($M = 5 \cdot 10^{-3}$ and $M = 5 \cdot 10^{-5}$), similar properties are exhibited as those seen in the best mutation rate. Namely, they have population variances that start high and decrease in the period window. With the higher mutation rate of the two ($M = 5 \cdot 10^{-3}$), its ideal sequence distance flattens out around the same time its population variance flattens out, which occurs the more quickly than the lower mutation rates ($M = 5 \cdot 10^{-4}$ and $M = 5 \cdot 10^{-5}$). The correspondence of the ideal sequence distance and population variance in their flattening out indicates that the mutations occurring are largely deleterious and/or there are clones competing in the population. This higher mutation rate ($M = 5 \cdot 10^{-3}$) would likely do better in a shorter period ($P < 1000$).

In the case of the lower mutation rate ($M = 5 \cdot 10^{-5}$), it improves more slowly; however, it seems likely to improve further, which could be seen if the period were longer. This lower mutation rate ($M = 5 \cdot 10^{-5}$) would likely then be better than the mutation rate of ($M = 5 \cdot 10^{-4}$) in a longer period. Also, the lower mutation rate ($M = 5 \cdot 10^{-5}$) has smaller lower peaks in its population variance than the higher mutation rates ($M = 5 \cdot 10^{-3}$ and $M = 5 \cdot 10^{-4}$), which are more clearly visible in the semilog version of this same data Figure 16 on page 24.

The semilog graph of the same data Figure 16 on page 24 more clearly shows the order of magnitude improvement in ideal sequence distance obtained by the populations with the best mutation rate ($M = 5 \cdot 10^{-4}$). Though, more importantly, the semilog graph shows that the good mutation rates of ($M = 5 \cdot 10^{-3}$, $M = 5 \cdot 10^{-4}$, and $M = 5 \cdot 10^{-5}$) all have average sequence variances that rise near the beginning of each period and fall afterwards. In fact, this behavior is clearer in the lower of the three mutations ($M = 5 \cdot 10^{-5}$) in the semilog graph than it was in the linear-linear graph. All other mutation rates do not seem to have this behavior. Though, $M = 5 \cdot 10^{-6}$ (yellow) has slight indications of this behavior in its average sequence variance, but it is not very perceptible. In all cases, increasing selection can cause the decrease in population variance to be more sudden.

As a result, it is clear that there are two important behaviors that are likely linked to what makes a set of parameters desirable for approaching the ideal sequence closest within a period. First, the ideal parameters result in a monotonic decrease in the ideal sequence distance. Second, the variance rises near to the beginning of each period and then falls throughout the remainder of the period to some non-zero value. Thus, it is important to determine the minimum ideal sequence distance value reached (last in the period) by each set of parameters. Also, it is important to determine the average sequence variance range (simply the difference between the maximum and minimum variance in the period) for each set of parameters. Noting how these quantities respond to different sets of parameters will help make sense of what parameters are optimal.

In order to calculate the minimum ideal sequence distance and average sequence variance range, each will be found for every period and then averaged over. However, the first period must be dropped from this averaging as the variance is abnormally high and equal in all cases, which results

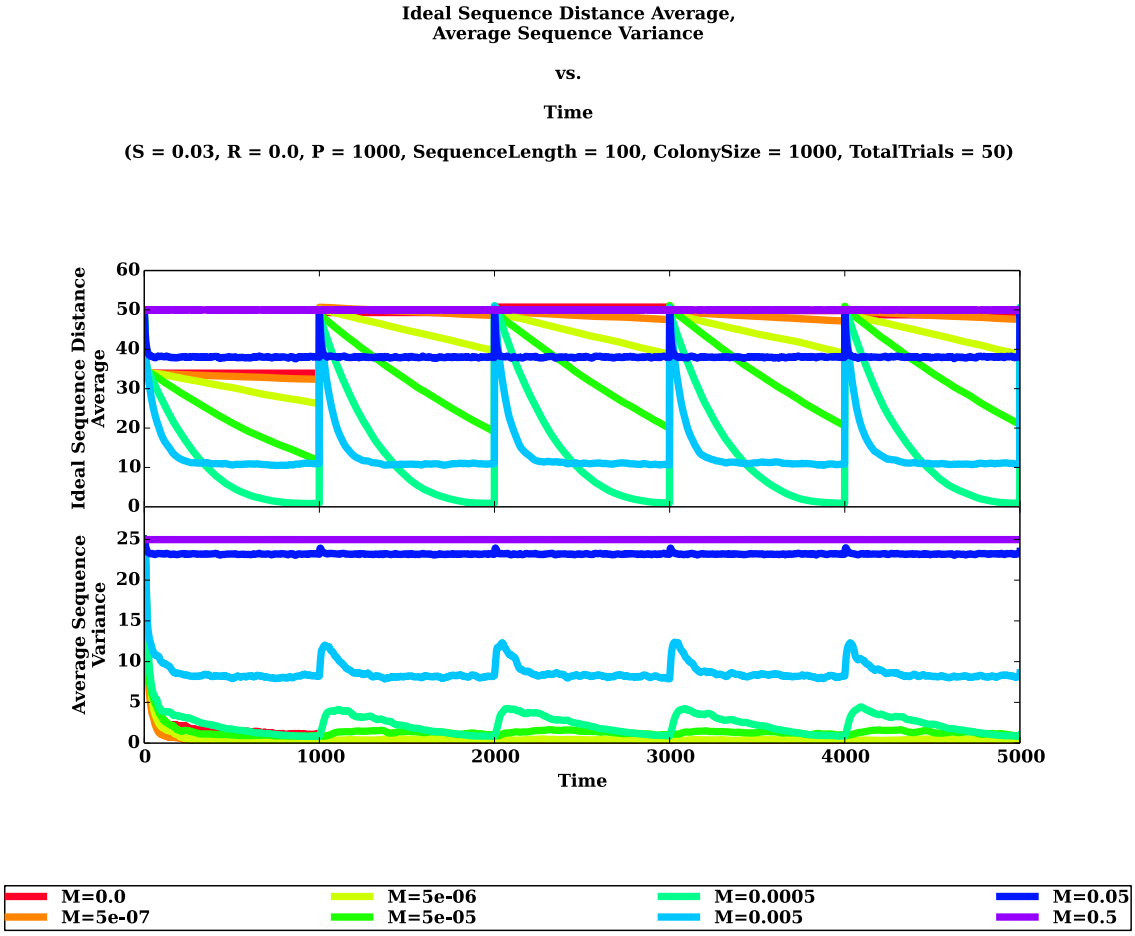

Figure 15: Ideal Sequence Distance Average and Average Sequence Variance vs. Time (Selection (S) held at $S = 0.03$, Recombination rate (R) held at $R = 0$, Mutation rate (M) ranging from $M = 5 \cdot 10^{-7}$ to $M = 5 \cdot 10^{-1}$ by orders of magnitude, includes 0 also)

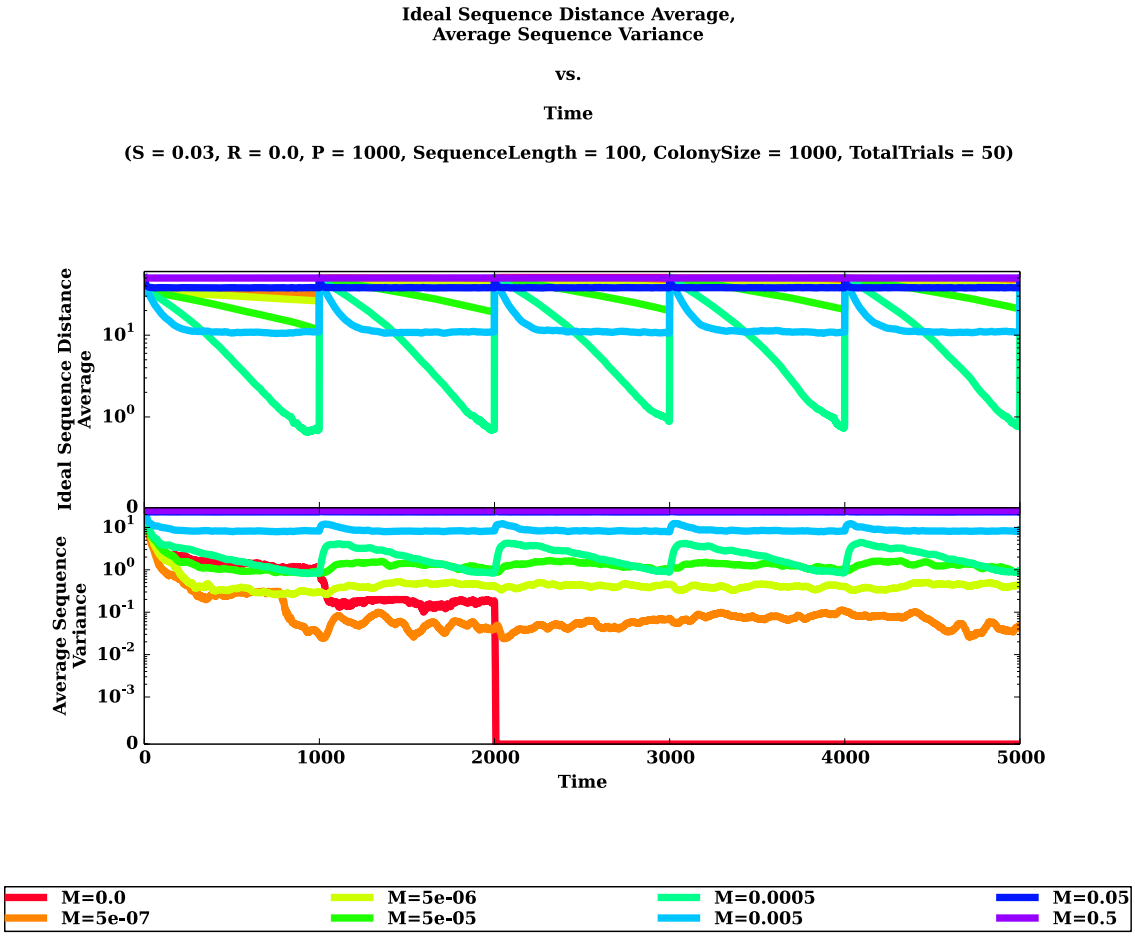

Figure 16: Log(Ideal Sequence Distance Average) and Log(Average Sequence Variance) vs. Time (Selection (S) held at $S = 0.03$, Recombination rate (R) held at $R = 0$, Mutation (M) ranging from $M = 5 \cdot 10^{-7}$ to $M = 5 \cdot 10^{-1}$ by orders of magnitude, includes 0 also, Period (P) held at $P = 1000$, Log(ideal sequence distance average) linearized between 10^0 and 0 and Log(average sequence variance) linearized between 10^{-3} and 0))

in behavior that is unusual compared to other periods (namely some ideal sequence distances drop lower than in other periods, the range of variance would be huge, different parameters have unequal initial variances later on, etc). Clearly, not all averages will contain the same number of points due to the fact that the number of time steps will be held constant and the period varied. It is clear though that this effect has vanished by the second period. So, all data used will be taken from the second period on.

The error is very small in both individual ideal sequence distance and average sequence variance. In the worst case, the error is less that 1 bit per individual, which is less than 1% of the sequence. Though, it may be worth noting that error goes up with lower mutation rate. This is likely attributable to the fact that incredibly low mutation rates may result in something on the order of 1 mutation per generation, which means that they are improving more slowly and as result they have not reached equilibrium (as is clear in the previous example Figure 15 on page 23). Also, the difference of one mutation becomes noticeable in averaging when they occur so infrequently. Regardless of the cause, the final point reached may vary more with lower mutation rates.

Included is a semilog plot showing the minimum ideal sequence distance of the population as well as the average sequence range (calculated as was specified before) versus the logarithm of the mutation rate Figure 17 on page 26. In this particular case, recombination has been excluded.

The graph has been coded such that lines that have the same color have the same period and lines with the same symbol have the same selection strength. It is clear as before that there are preferred parameters given selection and period (with colony size and sequence length also held constant).

Focusing on one period (i.e. $P = 200$) it is clear that the optimal mutation does not vary for different values of selection. Instead, increasing selection merely improves how effective that optimal mutation is compared to the rest. This appears to be true of selection for other periods as well. In the case of different periods, it appears that lower ideal sequence distance minimums occur for shorter periods at higher mutation rates. Similarly, for longer periods, lower mutation rates are preferred to minimize the ideal sequence distance.

Looking at the case of $P = 1000$, the ideal sequence distance curves of two highest selections ($S = 0.05$ and $S = 0.03$), though separated, are not separated by much. So, it would seem as the period becomes longer ($P \geq 1000$), the importance of increased selection for improving the population is less relevant. This is further supported by the fact that mutation rates near the optimum seem to be getting lower as well. Thus, high selection in this regime may actually relax constraints on the population.

Looking at the average sequence variance range (calculated as described before), the shorter period ($P = 200$) has a peak at the same point the ideal sequence distance is minimized. Though the peaks the average sequence range are close to the same values of mutation where the ideal sequence distance is minimized, the peaks do not match as well for larger periods. For the middle range period ($P = 500$), the ideal sequence distance is not clearly minimized in the values simulated. Thus, there may be a minimum between $M = 5 \cdot 10^{-4}$ and $M = 5 \cdot 10^{-3}$ for the ideal sequence distance with $P = 500$. Similarly, there may be a slightly higher peak for the average sequence variance in this range. Regardless, the error in the ideal sequence distance values shown is at most 1 and the error in the average sequence range is at most 0.6. So, it is more likely the peaks in the average sequence range could be slightly off. What is clear is that the narrow regime between roughly 10^{-4} and 10^{-2} contains the optimal mutation rate in all cases tried, which includes a large range in periods and selection strengths.

Knowing the effects of selection and period with regards to mutation, it is important to see how recombination fits into the mix as well. The following graph shows a variety of mutation and recombination rates with fixed selection ($S = 0.01$) and period ($P = 200$) Figure 18 on page 27. It would be expected that some amount of recombination would help find an optimum just as the case was with mutation. Surprisingly, when looking at the results , this does not appear to be the case. Instead, the highest recombination is always best regardless of mutation (with selection and period fixed). Similarly, the average sequence variance range is maximized for this case as well. The only notable exception to this is the highest recombination rate ($R = 0.5$) where the ideal sequence distance minimum and average sequence variance are off, which may be due to an optimum mutation between the two mutations sampled ($M = 5 \cdot 10^{-4}$ and $M = 5 \cdot 10^{-3}$).

Regardless, recombination appears to be always favorable as each higher recombination rate results in lower ideal sequence distance averages independent of mutation rate. Note that the optimal mutation rate remains essentially the same regardless of the recombination rate. However, the highest recombination rate, $R = 0.5$, and possibly the slightly lower recombination rate, $R = 0.05$, do appear to shift the optimal mutation rate to a lower value. Lower recombination rates may subtly affect the mutation rate as well. This would seem to suggest that recombination is able to add some variance to the population as well. Certainly, the range of the variance is increasing for increasing recombination rates. That being said, the range of variance remains peaked around the same optimal mutation rate favored by no recombination as opposed to shifting towards the newly favored lower mutation rates. Though, it is worth noting that the optimal mutation rate seems to have very similar range variance in all cases.

