Simulating Virus and Host Coevolution Using Evolutionary Computation

Laura Colbran, Samuel Greaves, Liz Shank

1 Introduction

Bacteria and viruses both cause disease in humans ranging from the common cold to deadly epidemics such as bubonic plague. However, there are many more species of both that don't interact directly with humans. One example of this is those species of viruses that can also cause disease in bacteria. The specialized viruses that do this are called bacteriophages, and are essentially packets of genetic information (DNA or RNA) enclosed in a protein capsule. Virus genomes are streamlined, containing only the genes necessary to replicate themselves and produce their protein capsules. With such limited materials to hand, they have to infect bacteria in order to actually carry out their replication. Infection is accomplished by landing on the bacteria's outer membrane and injecting the viral information into the cytoplasm (Figure 1) (Todar, 2012).

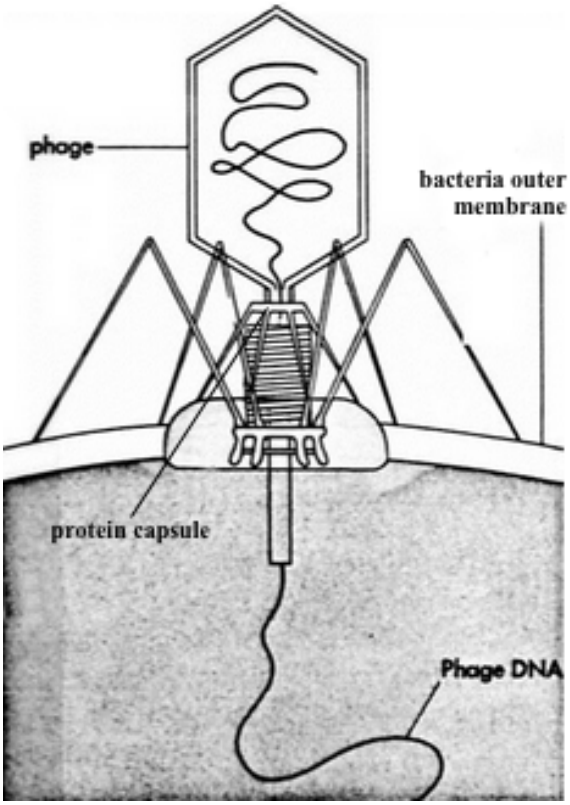

Figure 1: A bacteriophage injecting its genome into a bacteria. The protein capsule both protects the genetic information while between hosts and pierces the membrane to allow injection. Adapted from Todar, 2012.

Once the genome has gained entry to the bacteria, the virus hijacks the machinery the host uses to replicate its own genome. In one mechanism of this, the virus incorporates its genome into the host genome, where it gets replicated and transported as the host cell reproduces. A different mechanism uses the host's ribosomes to synthesize proteins to promote virulence as well as new proteins to encapsulate new copies of the virus. Once there are many copies of the virus, they synthesize lysozyme, which breaks open the bacterial cell membrane, killing the host and allowing many new copies of the virus to escape and spread to nearby cells.

For the second mechanism, because the success of the virus means the death of the host, bacteria have evolved methods of resisting viruses. Coevolution has led to an arms race between viruses and bacteria, as they constantly find new ways to get around each other. This is especially evident in one model of virus-bacteria interaction known as gene-for-gene interaction. In these systems, the virus has a number of genes encoding virulence proteins that help it invade a cell and replicate its genome. Likewise, the bacteria has complementary resistance proteins that recognize and destroy the viral proteins. Through many generations, the viruses keep evolving new genes to get around the bacterial defenses, while the bacteria evolve to block resist the new genes. The progression of the genomes is cyclic, since selective pressure only acts on those loci where members of the bacteria population are still susceptible to the virus, and where the virus can still infect some bacteria. As the pressure lifts, specific resistance and virulence genes disappear from the population, since both host and parasite are prioritizing the genes most relevant to their interaction at that point in time (Person, 1959). The other extreme for virus-host interaction is the matching allele model, where virus genomes are disguised by being exactly identical to segments of the host genome. If the host's defenses cannot tell the difference between invasive genetic information and their own, then the virus can successfully infect the cell. These two models are ends of a continuum of interaction; in mature, most interactions are some mixture of the two systems (Agrawal and Lively, 2002).

