Algorithm for the Simulation of Gene Regulatory Networks

Report on graduate thesis by Antonios Kioukis

1.1.1 Preface:

This program implements and extends the algorithm presented on the paper titled: “Does evolutionary plasticity evolve?” by Dr. Andreas Wagner. The program was written in C language, due to the large volume of data needed to be processed a higher level language would have had an immense overhead. Furthermore,it encompasses some new features not included in that paper. Instead of creating random gene interactions inside the **Gene Regulatory Network** (**GRN**), we have implemented the regulating regions of each gene (R1, R2) , and through those 2 regions, we create the gene interactions inside the GRN. In addition, individuals who reach maturity periodically, are not discarded like the base paper, but handled as special cases called: cyclic equilibriums. Finally, in addition to the 2 models of recombination which are implemented, there are options for neutral or not selection.

1.1.2 Theory:

A GRN is a collection of regulators that interact with each other and other parts of the cell. Through this interaction different expression levels of mRNA and proteins are created. These expression levels play a role in the creation of body structures which in turn are vital in evolutionary developmental biology.

In a GRN, the genes can be viewed as nodes in a network. The input is transcription factors and outputs being the level of gene expression. So, each GRN can be viewed as a programming library which is part of a much larger whole. As a result each node has double effect.

Firstly, as a global feature these networks create a hierarchical scale free network. A scale free network means that there are a few highly connected nodes and many poorly connected nodes. Another important characteristic of a GRN lays in the network motifs. Inside each GRN is an abundance of repetitive topological patters. In order to see those patterns we have to look at the GRN after dividing it into small blocks.

Studying GRNs is important because through those we can draw conclusions about the phenotypic plasticity. Phenotypic plasticity is the ability of an organism to change its phenotype in response to changes in the environment. Generally phenotypic plasticity is more important for immobile organisms than mobile organisms because mobile organism can migrate in order to find more favorable conditions.

Working against the effects of phenotypic plasticity is canalisation. Canalisation measures the ability of a population to produce the same phenotype regardless of variability of its environment or genotype. A canalisation - phenotypic plasticity metaphor says that: phenotypes are very robust to small perturbations,for which development does not exit the canal, and rapidly returns down. With little effect on the final outcome. Perturbations whose magnitude exceeds a certain threshold will break the canal and follow an uncharted path.

As a result hose two forces cannot be quantified easily, and any result is not easily attributed to one, and we simulate GRNs to get a view of at least the phenotypic plasticity in order to comprehend those forces of nature.

1.1.3 Background:

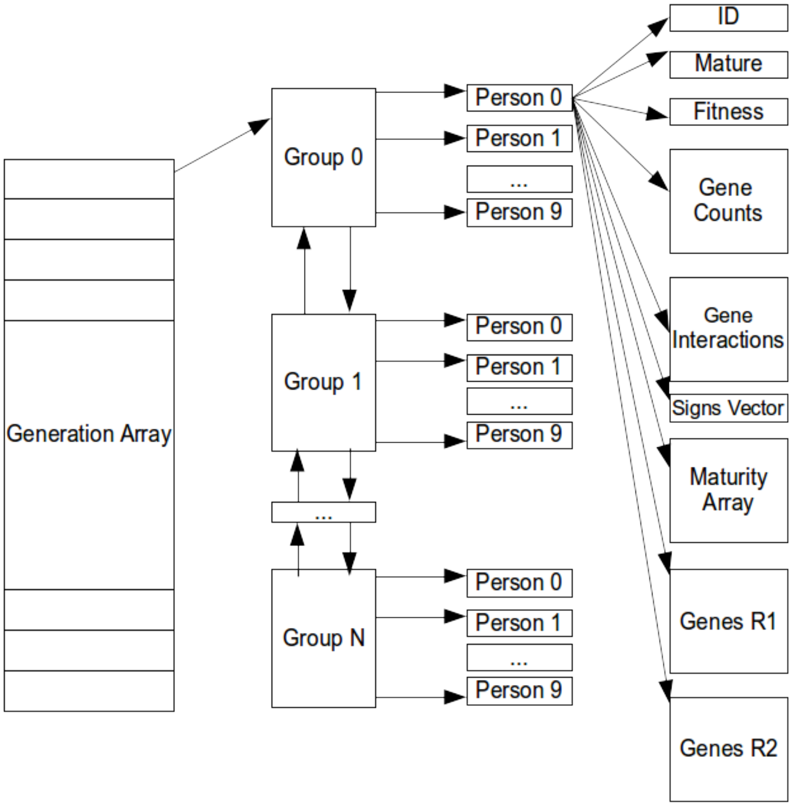
In 1996, Dr. Andreas Wagner publishes his paper: Does evolutionary plasticity evolve? From the early stages of development in each organism, a large number of interactions take place. Those interactions occur at the lowest levels as mutations. Those changes do not affect the organism immediately but instead are buffered. This phenomenon is call “epigenetic stability”. Variation in epigenetic stability can occur though. The research continues by modeling the transcriptional regulators that mutually regulate each other’s expression. Due to constraints in computer resources, the researcher was forced to experiment with a low number of vertexes in the network and find solutions to modeling problems. Through the advances of technology, we were capable of remodeling that research and adding new features.