Why would recombination always be favorable? It is important to note that both mutation and recombination are allowed to happen at the same time. Thus, even if recombination isn't creating something new, mutation still can. Also, as can be seen by the increase in range variance, recombination is able to add variability to a population by removing linkage disequilibrium. Though it has been argued that recombination can break apart good sequences that seems to be unfounded in this model. This is likely due to the fact that most bad alleles have been driven from the

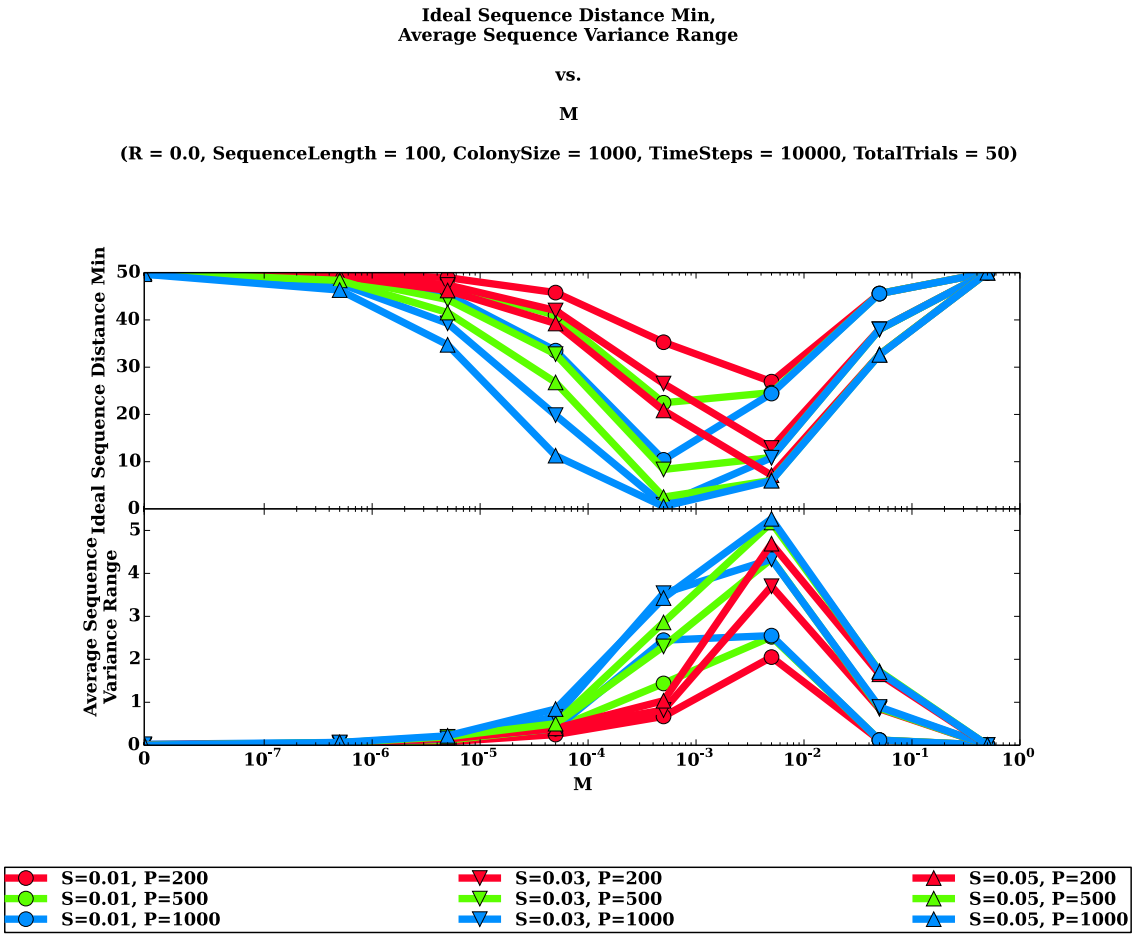

Figure 17: Ideal Sequence Distance Average Minimum and Average Sequence Variance Range vs. log(M) (Selection (S) ranges from 0.01 to 0.05 by steps of 0.02, Mutation rate (M) ranges from $5 \cdot 10^{-7}$ to $5 \cdot 10^{-1}$ by orders of magnitude (0 is included in semilog plot by linearizing between 0 and 10^{-7}), Recombination (R) is held at 0, and Period (P) has the value of 200, 500, or 1000 depending on the trial for a Random Fitness Function)

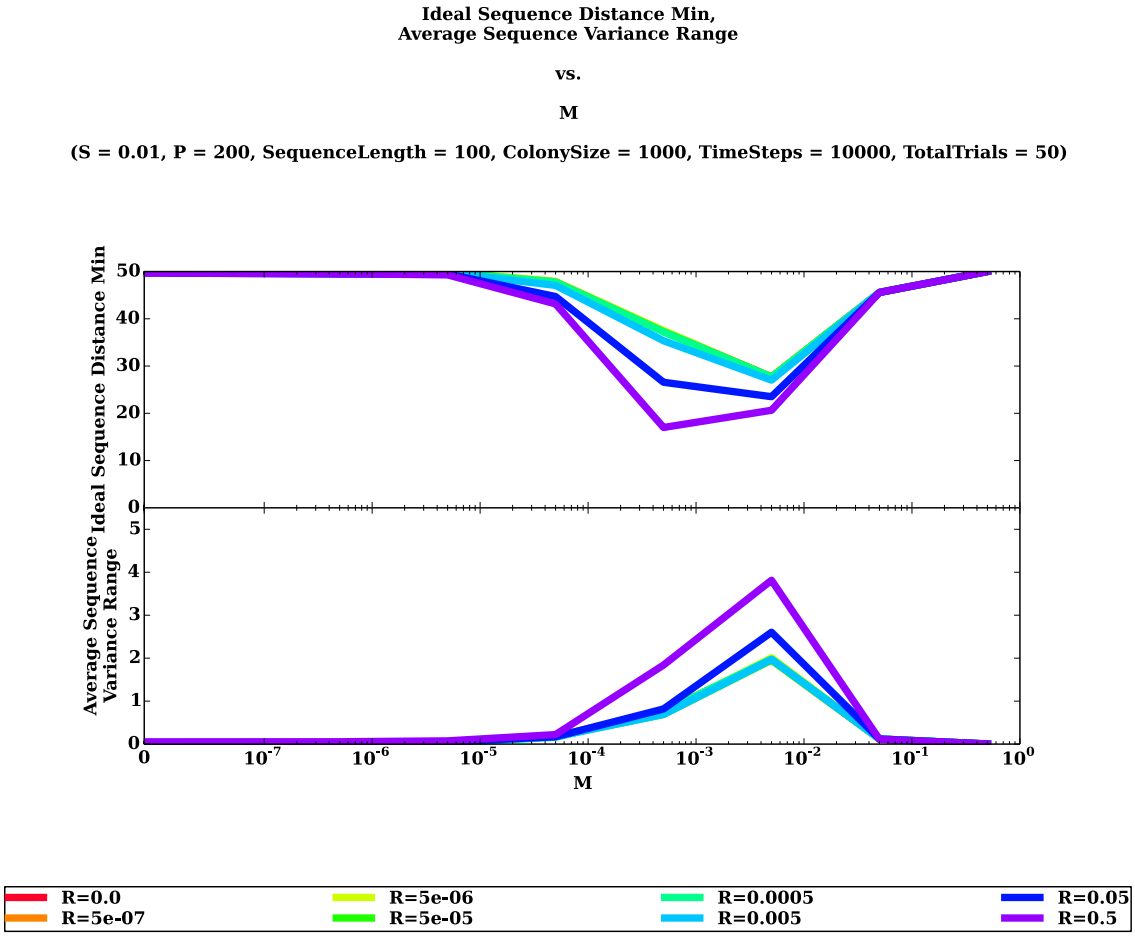

Figure 18: Ideal Sequence Distance Average Minimum and Average Sequence Variance Range vs. log(M) (Selection (S) held at 0.01, Recombination rate (R) ranges from $5 \cdot 10^{-7}$ to $5 \cdot 10^{-1}$ by factors of 10 (includes 0 as well), Mutation rate (M) ranges from $5 \cdot 10^{-7}$ to $5 \cdot 10^{-1}$ by steps of 10 (0 is included in semilog plot by linearizing between 0 and 10^{-7}), and Period (P) is held at 200 for a Random Fitness Function)

population by selection; thus, leaving a very homogenous population. As a result, recombination simply become useless as the population approaches the fitness peak. Perhaps, if the strength of selection were reduced, recombination might be able to do some more damage. Furthermore, this model has not considered more complex epistatic effects as all alleles have been treated as additive. If epistatic effects were considered, there would likely be some penalty for recombination in some cases. None of this, however, explains why recombination could lower the optimum mutation rate.

How could recombination lower the optimum mutation rate? It is possible that recombination is removing linkage equilibrium, which allows individual alleles to be selected. This would be the Hill-Robertson effect.[43] In other words, two alleles favorable alleles appear (non-errors in this model); however, they appear in two different individuals. As these two alleles try to fix, they would end up competing against each other. In this model, they could be clones, in which case drift would determine that one fixes and the other is lost from the population. If both favorable mutations appeared in the same individual, then they would fix, but this is unlikely. Recombination would remove this linkage disequilibrium by allowing the two alleles to appear together and separate in various combinations. Thus, selection would determine the proper combination of the mutations to keep. In short, recombination would make better use of the existing mutations and less mutations would be needlessly lost.

Another possibility is that recombination is breaking apart combinations of deleterious mutations. This would be related to Muller's ratchet.[68] In Muller's ratchet, a population centered around a peak on the fitness landscape is taken. Being on the peak, individual mutations would result in lowering fitness. However, given that mutations are spontaneous and occur with a certain rate, individuals in the population will begin to accumulate a few deleterious mutations. Even though the fitnesses of these individuals are lower, they will remain in the population due to the ease of adding a few deleterious mutations. Eventually, the most fit individual will be removed from the population. As the probability of back mutation is extremely low, the population will be stuck at this lower fitness. This process will repeat lowering the maximum fitness in the population over time. Recombination would resolve this problem by removing the linkage disequilibrium between deleterious mutations; thus, individuals with a few deleterious mutation would have some of them removed.

To better understand the behavior in this region and more closely examine these theories, a graph of the Ideal Sequence Distance Average and Average Sequence Variance vs. Time is shown for the unusual optimum mutation rate for the highest recombination rate $R = 0.5$ Figure 19 on page 29. Note that the two highest recombination rates ($R = 0.5$ and $R = 0.05$) demonstrate an unusual feature in their average sequence variance. Unlike the ramping behavior seen in pure mutation populations Figure 15 on page 23, the average sequence variance of recombination rises steadily from the beginning of the period; until, it reaches its maximum around the middle of the period; afterwards, the average sequence variance steadily falls until the end of the period (close to the same average sequence variance it began with).

This behavior is clearly a product of recombination and mutation together in this periodic environment. The initial quick rise is suggestive of the Hill-Robertson effect. Recombination acts on the population initial to create a variety of different pairings between existing alleles. This results in the initial rise in variance. Pairings of alleles initial results in many combinations (several that are favorable). Selection acts removing those that have large numbers of bad alleles. Eventually, due to the congregation of good alleles and selection's removal of unfavorable pairs, the population consists of members with a few good alleles to many good alleles. At this point, a few individuals have most of the good alleles available. The variance has peaked and selection begins removal of individuals with too few good alleles.

Recombination speeds the process of decreasing the variance by merging the few individuals with large numbers of good alleles together. As a result, recombination attempts to remove linkage between unfavorable alleles in the population. Thus, the population is now resisting the effects of Muller's ratchet. Note that pure mutational populations begin to slow down their progress to the ideal sequence due to deleterious mutations and clonal competitionFigure 15 on page 23. However, recombinatorial populations don't fall victim to clones as they allow selection to act on individual alleles with minimal correlation to other alleles. Thus, the populations with the highest recombination rates ($R = 0.5$ and $R = 0.05$) continue to improve even as those with the lowest recombination rates slow down.

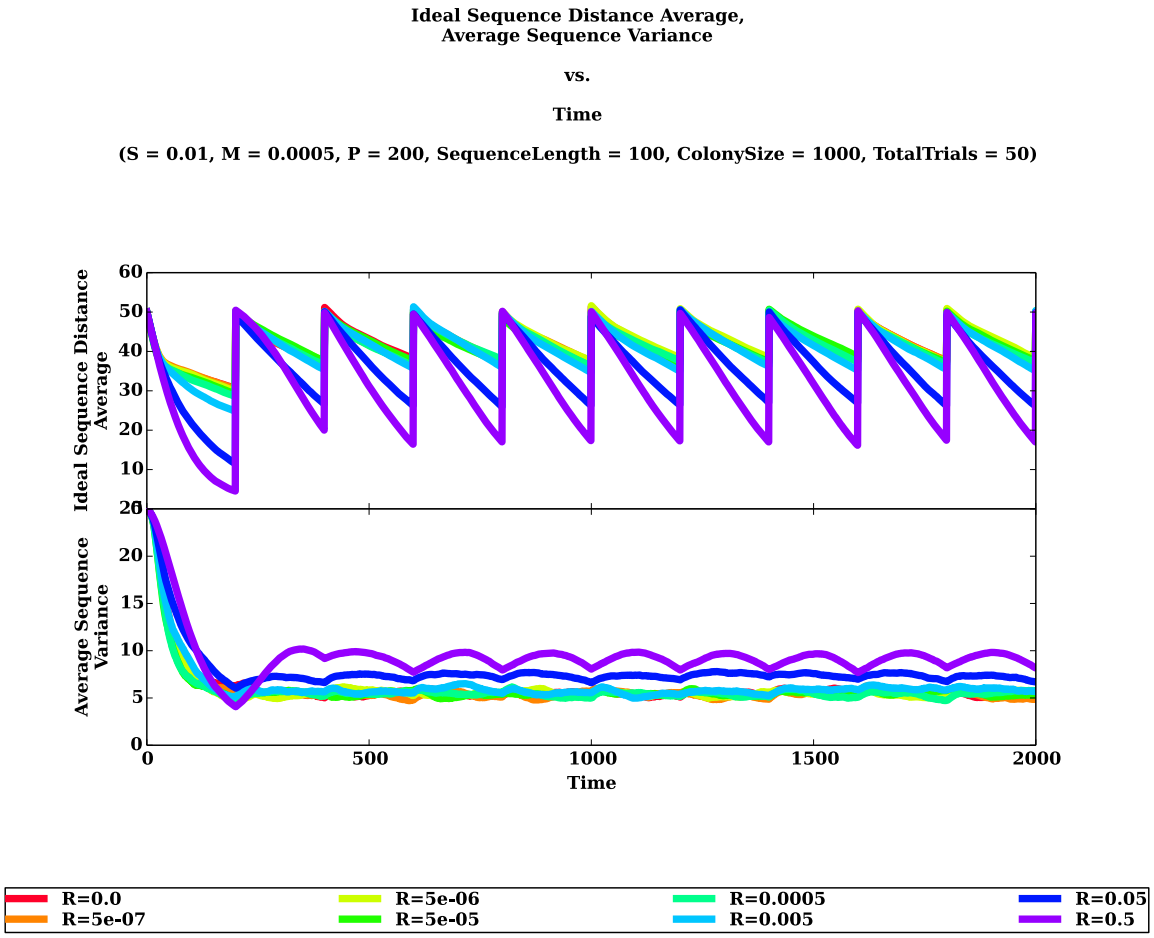

Figure 19: Log(Ideal Sequence Distance Average) and Log(Average Sequence Variance) vs. Time (Selection (S) held at $S = 0.01$, Recombination rate (R) ranging from $R = 5 \cdot 10^{-7}$ to $R = 5 \cdot 10^{-1}$ by orders of magnitude, includes 0 also, Mutation rate (M) held at $M = 5 \cdot 10^{-4}$, Period (P) held at $P = 200$, Log(ideal sequence distance average) linearized between 10^0 and 0 and Log(average sequence variance) linearized between 10^{-3} and 0)

4 Discussion

It is well understood that mutation adds variance to the population and selection removes it. The model reproduces these properties reliably. Further the model upholds the predictions of Fisher that the ability of a population to improve is dependent on the variation present in the population.[32] Though recombination, in the presence of selection and absence of mutation, is able to rapidly improve the population, it is unable to maintain variance in the population. Thus, recombination will optimize a population into a corner, leaving the population to fall victim to its own success. When mutation and recombination are considered together in the presence of selection, populations are found to be able to enjoy the benefits of rapid adaptation due to recombination while maintaining enough variance in the population to still adapt.