One interesting side-effect of this arms race is the relatively elevated mutation rate in many bacteria populations. In 2007, Pal et al. used a mixture of techniques to demonstrate that it was specifically coevolution with viruses that drove the increased mutation rate in susceptible bacteria. The mutation rate was mostly increased over time due to deleterious mutations in mismatch-repair genes, which code for proofreading proteins that normally check new copies of the bacterial genome for mutations and correct them. Without functioning mismatch-repair proteins, mutations are much more likely to passed on without being fixed. One of the tools Pal et al. used in their research was a genetic algorithm that simulated host-virus interactions, which they used specifically to model how mutation rates evolved over time. Their simulation behaved similarly to the predictions of the gene-for-gene model, with host mutation rate plateauing at the maximum allowed, while the frequencies individual genomes fluctuate over time as the selection pressure shifts in response to viral evolution (Figure 2) (Pal et al., 2007).

For our project, we attempted to replicate the simulation run by Pal et al., using a coevolution genetic algorithm. We based the representation of populations and fitness functions on those detailed by that paper. Unlike most GAs, we did not use crossover to introduce variability, because reproduction done in our model system is asexual, meaning there is only one parent for each child, and therefore no opportunity to undergo crossover. Besides complete replication, we also made some alterations to increase the similarity of our simulation to how the bacteria-virus system operates in nature.

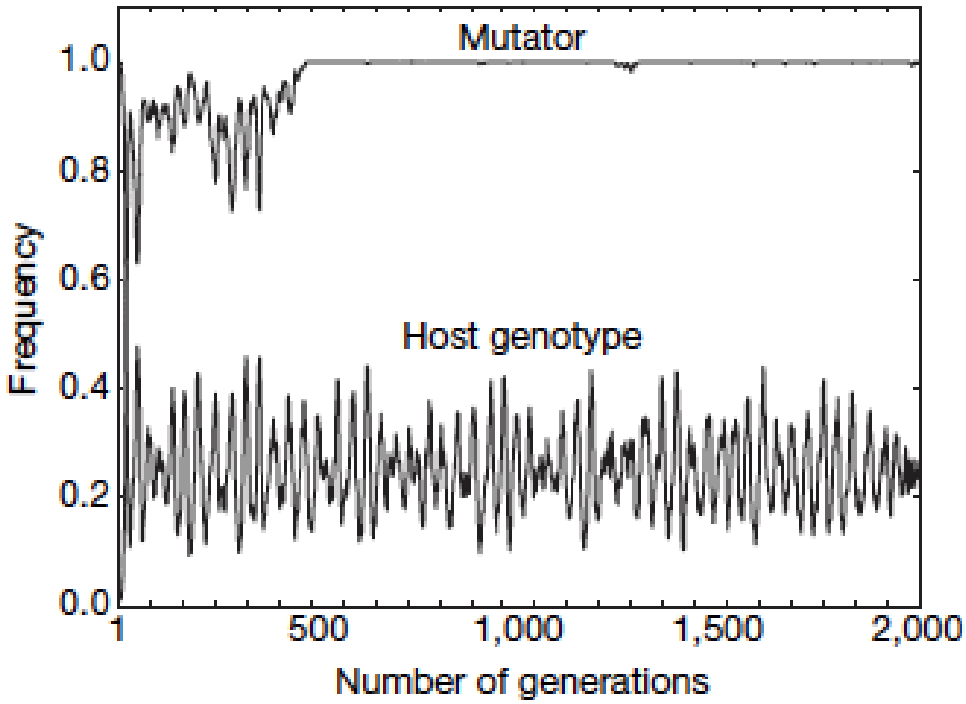

Figure 2: Frequencies over time of a mutator gene that increases mutation rate 100-fold in the host population and one particular host genotype (Pal et al. 2007).

2 Coevolutionary Genetic Algorithm

Viruses and hosts are divided into two populations, with individual genomes being represented by ArrayLists of bit arrays. Both types of genomes have indices for interaction models and mutators, while viruses also have a set of virulence genes and bacteria have a set of resistance genes and one of viability genes (Table 1). The interaction model is either the gene-for-gene model, where the resistance/virulence bits represent genes, or the matching-allele model, where they represent a recognition sequence. In the former, viruses are able to infect if they have a virulence gene for which the bacteria doesn't have the corresponding resistance gene. In the latter, the virus can infect if its virulence sequence exactly matches the bacterial sequence. The mutator in both is switched off by default, but can mutate to increase to mutation rate of carriers by 100-fold. In bacteria, the viability genes serve as a trade-off for increased mutation rate. They can mutate to be deleterious, decreasing fitness, and an increased mutation rate makes that change more likely to occur. Any aspect of each genome can undergo mutation at a rate constant for both populations, except the individuals with mutators switched on.

Individual	Interaction Model 1 bit	Mutator 1 bit	Resistance/Virulence n bits	Viability m bits
Host	yes	yes	yes	yes
Virus	yes	no	yes	yes

Table 1: Representation of Virus and Host Individuals. Both are ArrayLists of Arrays, and individuals are associated with unique ID codes.

Fitness for each virus is a function of the number of virulence genes in its genome, multiplied by the number of hosts in the current population it is capable of infecting:

$$W_{Pij} = (1 - a \cdot p)^v \cdot N, \tag{1}$$

where v is the number of virulence genes, and N is the number of potential hosts. There is also a cost of virulence, mimicking a trade-off with prioritizing virulence over other genes, represented by p. a is the bit representing the interaction model, which is 1 for the gene-for-gene model. For the matching-allele model, counting virulence and resistance genes is irrelevant, so that term equals zero.

The host's equation is more complicated, as it takes into account resistance, viability genes, and the fitness of any viruses infecting it. Fitness for host i with n deleterious mutations infected by virus j is:

$$W_H(i, n) = (1 - s_v)^n (1 - a \cdot c)^z (1 - s \cdot W_{Pij}), \tag{2}$$

where s_v is the effect of each deleterious mutation, c is the cost of each resistance allele, z is the number of resistance alleles, s is maximum virulence of the virus, and W_{Pij} is the fitness of virus j on host i. Total fitness for each host is $\sum W_H(i, n)$ for all viruses able to infect it.

These fitness functions work with the interaction models to operate as a version of fitness-proportional selection (Figure 3). Viruses interact with all of the bacterial population, and fitnesses are evaluated before reproduction. Both viruses and bacteria undergo asexual reproduction, and have the potential to make many children. How fit an individual is determines what proportion of its maximum potential children actually get created. These clonal children undergo mutation at the rate dictated by their genome and the systemic mutation rate. Crossover does not occur, as each child has only one parent. Individuals die after reproducing and are removed from the population. The populations are allowed to vary, within certain limits analogous to a carrying capacity. If a population overshoots the limit, individuals are culled in a fitness-proportional manner, with the least fit individuals being more likely to be removed. Culling fitness is determined by an objective measure, a function of the number of viability genes and deleterious alleles. Final overall fitness was determined by how many clones of each genome existed in the final population.

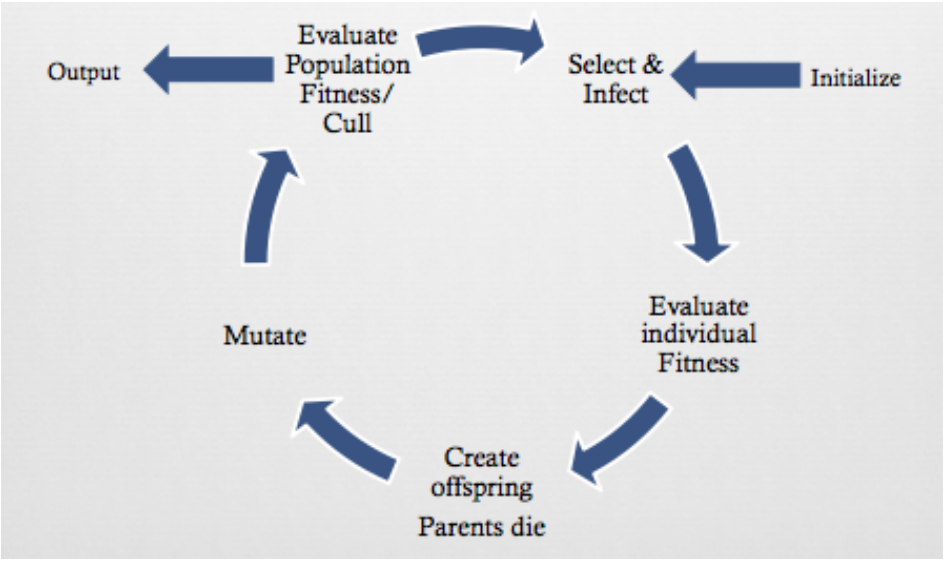

Figure 3: Flow chart diagramming what happens in each generation of the algorithm.