Using this combination of mutation and recombination in the presence of selection, evolution on a periodically randomized environment. As the environment was completely randomized, no benefit was gained by being well adapted to a previous environment. Instead, the ability to adapt to a new environment quickly was rewarded. The intent of these simulations was to determine the answer to the following question: What are the optimal mutation and recombination rates to most rapidly improve a population's fitness given a fixed period and selection strength? It was demonstrated that the recombination has no optimum (more recombination is always better). Furthermore, it was demonstrated that the mutation rate does have an optimum that is strongly affected by period and possibly weakly affected by selection.

Recombination does not appear to have an optimum in this model for several reasons. First, mutation continues in addition to recombination. As a result, if recombination does not do serious harm, then the population will improve at least as rapidly as it would with only mutation. Second, complex epistatic interactions were not considered. If two alleles had either positive or negative epistasis, breaking them apart would be much more costly. This could be implemented by making a more complex fitness landscape function. Third, much lower selection strengths were not tried. This may need to happen also when considering epistatic effects, but it certainly would be of interest without them just to see if mixing by recombination would slow improvement. Fourth, colony size and sequence length were not varied. The exploration of colony size and sequence length may have an impact on the value/affect of recombination on populations.

In an attempt to understand why higher recombination rates optimized lower mutation rates, it was determined that recombination initially adds variance to the population (by removing linkage disequilibrium), which selection then acts upon. Thus, recombination in this model resolves the Hill-Robertson effect. After creating a variety of favorably paired alleles, selection removes combinations with low fitness as recombination joins the few remaining collections of favorable alleles. Finally, recombination with the aid of selection removes the remaining deleterious alleles from the population. Thus, recombination in this model resolves problems similar to those addressed by Muller's ratchet. By resolving these problems, populations with high recombination are able to reach a higher fitness with a lower mutation. The add benefit of the lower mutation is that less deleterious mutation will be present in the population.

Nevertheless, the ability of recombination to act in this manner requires that mutation be present. It is not possibly for recombination to resolve the Hill-Robertson effect without having the variability in the population generated by mutation. Furthermore, it is not possible for the population to adapt in the next period if mutation is not present because recombination will aid selection in driving out variability in every loci. It is important to reiterate that recombination is only beneficial when acting on a background of variability generated by mutation.

There are several important areas of inquiry that follow from the results of this model (beyond those raised by recombination already). First, the form of recombination and how it is implemented should be further explored. Second, additional measures of the population should be considered. Third, a clearer understanding of how population variance results in an optimized ideal sequence distance. Fourth, a serious exploration of a variety of periodic environments should be considered.

In these experiments, the method of recombination's action has not been very seriously explored. The method of recombination has been fixed and it's mean has also. The mean of the current

distribution should be varied some. Different distributions should be tried to determine whether it affects the results regarding recombination no matter how unlikely. Recombination, in the current model, simply interleaves to sequences cutting sections out on the way. Creating recombination to act in a more diverse way would be desirable to see its other properties. In particular, allowing recombination to cause frameshifts. This would probably aid recombination to work at even higher mutations or still have some effect in low variance populations. Also, recombination has been carried out symmetrically in this model. In other words, both parents' sequences are combined in an even way on average. In nature, recombination does not seem to be roughly equal. Some parameter should be introduced and explored to tune the asymmetry between parents during recombination. Similarly, recombination can happen between many different bacteria, the exploration of recombination with multiple parents should be explored as well as their ordering. Finally, recombination, by its very nature, is mainly a spatial structured event. Recombination should be explored in spatial structure where only neighbors can recombine.

Even though the average ideal sequence distance and average sequence variance provide a great deal of information about a population and its evolution, the details of its internal structure are lost. Getting a measure of information entropy about a generation would be useful to determine whether it is dominated by a few extremely high fitness individuals or whether there are many individuals with elevated fitness. Determining quantities like the correlation between pairs and other sized groupings of bits in each generation would be desirable as such quantities could give insight into linkage disequilibrium, clonal interference, etc. Also, as easy as it is, keeping track of the minimum ideal sequence distance every generation. Though it would be substantially more difficult, determining some sort of connectivity measure between individuals of the same generation would be desirable as it could indicate how much clonal interference there is, how homogenous the population, how many new sequences are expected in the next generation, etc.

Though the measurement of range variance was useful in understanding the region of optimal mutation, the measure is incredibly simplistic (making its accuracy more surprising). A better measure should be determined to more accurately predict how changes in the population variance pre-dispose a population to an maximum fitness. For example, pure mutation cases (with a high enough mutation rate to have a noticeable effect) results in a delay followed by rapid build of population variance, which is then followed by a slow decline determined mainly by selection. The behavior of population variance in the pure mutation cases results in an ideal sequence distance that decreases rapidly at first, but begins slowing down due to the build of clones with a variety of deleterious mutations. In the cases with recombination, similar patterns in population variance can be observed. However, the phenomenon found in high recombination and lower mutation, demonstrates that population variances that do not undergo the rapid build-up avoid being slowed down by clones with deleterious mutation while trying to improve fitness. It would seem that some sort of Lagrangian for population variance could be written that would penalize sharp curves, reward large increases, etc.

Periodic fluctuations were considered here where the environment was completely randomized. Though this helps in understanding how population variance is changed as well as how it affects the population's fitness, it is not entirely realistic. More frequently, the environment maintains some consistency over time. Expansions on this model that have similar style periods, but that only randomizes some fraction of the peak, would be of interests. This would then allow adaptation to the previous peak to have value in reaching the next peak. It is possible that this sort of process would lower population variance even further. Additional expansions could consider whether the fraction of the environment to stay fixed should not also fluctuate in some manner either periodically or randomly. Certainly, the periods themselves could have randomized lengths as well. However, it seems doubtful that much more information can come from different period lengths than can already be found here. Though, in an environment that has some consistency over periods, it may be reasonable to consider periods of different length. Lastly, it would be worth considering an environment that varies slowly over time. In other words, the peak moves a little bit (if at all) each time step. This environment would be well informed by the insights from the previous period experiments, but still have plenty of additional facets to explore.

5 Conclusion

This thesis asked the following question: What are the optimal mutation and recombination rates to most rapidly improve a population's fitness given a fixed period and selection strength? Through the course of simulating a variety of different parameters the following has been determined. First, recombination has no optimum rate. In other words, more recombination is always better. Second, mutation has an optimum rate that is strongly affected by period and possibly weakly affected by selection. Third, high recombination is able to lower the optimum mutation rate by reducing linkage disequilibrium.

Nomenclature

Average Sequence Variance	Variance of sequences in a generation from the Average Sequence. Normally, a trial average is implied.
Average Sequence	The sequence is an averaging of all sequences in a generation. Each value in it represents the probability of having a 1 in that position.
Colony Size	Size of each generation. Also, shorthand is N. Held at 1000 for these simulations.
Ideal Sequence Distance Average	The average of all Ideal Sequence Distances of all sequences in one generation. This normally also implies averaging over trials as well.
Ideal Sequence Distance	Distance that an individual sequence is from the Ideal Sequence.
Ideal Sequence	The sequence that represents the peak of the Mt. Fuji Landscape.
Mutation	Mutation results in opposite state when it occurs. Normally, the probability of per site mutation is refered. Shorthand is μ in equations and M in graphs.
Recombination	Intertwining of sequences two individuals. Normally, the probability of recombination taking place between two individuals is refered to as R in graphs.
Selection	Determines how sharply peaked the probability distribution of individuals is. Also, shorthand is s in equations and S in graphs. Similar, to β in statistical mechanics.
Sequence Length	Length of all sequences. Also, shorthand is L. Held at 100 for these simulations.

Appendices

A Probability Distributions

A.1 Preliminaries

To denote a random number, X, drawn from a probability distribution, $P(X)$, the standard notation will be used 4.[3] Expectation values can be found if they are discrete 4 or continuous 6. The cumulative distribution function is the probability of having any value X drawn that is less than or equal to x (7).[3] It can be found by finding the expectation value of the step function (9); where, the step function is defined as such (8). Its complement can also be defined (10). The characteristic function will be defined as seen 11, which is essential a Fourier transform of the distribution.[3] From it, moments can be found if they are defined 12. The cumulant generating function will be defined as seen 13, which is not the same as the one defined from the moment generating function used by other authors.[3] From it, cumulants can be found if they are defined 14.[3] The zeroth moments is unity(15). Thus, the zeroth cumulant is null (16). The first cumulant and moment are the mean (17).[3] The second cumulant is the variance 18.[3]

$$X \sim P(X) \tag{4}$$

$$\langle f(X)\rangle_X = \sum_X f(X)\cdot P(X) \tag{5}$$

$$\langle f(X)\rangle_X = \int dx\, f(X)\cdot P(X) \tag{6}$$

$$C_X(x) \equiv P(X \leq x) \tag{7}$$

$$\Theta(x) = \begin{cases} 1 & , x \geq 0 \\ 0 & , x < 0 \end{cases} \tag{8}$$

$$C_X(x) = \langle \Theta(x - X)\rangle_X \quad , \forall X \sim P(X) \tag{9}$$

$$\overline{C}_X(x) \equiv 1 - C_X(x) \tag{10}$$

$$\phi_X(t) \equiv \left\langle e^{\imath\cdot t\cdot X}\right\rangle_X \quad , \forall X \sim P(X) \tag{11}$$

$$\mu_N = \imath^{-N}\cdot \tfrac{d^N}{dt^N}\left[\phi_X(t)\right]\Big|_{t\to 0} \tag{12}$$

$$g_X(t) \equiv \ln\left(\left\langle e^{\imath\cdot t\cdot X}\right\rangle_X\right) \quad , \forall X \sim P(X) \tag{13}$$

$$\kappa_N = \imath^{-N}\cdot \tfrac{d^N}{dt^N}\left[g_X(t)\right]\Big|_{t\to 0} \tag{14}$$

$$\mu_0 = 1 \tag{15}$$

$$\kappa_0 = 0 \tag{16}$$

$$\kappa_1 = \mu_1 = \mu_X \tag{17}$$

$$\kappa_2 = \mu_2 - (\mu_1)^2 = \sigma_X^2 \tag{18}$$

A.2 Bernoulli Distribution

The most basic distribution involved in the simulation, the Bernoulli distribution (19), is used to determine the probability of an event occurring successful or not; where, success happens with probability p [3]. This is essential a coin flip that is potentially biased. The cumulative distribution function is simple 20. The Bernoulli distribution's characteristic function can be found 22 and from it the characteristic function can be found 23. The mean 24 and variance 25 can thus be found.

$$X \sim P(X) = \begin{cases} p & , X = 1 \\ 1-p & , X = 0 \end{cases} , X \in \{0,1\} \tag{19}$$

$$C_X(x) = \begin{cases} 1 & , x \geq 1 \\ 1-p & , 1 > x \geq 0 \\ 0 & , 0 > x \end{cases} , x \in \{0,1\} \tag{20}$$

$$\begin{aligned} \phi_X(t) &= \textstyle\sum_X e^{\imath \cdot X \cdot t} \cdot P_X \\ &= p \cdot e^{\imath \cdot t} + (1-p) \end{aligned} \tag{21}$$

$$\phi_X(t) = p \cdot e^{\imath \cdot t} - p + 1 \tag{22}$$

$$g_X(t) = \ln\left(p \cdot e^{\imath \cdot t} - p + 1\right) \tag{23}$$

$$\mu_X = p \tag{24}$$

$$\sigma_X^2 = p \cdot (1-p) \tag{25}$$

A.3 Symmetric Bernoulli Distribution

The uniform or symmetric Bernoulli distribution is a special case of the Bernoulli distribution A.2; where, $p = 1 - p$ or in other terms $p = \frac{1}{2}$. Therefore, the distribution is (26). The cumulative distribution function is (27). The characteristic function is (28). The cumulant generating function is (29). The mean (30) and variance (31) are also found.

$$X \sim P(X) = \begin{cases} \frac{1}{2} & , X = 1 \\ \frac{1}{2} & , X = 0 \end{cases} , X \in \{0,1\} \tag{26}$$

$$C_X(x) = \begin{cases} 1 & , x \geq 1 \\ \frac{1}{2} & , 1 > x \geq 0 \\ 0 & , 0 > x \end{cases} , x \in \{0,1\} \tag{27}$$

$$\phi_X(t) = \left(\tfrac{1}{2}\right) \cdot e^{\imath \cdot t} + \tfrac{1}{2} \tag{28}$$

$$g_X(t) = \ln\left(e^{\imath \cdot t} + 1\right) - \ln(2) \tag{29}$$

$$\mu_X = \tfrac{1}{2} \tag{30}$$

$$\sigma_X^2 = \tfrac{1}{4} \tag{31}$$

A.4 Geometric Distribution

The geometric distribution is based off the Bernoulli distribution A.2. The question it tries to answer is when repeating draws from a Bernoulli distribution, how many times does the event fail to occur before it occurs successfully. As each draw comes from an i.i.d., the geometric distribution can be formulated as such (32)[3]; where, n is the number of failures before the first success. This distribution is already normalized, which can be proved by using the result for a geometric series (33). Note that the cases where $p = 0$ or $p = 1$ leave this distribution ill-defined as in both cases the event is always successful or not, which would result in the collapse of the distribution. The geometric distribution can also be thought of the discrete analog of the exponential distribution. The cumulative distribution function can be found using the same techniques involved in normalization 34. The characteristic function can be found 35 36 and from it the cumulant generating function 37. From the cumulant generating function, the mean 38 and variance can be found 39. Interestingly enough, the distribution can be rewritten in terms of its mean 40, which is useful in the implementation of certain algorithms.

$$N \sim P(N) = (1-p)^N \cdot p \quad , N \in \{0, 1, 2, \ldots\} \tag{32}$$

$$\begin{aligned} \textstyle\sum_N P(N) &= \textstyle\sum_N (1-p)^N \cdot p \\ &= p \cdot \left(\textstyle\sum_N (1-p)^N \right) \\ &= p \cdot \left(\frac{1}{1-(1-p)} \right) \\ &= p \cdot \left(\frac{1}{p} \right) \\ &= 1 \end{aligned} \tag{33}$$

$$C_N(n) = 1 - (1-p)^{n+1} \tag{34}$$

$$\begin{aligned} \phi_N(t) &= \textstyle\sum_N e^{\imath \cdot N \cdot t} \cdot P(N) \\ &= \textstyle\sum_N e^{\imath \cdot N \cdot t} \cdot (1-p)^N \cdot p \\ &= p \cdot \left(\textstyle\sum_N (e^{\imath \cdot t} \cdot (1-p))^N \right) \\ &= p \cdot \left(\frac{1}{1 - e^{\imath \cdot t} \cdot (1-p)} \right) \\ &= \frac{p}{1 - e^{\imath \cdot t} \cdot (1-p)} \end{aligned} \tag{35}$$

$$\phi_N(t) = \frac{p}{1 - e^{\imath \cdot t} \cdot (1-p)} \tag{36}$$

$$g_N(t) = \ln(p) - \ln(1 - e^{\imath \cdot t} \cdot (1-p)) \tag{37}$$

$$\mu_N = \frac{1-p}{p} \tag{38}$$

$$\sigma_N^2 = \frac{1-p}{p^2} \tag{39}$$

$$P(N) = \left(\frac{\mu_N}{\mu_N + 1} \right)^N \cdot \left(\frac{1}{\mu_N + 1} \right) \tag{40}$$

A.5 Binomial Distribution

The binomial distribution is a distribution that describes the distribution of the sum of N i.i.d. Bernoulli distributions A.2 with sum k (41). All of the binomial distribution's properties are simple extensions of the Bernoulli distributions' properties. The cumulative distribution function is not trivial for the binomial distribution and won't be needed; so, it will be skipped. The distribution's characteristic function can be found (42). From the characteristic function, the distribution's cumulant generating function can be found as well(43). Similarly, the distribution's mean (44) and variance (45) can be found.

$$P(k) = \begin{pmatrix} N \\ k \end{pmatrix} \cdot p^k \cdot (1-p)^{N-k} \quad , \; k \in \{0, 1, 2, \ldots, N\} \tag{41}$$

$$\phi_X(t) = (p \cdot e^{\imath \cdot t} - p + 1)^N \tag{42}$$

$$g_X(t) = N \cdot \ln(p \cdot e^{\imath \cdot t} - p + 1) \tag{43}$$

$$\mu_X = N \cdot p \tag{44}$$

$$\sigma_X^2 = N \cdot p \cdot (1-p) \tag{45}$$

A.6 Symmetric Binomial Distribution

The symmetric binomial distribution is a special case of the binomial distribution A.5 in that it is a sum of N i.i.d. symmetric Bernoulli distributions A.3. The distribution can be rewritten (46). Similarly, its characteristic (47) and cumulant generating functions (48) can be found. Also, its mean (49) and variance (50) can be found. When N becomes large, the binomial distribution can be approximated as a normal distribution.

$$P(k) = \left(\tfrac{1}{2}\right)^N \cdot \begin{pmatrix} N \\ k \end{pmatrix} \tag{46}$$

$$\phi_X(t) = \left(\tfrac{1}{2}\right)^N \cdot (e^{\imath \cdot t} + 1)^N \tag{47}$$

$$g_X(t) = N \cdot \ln(e^{\imath \cdot t} + 1) - N \cdot \ln(2) \tag{48}$$

$$\mu_X = \tfrac{N}{2} \tag{49}$$

$$\sigma_X^2 = \tfrac{N}{4} \tag{50}$$

A.7 Normal Distribution

The normal distribution is the maximum entropy probability distribution for a given mean, μ, and variance, σ^2.[3] It can also be thought of as the continuous form of the symmetric binomial distributionA.6. Also, the normal distribution has only trivial cumulants after the variance. The probability distribution can be represented by the following equation (51). The cumulative distribution function for the normal distribution is given by the following equation(52) with the given error function (53).[1] The characteristic function is found to be similar to the normal distribution(54). Its cumulant generating function is (55). Finally, its mean (56) and variance (57) simplify as well.

$$P(X) = \left(\frac{1}{\sqrt{2\cdot\pi\cdot\sigma^2}}\right)\cdot\exp\left(\left(\frac{1}{2}\right)\cdot\left(\frac{X-\mu}{\sigma}\right)^2\right) \quad , X\in\mathbb{R} \tag{51}$$

$$C_X(t) = \left(\frac{1}{2}\right)\cdot\left(1+\operatorname{erf}\left(\frac{t-\mu}{\sqrt{2\cdot\sigma^2}}\right)\right) \tag{52}$$

$$\operatorname{erf}(x) = \left(\frac{2}{\sqrt{\pi}}\right)\cdot\left(\int_0^x \exp\left(-t^2\right)\,dt\right) \tag{53}$$

$$\phi_X(t) = \exp\left(\left(\frac{1}{2}\right)\cdot\left(\frac{\mu}{\sigma}\right)^2\right)\cdot\exp\left(-\left(\frac{\sigma^2}{2}\right)\cdot\left(t-\imath\cdot\left(\frac{\mu}{\sigma^2}\right)\right)^2\right) \tag{54}$$

$$g_X(t) = \left(\frac{1}{2}\right)\cdot\left(\frac{\mu}{\sigma}\right)^2-\left(\frac{\sigma^2}{2}\right)\cdot\left(t-\imath\cdot\left(\frac{\mu}{\sigma^2}\right)\right)^2 \tag{55}$$

$$\mu_X = \mu \tag{56}$$

$$\sigma_X^2 = \sigma^2 \tag{57}$$

A.8 Uniform Distribution

The uniform distribution describes a continuous range of numbers that all have equal probability of being picked (58). The cumulative distribution function is found to be fairly straightforward 59. The distribution's characteristic function can be calculated with a few tricks (60) and is (61). The cumulant generating function is simply found (62). Taking derivative and using limits, the mean (63) and variance (64) can be found.

$$P(X) = \frac{1}{b-a} \quad , X\in[a,b) \tag{58}$$

$$C_X(x) = \frac{x-a}{b-a} \quad , X\in[a,b) \tag{59}$$

$$\begin{aligned}
\phi_X(t) &= \int e^{\imath \cdot t \cdot x} \cdot P(x)\, dx \\
&= \left(\tfrac{1}{b-a}\right) \cdot \int_a^b e^{\imath \cdot t \cdot x}\, dx \\
&= \left(\tfrac{1}{(b-a)\cdot t}\right) \cdot \int_{a \cdot t}^{b \cdot t} e^{\imath \cdot y}\, dy \\
&= e^{\imath \cdot a \cdot t} \cdot \left(\tfrac{1}{(b-a)\cdot t}\right) \cdot \int_0^{(b-a)\cdot t} e^{\imath \cdot y}\, dy \\
&= e^{\imath \cdot (b+a) \cdot t/2} \cdot \left(\tfrac{1}{(b-a)\cdot t}\right) \cdot \int_{-(b-a)\cdot t/2}^{(b-a)\cdot t/2} e^{\imath \cdot y}\, dy \\
&= e^{\imath \cdot (b+a) \cdot t/2} \cdot \left(\tfrac{1}{(b-a)\cdot t}\right) \cdot \left(\int_0^{(b-a)\cdot t/2} e^{\imath \cdot y}\, dy + \int_{-(b-a)\cdot t/2}^{0} e^{\imath \cdot y}\, dy\right) \\
&= e^{\imath \cdot (b+a) \cdot t/2} \cdot \left(\tfrac{1}{(b-a)\cdot t}\right) \cdot \left(\int_0^{(b-a)\cdot t/2} e^{\imath \cdot y}\, dy - \int_{(b-a)\cdot t/2}^{0} e^{-\imath \cdot y}\, dy\right) \\
&= e^{\imath \cdot (b+a) \cdot t/2} \cdot \left(\tfrac{1}{(b-a)\cdot t}\right) \cdot \left(\int_0^{(b-a)\cdot t/2} \left(e^{\imath \cdot y} + e^{-\imath \cdot y}\right)\, dy\right) \\
&= e^{\imath \cdot (b+a) \cdot t/2} \cdot \left(\tfrac{2}{(b-a)\cdot t}\right) \cdot \left(\int_0^{(b-a)\cdot t/2} \cos(y)\, dy\right) \\
&= e^{\imath \cdot (b+a) \cdot t/2} \cdot \left(\tfrac{2}{(b-a)\cdot t}\right) \cdot \sin\left(\tfrac{(b-a)\cdot t}{2}\right)
\end{aligned} \tag{60}$$

$$\phi_X(t) = \left(\tfrac{2}{(b-a)\cdot t}\right) \cdot e^{\imath \cdot (b+a) \cdot t/2} \cdot \sin\left(\tfrac{(b-a)\cdot t}{2}\right) \tag{61}$$

$$g_X(t) = \ln\left(\tfrac{2}{(b-a)\cdot t}\right) + \left(\tfrac{\imath \cdot (b+a) \cdot t}{2}\right) + \ln\left(\sin\left(\tfrac{(b-a)\cdot t}{2}\right)\right) \tag{62}$$

$$\mu_X = \tfrac{b-a}{2} \tag{63}$$

$$\sigma_X^2 = \tfrac{(b-a)^2}{12} \tag{64}$$

A.9 Standard Uniform Distribution

The standard uniform distribution is a special case of the uniform distribution A.8; where, $a = 0$ and $b = 1$. Thus, the probability distribution simplifies to (65). The cumulative distribution function for the standard uniform distribution is given (66). The characteristic function is (67). Its cumulant generating function is (68). Finally, its mean (69) and variance (70) simplify as well.

$$P(X) = 1 \quad , X \in [0, 1) \tag{65}$$

$$C_X(x) = x \quad , X \in [0, 1) \tag{66}$$

$$\phi_X(t) = \left(\tfrac{2}{t}\right) \cdot e^{\imath \cdot t/2} \cdot \sin\left(\tfrac{t}{2}\right) \tag{67}$$

$$g_X(t) = \ln\left(\tfrac{2}{t}\right) + \left(\tfrac{\imath \cdot t}{2}\right) + \ln\left(\sin\left(\tfrac{t}{2}\right)\right) \tag{68}$$

$$\mu_X = \tfrac{1}{2} \tag{69}$$

$$\sigma_X^2 = \tfrac{1}{12} \tag{70}$$

B Population Distribution Cases

B.1 Initial Distribution

B.1.1 Ideal Sequence Distance Average

In all of the simulations, the initial population has its sequences (of length L) drawn from L i.i.d. symmetric Bernoulli distributions (26). Regardless of what the ideal sequence is, a randomly picked individual will have a probability of k matches given by a symmetric binomial distribution (46) with the ideal sequence. Hence, in all cases the ideal sequence distance average is found to be (71). Given that $L = 100$ is held constant, the ideal sequence distance average is 50.

$$\mu_{Ideal\,Sequence\,Distance} = \frac{L}{2} \tag{71}$$

B.1.2 Average Sequence Variance

The average sequence is determined by averaging the sequences in the population (of size N); thus, each bit is also defined by a symmetric binomial distribution as it is a sum of N i.i.d. symmetric Bernoulli distribution A.5. Finding the variance for one bit can again be done (though it will require a little math). In the calculation that follows, l is the position in the sequence (all bits will follow the same distribution), i is the individual in the population (all are described by the same distribution so no difference), j is the value of the bit of an arbitrary individual's sequence and k is the number of individuals with a 1 in the same place) 72. The result is then found 73. Thus, simulations will start close to this average sequence variance of $\frac{L}{4}$. Given that $L = 100$ is held constant, this is 25.

$$\sigma^2 = \sum_{l=1}^{L} \sum_{i=1}^{N} \sum_{k=0}^{N} \sum_{j=0}^{1} \left(j - \tfrac{k}{N}\right)^2 \cdot \left(\tfrac{1}{2}\right)^{N+1} \cdot \binom{N}{k}$$

$$\sigma^2 = \sum_{l=1}^{L} \sum_{i=1}^{N} \sum_{k=0}^{N} \sum_{j=0}^{1} \left(j^2 - 2 \cdot j \cdot \left(\tfrac{k}{N}\right) + \left(\tfrac{k}{N}\right)^2\right) \cdot \left(\tfrac{1}{2}\right)^{N+1} \cdot \binom{N}{k}$$

$$\sigma^2 = \sum_{l=1}^{L} \sum_{i=1}^{N} \sum_{k=0}^{N} \left(1 - 2 \cdot \left(\tfrac{k}{N}\right) + 2 \cdot \left(\tfrac{k}{N}\right)^2\right) \cdot \left(\tfrac{1}{2}\right)^{N+1} \cdot \binom{N}{k}$$

$$\sigma^2 = \sum_{l=1}^{L} \sum_{i=1}^{N} \sum_{k=0}^{N} \left(\left(\tfrac{1}{2}\right) - \left(\tfrac{1}{N}\right) \cdot k + \left(\tfrac{1}{N}\right)^2 \cdot k^2\right) \cdot \left(\tfrac{1}{2}\right)^{N} \cdot \binom{N}{k}$$

$$\sigma^2 = \sum_{l=1}^{L} \sum_{i=1}^{N} \left(\tfrac{1}{2}\right) - \left(\tfrac{1}{N}\right) \cdot \left(\tfrac{N}{2}\right) + \left(\tfrac{1}{N}\right)^2 \cdot \left(\tfrac{N}{4}\right)$$

$$\sigma^2 = \sum_{l=1}^{L} \sum_{i=1}^{N} \tfrac{1}{4 \cdot N}$$

$$\sigma^2 = \sum_{l=1}^{L} \tfrac{1}{4}$$

$$\sigma^2 = \tfrac{L}{4} \tag{72}$$

$$\sigma^2_{Average\,Sequence\,Variance} = \tfrac{L}{4} \tag{73}$$

B.1.3 Approximate Best Individual

An interesting consequence of this initial distribution is apparent when approximating the closest individual to the ideal sequence at initialization. As all bits in all individuals' sequences are drawn from i.i.d. symmetric Bernoulli distributions A.3, the chance of any bit in an individual's sequence matching a bit in the same position in the ideal sequence will be described by a symmetric Bernoulli distribution. Thus, the number of matches between an individual's sequence and the ideal sequence, in other words the ideal sequence distance, will be described by a symmetric binomial distribution A.6. As the sequence length, L, is long ($L = 100$ in the cases looked at), the ideal sequence distance is roughly normally distributed A.7 and can be well approximated by it. Furthermore, the ideal sequence distance average for the initial distribution will also be normally distributed.

It seem reasonable that the individual that is best fit must occur once in the population, which will be taken to mean that the remaining probability in the tail will be around $\frac{1}{N}$; where, N is the number of individuals in each generation. Knowing the cumulative distribution function for

the normal distribution (52) (where μ is the mean and σ^2 is the variance) with the given the error function (53) [1] and its inverse 74, the mean, variance, and the tail probability can be substituted in, which results in the simplified expression (75). Solving this expression for x, the approximated distance of the closest sequence in the population, is found (76). When using the fixed simulation values of $L = 100$ and $N = 1000$, it is found that $x \approx 38.37$, which is a little higher than the simulation averages seen.

The fact that this result is higher is not surprising due to the fact that 50 trials are averaged over, which results in this being an overestimation. If N is made to be a product of the number of trials and the number of individuals in each generation ($N = 50,000$), so that one individual in all trials in the first generation will have a distance represented by x, it is found that $x \approx 32.30$. In this case, the value is lower than the simulation averages, which again is not surprising based on the fact that this distance likely represents one individual in one generation in one trial; thus, it will not greatly affect the average. Thus, it seems reasonable that the best individual at initialization will be in this range. Slightly better approximations could be made by using the cumulative distribution function for the binomial distribution instead; however, it is unlikely to result in a noticeably different approximation for large L and N.

$$\operatorname{erf}^{-1}(\operatorname{erf}(x)) = x \tag{74}$$

$$\tfrac{1}{N} = \left(\tfrac{1}{2}\right) \cdot \left(1 - \operatorname{erf}\left(\frac{1-\left(\frac{2}{L}\right)\cdot x}{\sqrt{\frac{2}{L}}}\right)\right) \tag{75}$$

$$x = \left(\tfrac{L}{2}\right) \cdot \left(1 - \sqrt{\tfrac{2}{L}} \cdot \operatorname{erf}^{-1}\left(1 - \tfrac{2}{N}\right)\right) = L \cdot \left(\tfrac{1}{2} - \sqrt{\tfrac{1}{2 \cdot L}} \cdot \operatorname{erf}^{-1}\left(1 - \tfrac{2}{N}\right)\right) \tag{76}$$

This can be further approximated by using an asymptotic expansion for the error function around 1 as given in the function (77).[102] Replacing x with $1 - \frac{2}{N}$, the asymptotic expansion can be simplified (78). As this simplification is intended for large N, it is clear that the ratio inside the square root is going toward zero. However, as the ratio is positive, this approximation will somewhat larger than the original. As a result, the asymptotic expansion can be further simplified to its final form (79). Thus, the distance of the closest sequence can be rewritten as shown (80). Using this approximation with the previously stated simulation values ($L = 100$ and $N = 1000$), it is found that $x \approx 31.42$, which is a bit lower than previous estimation, but still reasonably close.