3 Results of Simulation

Due to debugging issues, we ran out of time to do most of the experiments we wanted to. As is clearly shown in in Figure 3, we were unable to replicate the results of Pal et al. (2007), and didn't feel it was worth carrying on to doing other experiments without having the baseline comparison. Any results we got would be meaningless without the assurance that our program was mimicking virus-host interactions in a similar way to theirs. We worked out that the oscillations were because the population was settling on exactly one genome for every individual, even though they were uniquely created (attached id numbers were unique). We decided that probably had to do with how we were using fitness to determine the number of children, so we tweaked those equations. Originally, we used a form of objective fitness, and believe that that resulted in the populations each doing their own thing rather than coevolving. After reworking the fitness functions to incorporate those listed in the Algorithm section above more explicitly, we ended up with behaviour as demonstrated in Figure 4. The selection was so strong that individuals weren't replacing themselves, and populations would crash to zero individuals. Figure 4 is from one of our attempts to diagnose the problem by giving the bacteria constant numbers of children, which at least prevented the population crash. It did, however, bring back the problem of one dominating genome, since it removed selective pressure. The culling method used to cap the populations utilized objective fitness (based on viability genes) was the only thing controlling what individuals were in the population, and led to really strong selection for the most objectively fit individual. We did experiment with various parameters to see if they would change things. Increasing the population sizes and base mutation rates were the only alterations that noticeably altered the pattern. Both increased the number of generations it took for the bacteria population to settle to the oscillations between single genomes.

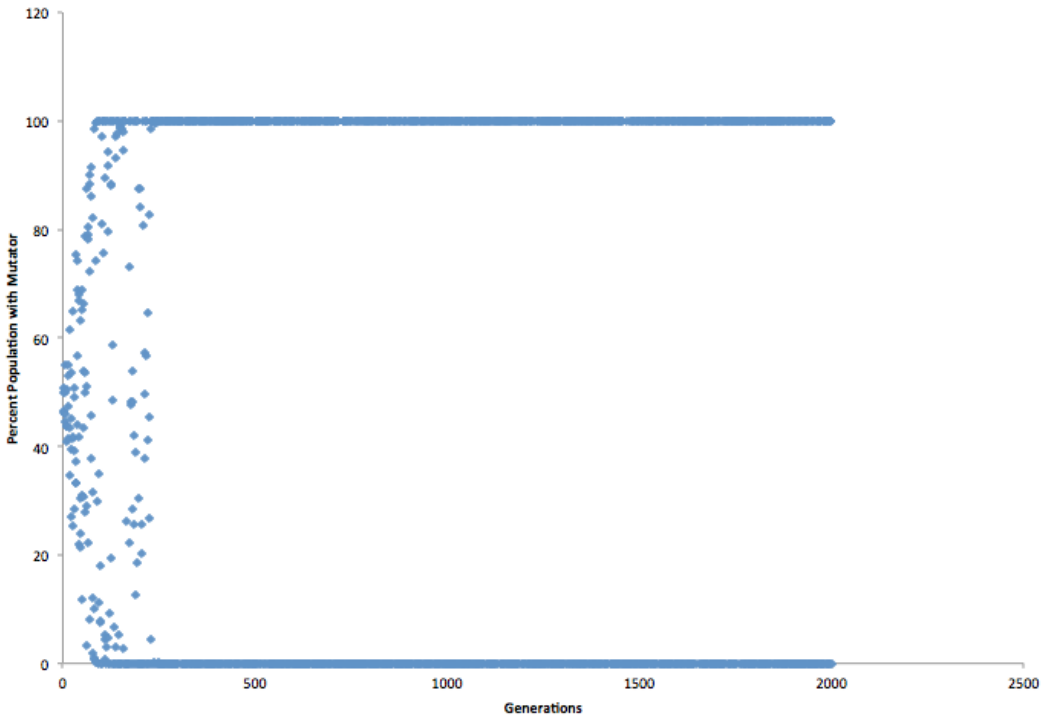

Figure 4: Frequencies over time of a mutator gene that increases mutation rate 100-fold in the host population. It eventually settled in a random oscillation between 100% and zero. Host population was capped at 5000, while virus max was 500.