This additional approximation helps clarify the scaling relation that controls the rapid selection in the offset of the simulation. The term, $\sqrt{\frac{\ln(N)}{2 \cdot L}}$, determines the probability of a bit having an error in the best individual's sequence. Note that if the population goes to unity or the sequence length goes to infinity the best individual is just a random individual. It is worth pointing out that that there is a critical population size at which the best individual should have the same sequence as the ideal sequence, which can be approximated (81). However, this poses little concern. Even for a sequence length previously given ($L = 100$), the population would need to be at least $N \approx 5.18 \times 10^{21}$, which is basically impractically large for the simulation.

$$\operatorname{erf}^{-1}(x) \approx \sqrt{\left(\tfrac{1}{2}\right) \cdot \left(\ln\left(\frac{2}{\pi \cdot (1-x)^2}\right) - \ln\left(\ln\left(\frac{2}{\pi \cdot (1-x)^2}\right)\right)\right)} \quad , x \to 1^- \tag{77}$$

$$\begin{array}{rcl} \text{erf}^{-1}\left(1-\frac{2}{N}\right) & \approx & \sqrt{\left(\frac{1}{2}\right)\cdot\left(\ln\left(\frac{N^2}{2\cdot\pi}\right)-\ln\left(\ln\left(\frac{N^2}{2\cdot\pi}\right)\right)\right)} \\ \text{erf}^{-1}\left(1-\frac{2}{N}\right) & \approx & \sqrt{\ln\left(N\right)-\left(\frac{1}{2}\right)\cdot\left(\ln\left(2\cdot\ln\left(N\right)-\ln\left(2\cdot\pi\right)\right)+\ln\left(2\cdot\pi\right)\right)} \\ \text{erf}^{-1}\left(1-\frac{2}{N}\right) & \approx & \sqrt{\ln\left(N\right)}\cdot\sqrt{1-\frac{\ln(2\cdot\ln(N)-\ln(2\cdot\pi))+\ln(2\cdot\pi)}{2\cdot\ln(N)}} \end{array} \tag{78}$$

$$\text{erf}^{-1}\left(1-\frac{2}{N}\right) \lessapprox \sqrt{\ln\left(N\right)} \quad , N \gg 1 \tag{79}$$

$$x \gtrapprox \left(\frac{L}{2}\right)\cdot\left(1-\sqrt{\frac{2\cdot\ln(N)}{L}}\right) = L\cdot\left(\frac{1}{2}-\sqrt{\frac{\ln(N)}{2\cdot L}}\right) \quad , N \gg 1 \tag{80}$$

$$N \gtrapprox \exp\left(\frac{L}{2}\right) \tag{81}$$

B.2 One Uncertain Bit in the Population

B.2.1 Average Sequence Variance

Suppose that every individual in a generation has an identical sequence to every other except for a bit. In some portion of the population, p, the bit is a 1 and in the remaining portion, $1-p$, it is a 0. Thus, the average sequence will be p in this bit. Thus, the average sequence variance can be calculated (82). The result (83) is unsurprisingly the variance of a Bernoulli random variable (as this is a Bernoulli random variable A.2). If the bit is made uniformly random, the average sequence variance will be $\frac{1}{4}$. Also, if the bit is fixed (either as 1 or 0), the average sequence variance will go to 0. As variances are additive, a variance of 1 could be attributed to 4 uniformly random bits in the population.

$$\begin{array}{rcl} \sigma^2 & = & p\cdot\left(1-p\right)^2+\left(1-p\right)\cdot\left(0-p\right)^2 \\ \sigma^2 & = & p\cdot\left(1-p\right)^2+\left(1-p\right)\cdot p^2 \\ \sigma^2 & = & p\cdot\left(1-p\right)\cdot\left(1-p+p\right) \\ \sigma^2 & = & p\cdot\left(1-p\right) \end{array} \tag{82}$$

$$\sigma^2_{Average\,Sequence\,Variance} = p\cdot\left(1-p\right) \tag{83}$$

Though, it is important to remember that the population is not really continuous, but discrete. So, instead, take N_1 to be the number of individuals with a 1. The average sequence variance can be easily be rewritten (84). This formula for average sequence variance is not particularly interesting or different from the continuous version in its own right. However, it does invite the question, what if 1 individual's sequence is different by 1 bit in the population. Taking $N_1 = 1$, it can be found that the variance of 1 bit in 1 individual is well approximated as $\frac{1}{N}$ (85). Given that $N = 1000$ is held constant, it is found that $\frac{1}{N} = 10^{-3}$. Of course, as all results are averaged over 50 trials, the average sequence variance can be as low as $2\cdot 10^{-5}$ (where this is 1 bit different in 1 individual in 1 trial). However, as will be seen, average sequence variances that go below 10^{-3} tend to go to 0 quickly.

$$\sigma^2_{Average\,Sequence\,Variance} = \left(\frac{N_1}{N}\right)\cdot\left(\frac{N-N_1}{N}\right) \tag{84}$$

$$\sigma^2_{Average\,Sequence\,Variance} = \frac{N-1}{N^2} \gtrapprox \frac{1}{N} \tag{85}$$

B.3 Equally dispersed clones with one bit different from the ideal sequence (similar to Muller's ratchet)

B.3.1 Average Sequence Variance

Take, for instance, a case where there are L groups of individuals; where, L is the sequence length. Within each group, all individuals have identical sequences. No two individuals between groups have the same sequence. Each of these groups of individuals is identical to the ideal sequence except for 1 bit that is different. For simplicity, all groups will have identical sizes. Also, for simplicity, assume the ideal sequence is all 0's. Thus, the bits that are different will be 1's. As a result, the average sequence will have $\frac{1}{L}$ in every place (note that the population size, N, will vanish everywhere as all group sizes will be multiples of $\frac{N}{L}$, but averaging will divide everything by N). The average sequence variance for this case can be calculated (86). The result is found to be well approximated by $\frac{1}{L}$ (87), which is similar to the result found for having one different bit in the population. Given that $L = 100$ is held constant, it is found that $\frac{1}{L} = 10^{-2}$. As before averaging over 50 trials can be included, which results in $2 \cdot 10^{-4}$. So, this is pretty important phenomenon and more complex clones have been ignored. It is reasonable to assume this is maximal variance in terms of the sizes of the groups. This is a somewhat extreme case of clonal competition. However, it can occur close to equilibrium and is related to Muller's ratchet. Each of these mutation is unfavorable, but the fitness will only be off from perfection by a factor of e^s; so, the cost is low. Regardless, selection is ineffective at removing them and mutation may add a couple more unfavorable mutations assuming the cost in fitness isn't enough to completely remove such individuals from the population. Recombination should be effective at removing this sort of problem (by replacing unfavorable sections in one individual by favorable sections in another individual) and it would appear to be reducing variance when this happens.

$$\begin{aligned} \sigma^2 &= \left(\tfrac{L-1}{L}\right) \cdot \left(0 - \tfrac{1}{L}\right)^2 + \left(\tfrac{1}{L}\right) \cdot \left(1 - \tfrac{1}{L}\right)^2 \\ \sigma^2 &= \left(\tfrac{1}{L}\right) \cdot \left((L-1) \cdot \left(\tfrac{1}{L}\right)^2 + \left(\tfrac{L-1}{L}\right)^2\right) \\ \sigma^2 &= \left(\tfrac{L-1}{L^3}\right) \cdot (1 + L - 1) \\ \sigma^2 &= \tfrac{L-1}{L^2} \end{aligned} \tag{86}$$

$$\sigma^2_{Average\,Sequence\,Variance} = \tfrac{L-1}{L^2} \gtrapprox \tfrac{1}{L} \tag{87}$$

B.4 Competition between two types of clones

B.4.1 Average Sequence Variance

Consider a situation where two clones compete and all individuals in the population are one of the two types of clones. Certainly, some complexity comes into the problem when considering how many differences exist between the clones. However, if they are assumed to be identical except for two bits (it must always be a multiple of two for them to have the same fitness as a pair of opposing bits must be flipped together). It can quickly be realized that the average sequence variance will be twice that of a Bernoulli random variable (25). So, this relationship for the average sequence variance can be generalized for C pairs of different spins (88). This variance must be discretized again and the formula will essentially be the same as before (89). The question is what is the average sequence variance due to having one clone in a population that is otherwise identical and the result is much the same (90). For simplicity in the calculations to follow, take C to be 1. Given that $N = 1000$ is held constant, it is found that $\frac{2}{N} = 2 \cdot 10^{-3}$. Taking into account the averaging over 50 trials, the average sequence variance can be as low as $4 \cdot 10^{-5}$. Thus, it is important to note that one nearly identical clone (off by one pair of bits) has a very similar effect to having one mutation different in the population. In order to tell the two apart, it is important to realize that clones have the same fitness and hence same ideal sequence distance. Furthermore, clones are not effected differently by

selection. So, selection amongst clones will only be effected by genetic drift. As a result, when only clones are present, the average sequence variance will change due to the distribution of clones present, but the ideal sequence distance will not. However, the disentanglement of mutants and clones when both are present is not so trivial.

$$\sigma^2_{Average\,Sequence\,Variance} = 2 \cdot C \cdot p \cdot (1-p) \tag{88}$$

$$\sigma^2_{Average\,Sequence\,Variance} = 2 \cdot C \cdot \left(\tfrac{N_1}{N}\right) \cdot \left(\tfrac{N-N_1}{N}\right) \tag{89}$$

$$\sigma^2_{Average\,Sequence\,Variance} = \tfrac{2 \cdot C \cdot (N-1)}{N^2} \gtrapprox \tfrac{2 \cdot C}{N} \tag{90}$$

C Code Documentation

The code is broken up into four main sections. The first section is called Generics Library and it defines a variety of basic types that are of use both in this code and in other applications (information about it can be found in the following subsection C.1). The second section is called Bacteria Multiplication Library and it defines more specific types of use in the simulation (information about it can be found in the following subsection C.2). The third section is called Bacteria Multiplication and this is what actually sets up and executes the simulation (information about it can be found in the following subsection C.3). The fourth and final section is Bacteria Multiplication Log Parser and this consists of a variety of scripts involved in reading log files from the simulation, averaging them, and reading data in for graphing, etc (information about it can be found in the following subsection C.4). Only a summary of how each of these groupings work will be included here. For more details, the code itself has documentation embedded in it using Doxygen[95]. The code is commented in the Doxygen way, but it also contains an html file in each code directory. Clicking on it will redirect your browser to the documentation for that package. Linking is included between packages; so, if all the packages are present in the same directory then a function or object mentioned in one such, but in a different library, will be linked to. This makes easy to see the larger picture as to how the code is connected without getting to lost in details.

C.1 Generics Library

There are a few main categories worth mentioning in the Generics Library, but the details of organization will be skipped simply due to the vastness of the library and by the fact that the Doxygen documentation does a much better job. By its very nature, this library can and should be used in just about everything. Largely, it has come about by improving the simulation and also by providing various optimization to make sure the simulation runs with minimal obstructions.

A variety of aliases are used for standard C++ types. These aliases can be change in the Generics Library and it will impact everything everywhere else. The main types used are real_type (used anywhere a floating-point would be expected), size_type (used anywhere an index for a container is expected), int_type (used anywhere a signed integer is expected), and num_type (at present the same as size_type, but is meant to be used anywhere an unsigned integer is expected).

Type introspection is encouraged in the latest C++ standard C++11. Though, as C++11 did not included Concepts (a way of placing restrictions on template parameters), developers are forced to write really good documentation on how template parameters should behave and/or attempt to implement their own forms of type introspection. Combined with static_asserts type introspection can be concept-like. Some facilities are implemented here that are community inspired that mainly use SFINAE technique (Substitution failure is not an error) combined with macros to prepare the SFINAE tests. A few tricks are included like _type_ to help shorten expressions.

The Standard Template Library (STL) provides a rich set of objects to use and includes many new ones in the C++11 standard like std::function, std::bind, etc to deal with a variety of function types (function pointers, method pointers, function objects (or functor), lambdas, etc). However, there are still some gaps and some undesirable properties. For instance, Python allows generator objects, which are very useful for iteration as they have state and each time they are called return a new value and suspend until they exit. Instead of having objects like this, C++ encourages the use of iterators that behave similarly, but normally have to be connected to some sort of container on the backend. It also means for every task, a new iterator must be written. Iterators have many functions accessed through operators; so, writing iterators results in generating a lot of repeated code for slightly different effect. The only way that the STL provides to get around this is essentially performing copy operations with some operations on an object between to objects. This will always be $O(N)$ both in computation and memory. In addition, whatever new container is generated will likely b destructed by the end of the loop making the waste more frustrating. As such, these solutions are undesirable.

The Generics Library contains a C++ style of generators that behave like STL containers and can be used anywhere STL containers can. Also, the generators provide forward iterators (one directional) that can be used anywhere iterators are. A few basic types of generators are included. Another related problem is when dealing with an existing container and its iterators. Frequently, one wants to perform a simple operation on every element of an iterator either to change the element's state and/or to generate the actual desired quantity. Normally, this is no problem as the for loop occurs in the realm accessible to the program. However, in some cases, this is not true. Certainly, a generator, as previously described, could be used to solve this problem. Though, using a generator restricts the iterators to being forward iterators and loses the efficiency of random access iterators that can use indexing, or even simple bidirectional iterators. This can result in inefficiency occurring in the code using the iterators as it tries to compensate for unavailable feature. Instead, it would be more desirable to take the existing iterators and adapt them so they perform the call on access. These iterator adaptors are provided for, in particular, GenericOperationIterator takes any callable object that can be called using function syntax on the object returned by dereferencing the iterator and its result is returned.

When using the GenericOperationIterator with std::function objects containing nothing more than method pointers, noticeable inefficiency was discovered in comparison to providing the method pointer as a template parameter to MethodIterator (basically the same as GenericOperationIterator, but for a method pointer passed as a template parameter). However, the syntax of MethodIterator is a bit frustrating to understand and use. Also, it would preferable to not have a separate iterator adapter for functions and methods. Furthermore, it would be nice not to have to

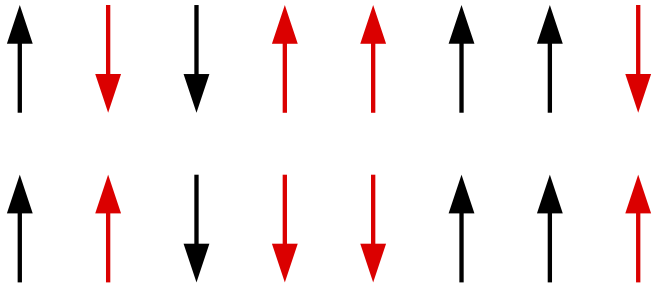

Figure 20: Example of the hamming distance: First BitString is on top. Second BitString is on bottom. Bits are represented by spins (up is 1 and down is 0). Matches (with value 0) are colored black and mis-matches (with value 1) are colored Red. There are 4 matches (with value 0) and 4 mis-matches (with value 4), which sum to give the result of the hamming distance as 4.

determine whether the method is acted on by an object or a pointer. Instead, there should be some object that performs compile-time distinctions to determine how to invoke its pointer type. Ideally, this invocation should be such that the compiler can optimize it out entirely.

These features are implemented by template callables. The heavy lifting is primarily done by CallablePointerTypeRunner, which, using template specialization, provides a static method run that takes the callable pointer as the first argument, the object (if it is a method) as the second argument (is able to distinguish between pointer and non-pointer types), and the remainder are arguments for the callable pointer. An actual type can be created that has the callable pointer in it at compile-time as well as providing methods that should be either executed at compile time or optimized out. This is done using with BoundCallables using either makeBoundCallable or getBoundCallable (the result is the same) and details are in the documentation. Also, objects can be constructed and used at compile-time. There are other types of callable objects as well. All of them try to do as much as possible at compile-time so that their imprint is minimal at runtime.

There are other container-like objects like the GenericParameters, which is used extensively in coding the simulation, and BitString, which is probably the single most essential part of the simulation. Briefly, GenericParameters is not intended to be used its self, but is intended to be derived from. Essentially, GenericParameters is a nicer version of std::tuple in that it provides a way to contain a variety of unlike arguments together. Versions that derive from it have methods named after the parameter they are requesting. It is designed to be an argument for a constructor. The main reason it is used instead of passing the parameters directly is that changes were constantly being made to the simulation in terms of what parameters should be passed and which objects should need them. Thus, parameter collections solves this as rearrangement of the ordering of parameters does not effect the ability to access them simply. Also, construction of the parameter collection occurs generally at one location. As a result, changing parameters sent becomes easy to do.

An IDHandler is defined as well, which ensures that every object of a specified type has a unique ID. This is useful when trying to track how many offspring an individual in the simulation had. It could be used also to determine, which strains become most prominent over time as well as identifying bottlenecks. Also, it can be extended to track multiple colonies as well.

BitString is a wrapper of a std::vector that contains an array of bit-like objects. Note that std::vector<bool> is a specialized class that optimizes memory efficiency by performing bit twiddling operations; thus, a bit-like object has been used in place of bool to avoid inefficiencies caused by the specialization. BitString implements a distance, which is the hamming distance[6] and counts the number of mis-matched bits. An example of the hamming distance taken between two BitStrings is shown in the figure Figure 20 on page 48.

Also, BitStrings have a dot product operation akin to that of mathematical vectors. It is easiest to think of the bits as spins in this case. For the BitString, adjacent bits that are aligned (or matching) get a value of 1 and bits that are unaligned (or mis-matching) get a value of -1. The result of the dot product is the sum of all matches and mis-matches. Alternatively, the dot product

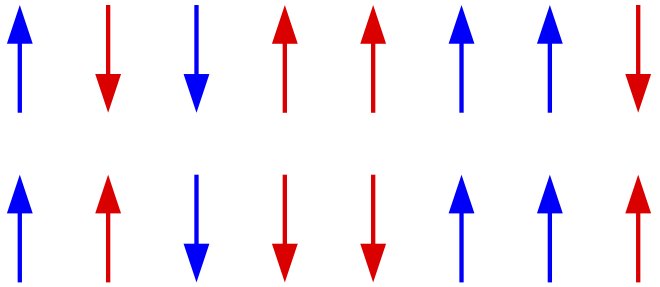

Figure 21: Example of the BitString's Dot Operation: First BitString is on top. Second BitString is on bottom. Bits are represented by spins (up is 1 and down is 0). Matches (with value 1) are colored blue and mis-matches (with value -1) are colored Red. There are 4 matches (with value 4) and 4 mis-matches (with value -4), which sum to give the result of the dot operation as 0.

can be thought of counting all the matches and mis-matches, then taking the difference between the two quantities. The figure contains an example of this operation Figure 21 on page 49.

There is a clear relationship between the hamming distance and the dot operation. If two BitStrings of length L (number of "bits") are identical, then the hamming distance is 0 and the dot product is L. If two BitStrings (also of length L) have only one mis-match between them, then the hamming distance is 1 and the dot product is $L - 2$. The reason the dot product decreases by 2 is that creating a mis-match also means removing a match, which also corresponds to the difference in the value of a match and mis-match for the dot product. From these examples, it can be seen that given the length of two BitStrings is L and the hamming distance between them is d; the result of the dot product (represented by f) is given by the following formula (91). BitString contains other operation as well and the documentation with the code can help clarify those. BitString contains other operation as well and the documentation with the code can help clarify those.

$$f = L - 2 \cdot d \tag{91}$$

One of the major aspects of the simulation that was very important for efficiency and made it possible to run effectively on a multiprocessor machine is a thread pool. The generation of threads is very resource intensive. So, instead of having every task run on a new thread, it is preferable to reuse the same threads to run new tasks. This is known as a thread pool and the threads being reused are worker threads. Unlike other pool concepts (i.e. memory pools), the thread pool is somewhat difficult to implement because the object don't have easy access to each other once in separate threads. As a result, communication between the external thread pool (the only object the developer uses) and the worker threads is mediated through the task queue. Thus, the task queue needs to be carefully designed avoid race cases (where two threads manipulate a shared variable at the same time). This is done by adding locking mechanisms to ensure only one thread has access at a time.

To help further clarify the dynamics of the thread pool (called ThreadPool in the code), a figure of its workflow has been included 22. Initially, when the ThreadPool is constructed, it builds the TaskQueue. Afterwards, it builds the WorkerThreads and provides them with access to the TaskQueue. As there are no Tasks in the TaskQueue initially and the TaskQueue is open for adding Tasks, the WorkerThreads block and wait for notification. As soon as the ThreadPool is called to add a Task, it passes it on to the TaskQueue to package and store. Once the TaskQueue stores the Task, it attempts to pass it to a waiting WorkerThread. If no WorkerThreads are waiting, the Task sits in the TaskQueue until a WorkerThread requests a new Task. Meanwhile, the WorkerThreads simply run Tasks and attempt to get new ones. Once the ThreadPool enters its destructor, it waits on the TaskQueue to empty. After finishing all the Tasks and blocking all the WorkerThreads, the TaskQueue wakes all the WorkerThreads for destruction by issuing empty Tasks, after which the

TaskQueue and then ThreadPool are destructed.

Random number generation is also very important for the simulation. Though random number generation must be done efficiently, it also must be done with multithreading in mind. All random number generators designed in the Generics Library, keep multithreading in mind. Seeds are generated using the DefaultSeedHandlerEngine, which adapts RandomDeviceSeedHandler (either uses a hardware based random device or a Mersenne twister seeded with the current time) or TimeSeedHandler (simply provides the current time in the same fashion as a general seed handler would) depending on specification. The DefaultSeedHandlerEngine behaves like a mixture of a random number generator and a seed handler from STL for simplicity. Both are designed to be lock-free, meaning that each instance in each thread interacts independently of the other instances in other threads.

Though, the DefaultSeedHandlerEngine need not be dealt with directly as it is bound in an EngineTypeSeedHandler. This EngineTypeSeedHandler generates a Mersenne twister (or other engine) using the DefaultSeedHandlerEngine as the source of seeds. In addition, the EngineTypeSeedHandler automatically reseeds when a preset number of draws is reached. This EngineTypeSeedHandler is bound within the RandomValueGenerator, which is used for generation of all random numbers by all distributions in the Generics Library.

In each of the binds, objects are generated once per thread. However, multiple pointers may exist in each thread to the shared object. This means that in each thread all engines are built once in the beginning and persist until the end, which keeps the overhead down. Sharing between threads can result in simulations not running; so, it is still preferable to have one instance per thread. Furthermore, by keeping engines on a per thread basis, new simulations run in an existing thread do not need to generate new engines.

Event notification is very important for the control flow of the code. Though it may seem difficult to follow at first, it is very natural to follow and improves the maintainability and simplicity of the code. Also, it ensures a sort of control flow safety. Pieces of the structure are set up in the Generics Library, but most of the implementation is in Bacteria Multiplication Library. There are two main events that are used: Time Event and Fitness Function Event. However, more could be added as necessary using the same structure. Also, objects currently not using the notification system can simply implement it.

There are two main steps of the event notification system. The first step is handlers (objects that want to be notified of the event) register for the event, which is depicted in the figure 23. Initiation begins from the controller level, which registers itself with the event handler. The event handler adds the controller to the top of its stack of registered handles. Afterward, it calls the controllers event register method, which registers all the objects that it contains that need to be notified. This process continues until there is nothing else that needs to be registered. As each object registers it gets added to the top of the event handler's stack.

The second step includes the initiation of the event and subsequent notification of all registered listeners to update, which is depicted in the figure 24. The event is triggered by the controller (or possibly another event). After the event occurs, the event handler grabs the first handle off the stack notifies it of the event. The handle responds to this event in some fashion. Afterwards, control is passed back to the event handler, which pulls the next handle off the stack. This process continues until all handles have been pull from the stack and the stack is empty. At this point, control has been passed back to the controller or whatever initiated the event. Before the event happens again, registration must occur again. As all the handles are stored in stack, LIFO (last-in, first-out) ordering of the handles. By having LIFO ordering, it is ensured that the lowest level handles are updated first and the highest-level handles are updated last; thus, when reaching higher-level handles, it is ensured that their components have already been updated.