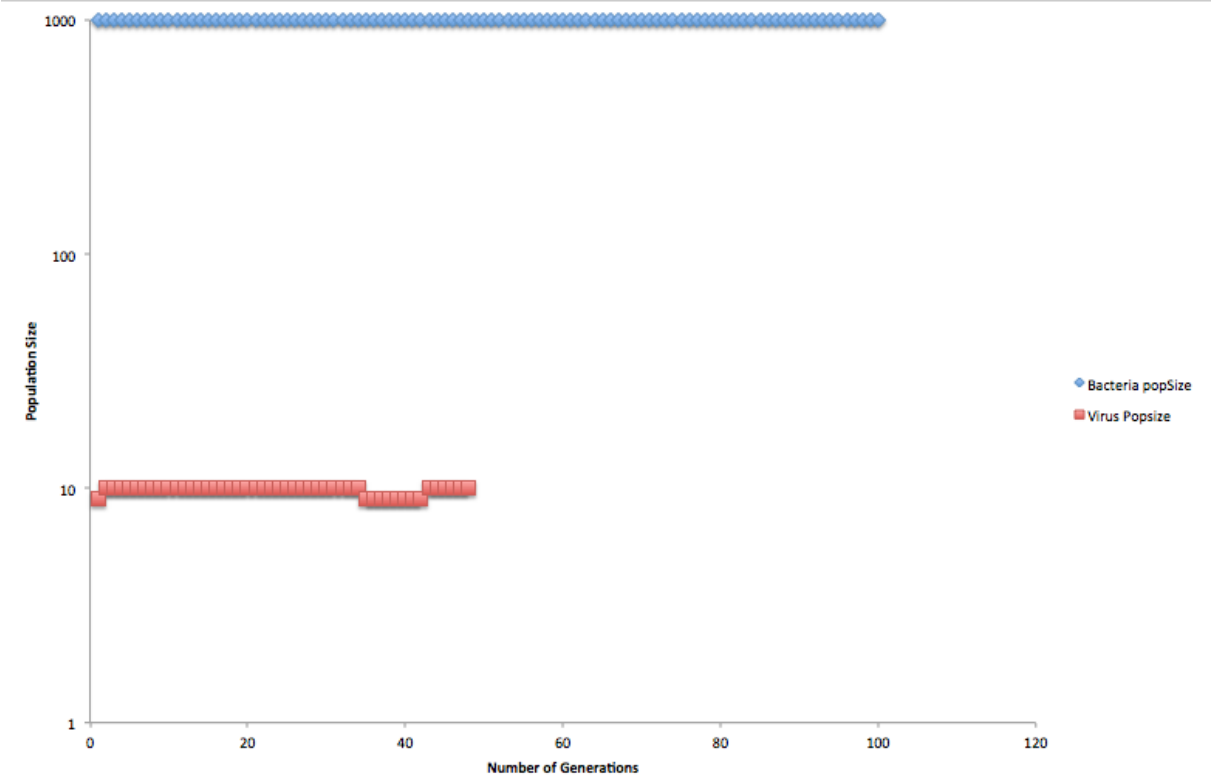

Figure 5: Sizes of bacteria and virus populations over 100 generations. Virus population crashed to zero in generation 49. Host population was capped at 1000, while virus max was 10.

4 Discussion

Because we didn't get any positive results, there isn't much true research to discuss. The biggest problems we had were with interpreting the information in the paper in order to actually code things. Because it was written as biology research rather than for computer science, the authors were not terribly specific about what precisely their algorithm. The problems we were struggling with at the end were mostly because they didn't tell us how the fitness functions translated to reproduction. We were therefore left to our own inspiration to come up with those sections of the algorithm, and due to our relative lack of experience with these types of of algorithms we probably were slower to implement new attempts as well as less quick to recognize dead-ends than would be ideal. Our

Given more time, there are many things we would like to do once the fitness functions have been sorted. For fixing those, the next thing on our list to try was adapting the fitness proportion selection method from a previous homework assignment to choose children for the populations. We decided pretty early on that it would be useful to be able to track phylogenies as evolution took place, and spent quite a lot of time trying to implement a tree that would allow us to do that. We were unsuccessful at that as well, and ended up with a less powerful implementation using hashtables and tracking clones rather than all children. Getting the full version we envisioned running would be a priority. We would also like to tweak the parameters to determine the optimal settings for our simulation. With the semi-broken version from Figure 3, we can infer that it takes about 400 generations to stabilize. However, that will not necessarily be true in the working version. Additionally, there are trade-off values that introduce costs for being very resistant, or

very virulent. It would be very interesting to find what values of those best support coevolution, and how much it would take to bias the outcome toward one population or the other.

Our results overall were disappointing, mostly because there weren't any worth anything. We got so mired in debugging and wrestling with code that we didn't get to spend any time working with the actual simulation, which is what we were excited about in the first place. In that sense, we were not successful at all. However, we did learn a lot about dealing with fitness functions in particular, and that having only one parent doesn't necessarily make things simpler.

5 Works Cited

Agrawal, A. and C.M. Lively. 2002. Infection genetics: gene-for-gene versus matching-alleles models and all points in between. Evolutionary Ecology Research. 4: 79-90.

Pal, C., M.D. Marcia, A. Oliver, I. Schachar, and A. Buckling. 2007. Coevolution with viruses drives the evolution of bacterial mutation rates. Nature Letters. 450: 1079-1081.

Person, C. 1959. Gene-for-gene relationships in host:parasite systems. Canadian Journal of Botany. 37: 1101- 1130.

Todar, K. 2012. Bacteriophage. Online Textbook of Bacteriology. Accessed 6.4.2015. http://textbookofbacteriology.net/phage.html