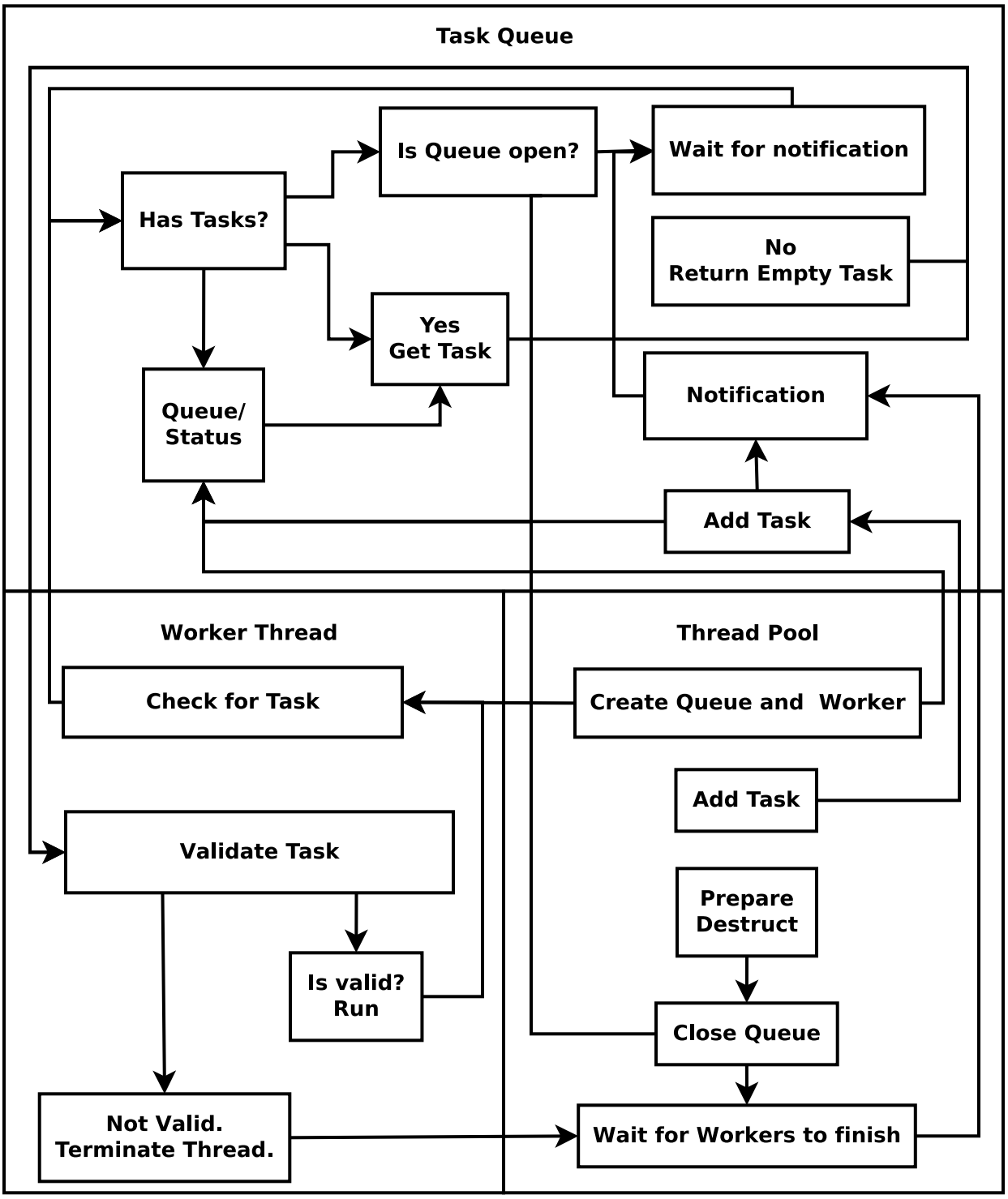

Figure 22: Thread Pool Process Flow

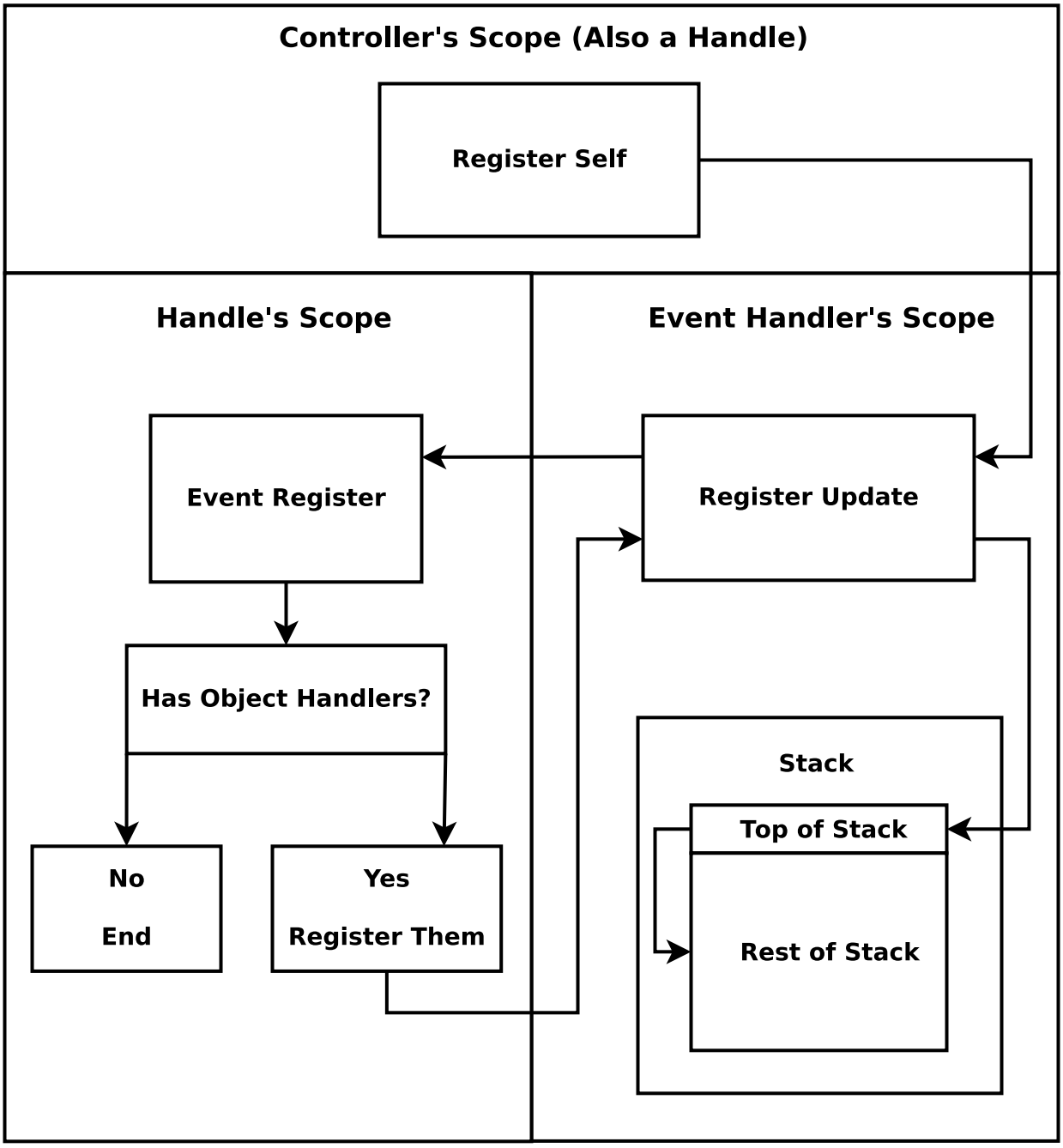

Figure 23: Event Handle Registration Process Flow

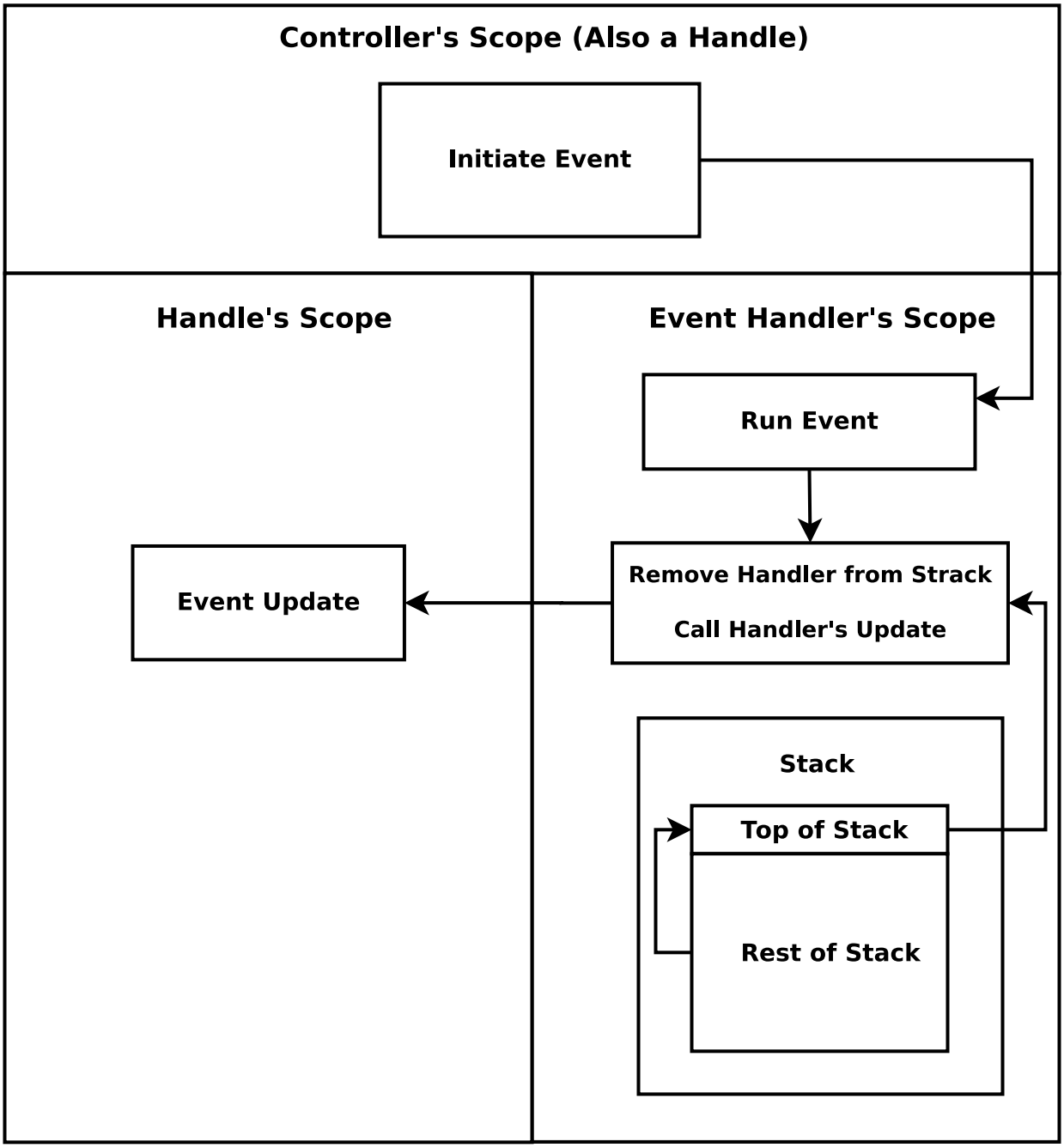

Figure 24: Event Handle Update Process Flow

C.2 Bacteria Multiplication Library

A brief overview of the objects in Bacteria Multiplication Library will be provided here. More details about their make-up can be found in documentation provided with the source code. The Bacteria Multiplication Library defines all objects in use in the simulation and how they interact. All objects in the library can be constructed by using their parameters (GenericParameters' derivatives C.1). As mentioned in the Generics Library, the parameters provide a way of easily initializing objects and easily accessing the correct quantities. In addition, most objects can be copied constructed as well.

The lowest level object used to represent individuals is the Bacterium. Inside the Bacterium (in addition to its parameters), are its DNA (represented by a BitString), its Fitness (updated by a FitnessFunction), and its parent's or parents' ID's as provided by the Bacterium IDHandler C.1. The Bacterium is able to be copy constructed or it can be reproduced by one or two parents. The Bacterium's ability to mutate and recombine are controlled by objects in its parameters; however, this may change. Updating the Bacterium's fitness, is done externally by a FitnessFunction. This FitnessFunction acts on any change in time and on the DNA provided to determine the Fitness.

All Bacteria belong to a Generation, which contains individuals constructed at the same time (as this follows a Wright-Fisher Model). A new Generation can then be constructed using the previous Generation as the parents. Offspring pick their parents with a probability proportional to their parents' fitnesses. If the offspring pick a parent that is able to perform recombination, then it will recombine with the recombination probability given by the parent picked. In both cases, recombined or not, the DNA is mutated before it is accepted in the new offspring. The Generation provides access for a FitnessFunction to register and update all its Bacteria. The Generation also tracks quantities like averages (ideal sequence distance average, average sequence, average fitness) and variances (average sequence variance).

The Generation is contained in the Colony, which is capable (currently suppressed) of tracking the history of the individuals. The Colony also performs output of relevant average quantities. The Colony also provides ways for a FitnessFunction to register and update. Also, the Colony provides ways for an EventClock to register and update.

The PetriDish manages the entire simulation. It generates the Colony and as well as increases the EventClock, which in turn calls the FitnessFunction after the new Generation is built. Also, the PetriDish writes out the output of the Colony to some form of buffered output (std::ostream) determined by the user. Normally, this should be a file stream, i.e. std::ofstream, but it will use any std::ostream derived type. The PetriDish uses a FitnessFunctionEvent to iterate over all Bacterium and update the Fitness of each one according to the FitnessFunction used. Further, the PetriDish ensures that events are properly ordered in time. In other words, the Colony generates a new Generation based on the old one and then the FitnessFunction updates the Fitnesses of the Bacteria only after dealing with the changes triggered by the EventClock. During this time, the FitnessFunction is updated if it varies due to time; thus, avoiding duplicate calculations when determining the Fitness of each Bacterium.

C.3 Bacteria Multiplication

This program simply includes the two previous libraries Bacteria Multiplication Library C.2 and Generics Library C.1. Variations could be made for different types of trials very easily due to the clean nature of the libraries provided. There is simply one function run_trial in this program that takes a variety of different parameters and creates a PetriDish using them. This PetriDish runs for the time specified. All data is written to an output file that includes its parameters in its title name. The main function creates a ThreadPool for running different trials. Then, it calls various combinations of run_trial that run and output their data. Debugging options are possible using the "DEBUG" preprocessor defined variable in DebugTools.

C.4 Bacteria Multiplication Log Parser

After completing a set of runs, the data needs to be processed into some more meaningful form. To do this, all trials with the same parameters are averaged and their standard deviations are calculated and are written out to a new set of fewer files. Their format is largely the same as the original files. This is accomplished by AverageBacteriaLogs.py (does not use period parameter) and AverageBacteriaLogs_P.py (uses period parameter). These two scripts probably can be combined in some fashion. Though, these averaged files are condensed they still do not give a good picture of the data. A good first step is to use the script IdealSequenceDistanceAverageSequenceVariance-TimeGraphs.py, which can generate a graph in Matplotlib that shows the ideal sequence distance average and average sequence variance versus time graphs shown in the same format as those shown here. It can also graph other things in the same style. It can be changed to due linear, log, or symlog (log until some small threshold is reached, then linear below). To get a better idea of how the grid style graphs are generated look at MatplotlibGraphs.py; however, no detailed explanation will be provided here as there is plenty of information about Matplotlib. Also, some details, will provided in the code/Doxygen documentation. In order for text wrap to work, the graphs must be done in some form of iPython [79] shell in interactive mode. Spyder [19] was used for generating the graphs in this thesis and is open source. It works on many platforms with relative ease; however, there are plenty of other options that would work.

D Algorithmic Details

D.1 Probability Distributions

D.1.1 Preliminaries

In all cases, a Mersenne Twister is used as a random number generator[62]. In particular, a 64-bit variant of the Mersenne Twister is used as provided through the C++11 standard. In the C++11 standard, a function (generate_canonical) is used to convert random number generator output into a standard uniform value A.8 of some floating-point type. Once a standard uniform value is created, it can be mapped to the respective distribution using an inverse cumulative distribution function.[35] For simplicity, $u \in [0, 1)$ will represent a standard uniform variate A.8.

D.1.2 Bernoulli Distribution Mapping

This distribution is simple to map from a standard uniform variate 92.

$$M(u) = \begin{cases} 0 & , u \geq p \\ 1 & , u < p \end{cases} \tag{92}$$

D.1.3 Uniform Distribution Mapping

Mapping a standard uniform variate onto a uniform distribution A.8 is really a matter of rescaling and shifting (93).

$$M(u) = (b - a) \cdot u + a \tag{93}$$

D.1.4 Optimized Bernoulli Distribution Mapping

As shown with the Bernoulli distribution before, mapping a standard uniform variate into a Boolean value is simple D.1.2. However, drawing each random number is costly. So, it would be better if the uniform variate could be reused. In order to reuse the uniform variate, it should be remapped to its uniform variate range. This way the same map for a Bernoulli distribution (92) can be reused (94). Remapping can be accomplished as so (95). However, the uniform variate does not have truly infinite precision as real numbers do. Thus, some idea of what remaining information exists in the random number should be kept, which can be done by keeping track of a scale (96). Actually, a product of all previous scales until the last draw must be stored. The log of this scale represents the amount of information that has been used.

Using this information, at some point a new random number must be drawn. This will have to be determined by whether the scale can distinguish between p and $1 - p$ accurately. Assume briefly (and this can be repeated with this condition reversed) that p is smaller than $1 - p$. Then, the range from $[0, 1)$ can be subdivided into $\frac{1}{p}$ regions. These regions are effectively the smallest required bins. Thus, if there is some smallest machine value, ϵ, which determines the smallest representable difference between two values. Then, the rescaled region can be no smaller that $\frac{\epsilon}{p}$ (otherwise the regions will not all be distinguishable). As a result, if the rescaled region goes below $\frac{\epsilon}{p}$, it must be redrawn. After redrawing, the scale can be reset to unity. Following this method, a standard uniform variate can be reused with either the same or different Bernoulli distributions until exhausted.

$$M(u) = \begin{cases} 0 & , u \geq p \\ 1 & , u < p \end{cases} \tag{94}$$

$$R(u) = \begin{cases} \frac{u-p}{1-p} & , u \geq p \\ \frac{u}{p} & , u < p \end{cases} \tag{95}$$

$$S(u) = \begin{cases} 1-p & , u \geq p \\ p & , u < p \end{cases} \tag{96}$$

D.1.5 Geometric Distribution Mapping

This distribution is a little trickier to map, but it can be done. The cumulative distribution function needs to be inverted. Due to the process of converting a real value to a discrete value, a few changes must be made. For instance, the cumulative distribution function for $n-1$ will be used as this determines the minimum probability. Afterwards, all real values will be truncated using the floor function, which ensures than map to the minimum probability. The calculation of the map is shown (97). The map is found to be (98). Note that if $u = 0$, then $n = 0$. Also, note that if $u = p$, then $n = 1$. If $u < p$, then $n = 0$. Finally, if $u = 1$, then $n \to \infty$. These cases show the map demonstrates the right behavior.

$$\begin{array}{rcl} u & = & 1-(1-p)^n \\ 1-u & = & (1-p)^n \\ \ln(1-u) & = & n \cdot \ln(1-p) \\ n & = & \frac{\ln(1-u)}{\ln(1-p)} \\ n & = & \left\lfloor \frac{\ln(1-u)}{\ln(1-p)} \right\rfloor \end{array} \tag{97}$$

$$M(u) = \left\lfloor \frac{\ln(1-u)}{\ln(1-p)} \right\rfloor \tag{98}$$

6 References

[1] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions: With Formulas, Graphs and Mathematical Tables*. A Wiley-Interscience publication. John Wiley & Sons, Incorporated, 1972.

[2] Denis Arutyunov and Laura S. Frost. F conjugation: Back to the beginning. *Plasmid*, (0):–, 2013.

[3] N. Balakrishnan and V.B. Nevzorov. *A Primer on Statistical Distributions*. Wiley, 2004.

[4] N. H. Barton and B. Charlesworth. Why sex and recombination? *Science*, 281(5385):1986–1990, 1998.

[5] Mark A Bedau and Norman H Packard. Evolution of evolvability via adaptation of mutation rates. *Biosystems*, 69:143 – 162, 2003.

[6] Paul E. Black. Hamming distance. In Paul E. Black, editor, *Dictionary of Algorithms and Data Structures [online]*. U.S. National Institute of Standards and Technology, http://www.nist.gov/dads/HTML/HammingDistance.html, 2006.

[7] Sebastian Bonhoeffer and Peter F. Stadler. Error thresholds on correlated fitness landscapes. *Journal of Theoretical Biology*, 164(3):359 – 372, 1993.

[8] John Cairns, Julie Overbaugh, and Stephan Miller. The origin of mutants. *Nature*, 335(6186):142–145, September 1988.

[9] Ines Chen and David Dubnau. Dna uptake during bacterial transformation. *Nat Rev Micro*, 2(3):241–249, March 2004.

[10] Inês Chen, Peter J. Christie, and David Dubnau. The ins and outs of dna transfer in bacteria. *Science*, 310(5753):1456–1460, 2005.

[11] Freddy B. Christiansen, Sarah P. Otto, Aviv Bergman, and Marcus W. Feldman. Waiting with and without recombination: The time to production of a double mutant. *Theoretical Population Biology*, 53(3):199 – 215, 1998.

[12] Alvin J Clark. Recombination deficient mutants of e. coli and other bacteria. *Annual review of genetics*, 7(1):67–86, 1973.

[13] Alvin J. Clark and Ann Dee Margulies. Isolation and characterization of recombination-deficient mutants of escherichia coli k12. *Proceedings of the National Academy of Sciences*, 53(2):451–459, 1965.

[14] Margaret Clarke, Lucinda Maddera, Robin L. Harris, and Philip M. Silverman. F-pili dynamics by live-cell imaging. *Proceedings of the National Academy of Sciences*, 105(46):17978–17981, 2008.

[15] Frederick M Cohan. What are bacterial species? *Annual Reviews in Microbiology*, 56(1):457–487, 2002.

[16] Elisheva Cohen, David A. Kessler, and Herbert Levine. Recombination dramatically speeds up evolution of finite populations. *Phys. Rev. Lett.*, 94:098102, Mar 2005.

[17] Tom M Conrad, Nathan E Lewis, and Bernhard O Palsson. Microbial laboratory evolution in the era of genome-scale science. *Mol Syst Biol*, 7:–, July 2011.

[18] Rosa-Lee Cooke.

[19] Carlos Cordoba. Spyder is the scientific python development environment. online, 2012.

[20] Edward C Cox. Bacterial mutator genes and the control of spontaneous mutation. *Annual review of genetics*, 10(1):135–156, 1976.

[21] Fernando de la Cruz and Julian Davies. Horizontal gene transfer and the origin of species: lessons from bacteria. *Trends in Microbiology*, 8(3):128 – 133, 2000.

[22] Harold P. de Vladar and Nicholas H. Barton. The contribution of statistical physics to evolutionary biology. *Trends in Ecology & Evolution*, 26(8):424 – 432, 2011.

[23] Xavier Didelot and Martin C.J. Maiden. Impact of recombination on bacterial evolution. *Trends in Microbiology*, 18(7):315 – 322, 2010.

[24] J W Drake. Spontaneous mutation. *Annual Review of Genetics*, 25(1):125–146, 1991. PMID: 1812804.

[25] John W. Drake. The distribution of rates of spontaneous mutation over viruses, prokaryotes, and eukaryotes. *Annals of the New York Academy of Sciences*, 870(1):100–107, 1999.

[26] John W Drake, Brian Charlesworth, Deborah Charlesworth, and James F Crow. Rates of spontaneous mutation. *Genetics*, 148(4):1667–1686, 1998.

[27] Barbara Drossel. Biological evolution and statistical physics. *Advances in Physics*, 50(2):209–295, 2001.

[28] Manfred Eigen, John McCaskill, and Peter Schuster. Molecular quasi-species. *The Journal of Physical Chemistry*, 92(24):6881–6891, 1988.

[29] W.J. Ewens. An interpretation and proof of the fundamental theorem of natural selection. *Theoretical Population Biology*, 36(2):167–180, October 1989.

[30] Joseph Felsenstein. The evolutionary advantage of recombination. *Genetics*, 78(2):737–756, 1974.

[31] Daniel Fisher, Michael Lässig, and Boris Shraiman. Evolutionary dynamics and statistical physics. *Journal of Statistical Mechanics: Theory and Experiment*, 2013(01):N01001, 2013.

[32] Ronald Aylmer Fisher. *The genetical theory of natural selection: a complete variorum edition.* Oxford University Press, 1999.

[33] Walter Fontana, Peter F. Stadler, Erich G. Bornberg-Bauer, Thomas Griesmacher, Ivo L. Hofacker, Manfred Tacker, Pedro Tarazona, Edward D. Weinberger, and Peter Schuster. Rna folding and combinatory landscapes. *Phys. Rev. E*, 47:2083–2099, Mar 1993.

[34] Christophe Fraser, Eric J. Alm, Martin F. Polz, Brian G. Spratt, and William P. Hanage. The bacterial species challenge: Making sense of genetic and ecological diversity. *Science*, 323(5915):741–746, 2009.

[35] J.E. Gentle. *Random Number Generation and Monte Carlo Methods.* Statistics and Computing. Springer, 2003.

[36] John H Gillespie. *Population genetics: a concise guide.* JHU Press, 2010.

[37] J. Peter Gogarten and Jeffrey P. Townsend. Horizontal gene transfer, genome innovation and evolution. *Nat Rev Micro*, 3(9):679–687, September 2005.

[38] Myron F. Goodman. Error-prone repair dna polymerases in prokaryotes and eukaryotes. *Annual Review of Biochemistry*, 71(1):17–50, 2002. PMID: 12045089.

[39] W D Hamilton, R Axelrod, and R Tanese. Sexual reproduction as an adaptation to resist parasites (a review). *Proceedings of the National Academy of Sciences*, 87(9):3566–3573, 1990.

[40] William D. Hamilton. Sex versus non-sex versus parasite. *Oikos*, 35(2):pp. 282–290, 1980.

[41] Matthew Hartfield and Peter D. Keightley. Current hypotheses for the evolution of sex and recombination. *Integrative Zoology*, 7(2):192–209, 2012.

[42] PaulG. Higgs and Glenn Woodcock. The accumulation of mutations in asexual populations and the structure of genealogical trees in the presence of selection. *Journal of Mathematical Biology*, 33(7):677–702, 1995.

[43] W. G. Hill and Alan Robertson. The effect of linkage on limits to artificial selection. *Genetics Research*, 8:269–294, 12 1966.

[44] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science and Engineering*, 9(3):90–95, 2007.

[45] Yu ichiro Tago, Masaru Imai, Makoto Ihara, Hironari Atofuji, Yuki Nagata, and Kazuo Yamamoto. Escherichia coli mutator δ pola is defective in base mismatch correction: The nature of in vivo {DNA} replication errors. *Journal of Molecular Biology*, 351(2):299 – 308, 2005.

[46] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[47] Nadav Kashtan, Elad Noor, and Uri Alon. Varying environments can speed up evolution. *Proceedings of the National Academy of Sciences*, 104(34):13711–13716, 2007.

[48] Stuart Kauffman and Simon Levin. Towards a general theory of adaptive walks on rugged landscapes. *Journal of Theoretical Biology*, 128(1):11 – 45, 1987.

[49] Stuart A. Kauffman and Edward D. Weinberger. The {NK} model of rugged fitness landscapes and its application to maturation of the immune response. *Journal of Theoretical Biology*, 141(2):211 – 245, 1989.

[50] John W. Kimball. *Biology*. Wm. C. Brown Publishers, 1994.

[51] Joachim Krug. Statistical physics of biological evolution. Published in the Proceedings of the 4th Warsaw School of Statistical Physics, ed. by B. Cichocki, M. Napiorkowski and J. Piasecki (Warsaw University Press, 2012), pp. 123-129, 2012.

[52] A Landy. Dynamic, structural, and regulatory aspects of lambda site-specific recombination. *Annual Review of Biochemistry*, 58(1):913–941, 1989. PMID: 2528323.

[53] R E Lenski, M Slatkin, and F J Ayala. Mutation and selection in bacterial populations: alternatives to the hypothesis of directed mutation. *Proceedings of the National Academy of Sciences*, 86(8):2775–2778, 1989.

[54] Ira Leuthäusser. Statistical mechanics of eigen's evolution model. *Journal of Statistical Physics*, 48(1-2):343–360, 1987.

[55] Bruce R. Levin and Carl T. Bergstrom. Bacteria are different: Observations, interpretations, speculations, and opinions about the mechanisms of adaptive evolution in prokaryotes. *Proceedings of the National Academy of Sciences*, 97(13):6981–6985, 2000.

[56] Bruce R. Levin, Véronique Perrot, and Nina Walker. Compensatory mutations, antibiotic resistance and the population genetics of adaptive evolution in bacteria. *Genetics*, 154(3):985–997, 2000.

[57] C. C. Li. Fundamental theorem of natural selection. *Nature*, 214(5087):505–506, April 1967.

[58] K B Low and D D Porter. Modes of gene transfer and recombination in bacteria. *Annual Review of Genetics*, 12(1):249–287, 1978. PMID: 106767.

[59] S. E. Luria and M. Delbrück. Mutations of bacteria from virus sensitivity to virus resistance. *Genetics*, 28(6):491–511, 1943.

[60] Laura R Marks, Ryan M Reddinger, and Anders P Hakansson. High levels of genetic recombination during nasopharyngeal carriage and biofilm formation in streptococcus pneumoniae. *mBio*, 3(5), 2012.

[61] Ivan Matic, François Taddei, and Miroslav Radman. Genetic barriers among bacteria. *Trends in Microbiology*, 4(2):69 – 73, 1996.

[62] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, January 1998.

[63] Richard Michod and Bruce Levin. *The Evolution of sex : an examination of current ideas.* Sinauer Associates, Sunderland, Mass, 1988.

[64] Richard E. Michod, Harris Bernstein, and Aurora M. Nedelcu. Adaptive value of sex in microbial pathogens. *Infection, Genetics and Evolution*, 8(3):267 – 285, 2008.

[65] P Modrich. Mechanisms and biological effects of mismatch repair. *Annual Review of Genetics*, 25(1):229–253, 1991. PMID: 1812808.

[66] Laurence A. Moran. Mutation rates. Blog, July 2007.

[67] H. J. Muller. Some genetic aspects of sex. *The American Naturalist*, 66(703):pp. 118–138, 1932.

[68] H.J. Muller. The relation of recombination to mutational advance. *Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis*, 1(1):2 – 9, 1964.

[69] Ville Mustonen and Michael Lässig. From fitness landscapes to seascapes: non-equilibrium dynamics of selection and adaptation. *Trends in Genetics*, 25(3):111–119, March 2009.

[70] Ville Mustonen and Michael Lässig. Fitness flux and ubiquity of adaptive evolution. *Proceedings of the National Academy of Sciences*, 107(9):4248–4253, 2010.

[71] Enrique Muñoz, Jeong-Man Park, and Michael W. Deem. Quasispecies theory for horizontal gene transfer and recombination. *Phys. Rev. E*, 78:061921, Dec 2008.

[72] Richard A. Neher and Boris I. Shraiman. Statistical genetics and evolution of quantitative traits. *Rev. Mod. Phys.*, 83:1283–1300, Nov 2011.

[73] Howard Ochman, Jeffrey G. Lawrence, and Eduardo A. Groisman. Lateral gene transfer and the nature of bacterial innovation. *Nature*, 405(6784):299–304, May 2000.

[74] Tomoko Ohta. The nearly neutral theory of molecular evolution. *Annual Review of Ecology and Systematics*, 23:263–286, January 1992.

[75] Travis E. Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007.

[76] Sarah P. Otto and Thomas Lenormand. Resolving the paradox of sex and recombination. *Nat Rev Genet*, 3(4):252–261, April 2002.

[77] Mitradas M Panicker and EG Minkley. Dna transfer occurs during a cell surface contact stage of f sex factor-mediated bacterial conjugation. *Journal of bacteriology*, 162(2):584–590, 1985.

[78] Su-Chan Park, Damien Simon, and Joachim Krug. The speed of evolution in large asexual populations. *Journal of Statistical Physics*, 138:381–410, 2010.

[79] Fernando Perez and Brian E. Granger. Ipython: A system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29, 2007.

[80] Massimo Pigliucci. Is evolvability evolvable? *Nat Rev Genet*, 9(1):75–82, January 2008.

[81] Adam Prügel-Bennett and Jonathan L. Shapiro. The dynamics of a genetic algorithm for simple random ising systems. *Physica D: Nonlinear Phenomena*, 104(1):75 – 114, 1997.

[82] C M Radding. Molecular mechanisms in genetic recombination. *Annual Review of Genetics*, 7(1):87–111, 1973. PMID: 4593311.

[83] Rosemary J. Redfield. Do bacteria have sex? *Nat Rev Genet*, 2(8):634–639, August 2001.

[84] Susan M. Rosenberg. Evolving responsively: adaptive mutation. *Nat Rev Genet*, 2(7):504–515, July 2001.

[85] R. Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006.

[86] Marcel Salathé, Roger D. Kouyos, Roland R. Regoes, Sebastian Bonhoeffer, and M. Van Baalen. Rapid parasite adaptation drives selection for high recombination rates. *Evolution*, 62(2):295–300, February 2008.

[87] Sasaki and Iwasa. Optimal recombination rate in fluctuating environments. (0016-6731 (Linking)):–, Feb 1987.

[88] Paul D. Sniegowski, Philip J. Gerrish, Toby Johnson, and Aaron Shaver. The evolution of mutation rates: separating causes from consequences. *BioEssays*, 22(12):1057–1066, 2000.

[89] Paul D. Sniegowski, Philip J. Gerrish, and Richard E. Lenski. Evolution of high mutation rates in experimental populations of e. coli. *Nature*, 387(6634):703–705, June 1997.

[90] Brian G Spratt, William P Hanage, and Edward J Feil. The relative contributions of recombination and point mutation to the diversification of bacterial clones. *Current Opinion in Microbiology*, 4(5):602 – 606, 2001.

[91] Taison Tan, LeonardD. Bogarad, and MichaelW. Deem. Modulation of base-specific mutation and recombination rates enablesfunctional adaptation within the context of the genetic code. *Journal of Molecular Evolution*, 59:385–399, 2004.

[92] P. Tarazona. Error thresholds for molecular quasispecies as phase transitions: From simple landscapes to spin-glass models. *Phys. Rev. A*, 45:6038–6050, Apr 1992.

[93] SymPy Development Team. *SymPy: Python library for symbolic mathematics*, 2008.

[94] Christopher M. Thomas and Kaare M. Nielsen. Mechanisms of, and barriers to, horizontal gene transfer between bacteria. *Nat Rev Micro*, 3(9):711–721, September 2005.

[95] Dimitri van Heesch. Doxygen. Online, August 2013.

[96] Michiel Vos. Why do bacteria engage in homologous recombination? *Trends Microbiol*, 17(6):226–232, June 2009.

[97] Gunter P. Wagner and Lee Altenberg. Perspective: Complex adaptations and the evolution of evolvability. *Evolution*, 50(3):pp. 967–976, 1996.

[98] Ying A. Wang, Xiong Yu, Philip M. Silverman, Robin L. Harris, and Edward H. Egelman. The structure of f-pili. *Journal of Molecular Biology*, 385(1):22 – 29, 2009.

[99] West, Lively, and Read. A pluralist approach to sex and recombination. *Journal of Evolutionary Biology*, 12(6):1003–1012, 1999.

[100] Claus O. Wilke and Thomas Martinetz. Adaptive walks on time-dependent fitness landscapes. *Phys. Rev. E*, 60:2154–2159, Aug 1999.

[101] Dr. Bob Winning. Recombination in bacteria, 2006.

[102] Inc. Wolfram Research. Inverse error function: Series representations (formula 06.29.06.0002), 10 2001.

[103] Glenn Woodcock and Paul G. Higgs. Population evolution on a multiplicative single-peak fitness landscape. *Journal of Theoretical Biology*, 179(1):61–73, March 1996.