1.2.1 Initialization of Population:

Each generation has N number of individuals. For acceleration purposes, those N individuals are then grouped into groups of 10. The groups are part of a doubly-linked list to accommodate changes in population sizes from one generation to the next, in case of an event. Each individual holds different kind of records.

Finally, each generation is stored into an array of pointers, called Generation_Array, each entry points to the population of a generation.

Using the above schema we are capable of finding the information for any individual with 2 hops, 1 from the generation to the group and from the group to our goal. For clarity purposes, check adjacent figure.



1.2.2 Initialization of each Individual:

Each member of the population, is represented with a structure holding many different elements. ID is the personal number of that member inside his group. The mature field is a boolean variable, indicating whether of not the member has reached maturity or not. Fitness is a float number in the range of 0 and 1. The closer fitness is to 1 the more chances the individual has to leave progeny. Gene counts is an array of integers, with each cell representing the expression of the corresponding gene. Its values are randomly selected from the range of specified by the user at the beginning of each run. The Maturity array has room for 1024 integers and at this stage it is empty. It is used in the maturation process of the individual and in calculating the personal period of maturity. Genes R1 and Genes R2 are 2 arrays of 10 integers each which are filled with random positive integers from the range given by the user.

Lastly, is the creation of the Gene Interactions Array. Each cell of this square matrix, stores the result of the following operation: From every integer M in R1, take the last bit. Do the same for every integer L in R2. Compare those 2 bits. If the produced pair is 11 then we have a positive interaction. If the pair is 00 we have a negative interaction. The other 2 pairs (01.10) indicate no regulation. The strength of those interactions are calculated by the number of set bits in the respective R1,R2. Store those interactions at the Gene Interactions Array at the M,L cell. We continue the same operation until the array is full.

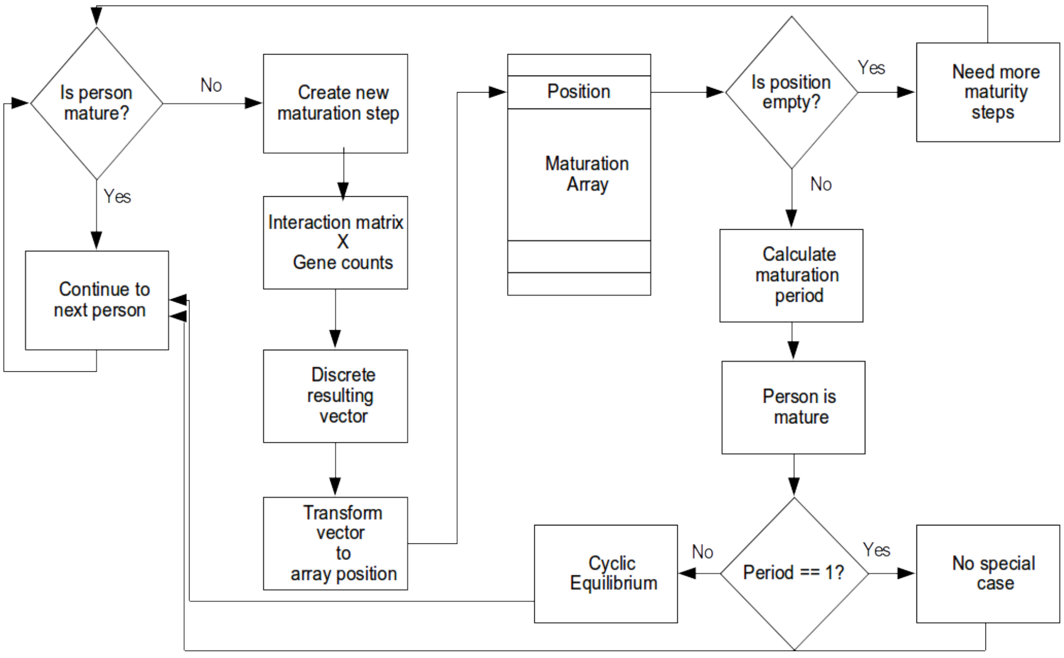
1.3 Maturation:

After creating the population, we proceed to the stage of maturation of the population. We create maturity steps for the whole population,by multiplying the gene_interactions matrix and the vector_of_signs (initialized to 1). The result is then made discrete, if its positive it gets 1, if its zero or negative it gets 0.We use those bits to create the new vector of signs and through concatenation of those bits we create an integer. This integer is used to indicate the position in the maturity array, in which we place the number of the maturity step.

If the position is empty, it indicates that this particular version of the vector of signs is for the first time encountered and we need more maturity steps. Before creating another maturity step, the current maturity step number is saved in the maturity array. If the position in the maturity array is not empty that means that the individual is now mature and its maturity period is the number of the current step – the number of the step present at the cell. Since personal maturity has been achieved, the individual is excluded from further maturity steps. This process continues until all the population is mature.

For the majority of the population, the maturity period is 1, but for a small fraction of the population this is not the case. In those cases, a cyclic equilibrium is encountered. It doesn’t affect the maturation process but it affects the fitness calculation process following.

Maturation Process



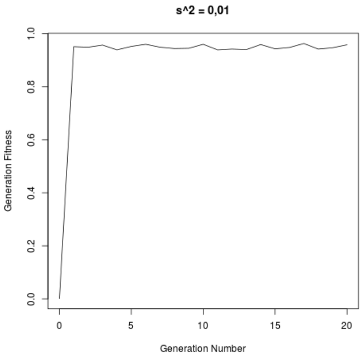
1.4 Fitness Calculation:

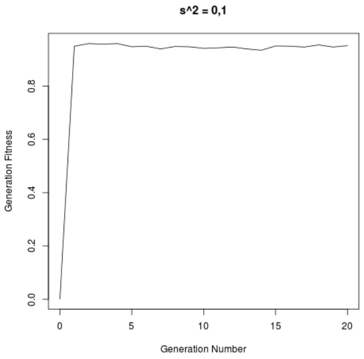
Each mature person now has a discrete vector consisting of 0s and 1s. For the individuals with a personal maturation period of 1, we proceed by calculating the euclidean distance between the personal signs vector and the optimal vector. For those who present a cyclic maturity equilibrium a different approach is used. If a person has an M maturation period, we create M maturation steps. Those steps create M vectors of signs. For every vector, we calculate the euclidean distance. The personal distance of the person is the mean of those distances. The personal fitness is then calculated with this formula: fitness=$e^{(-l*distance)}$.

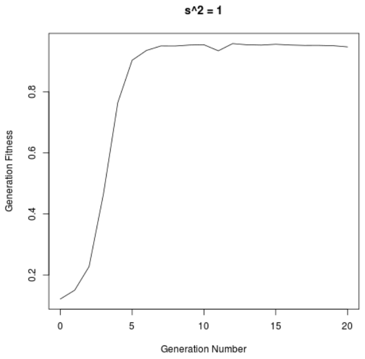
The lambda is indirectly given by the user since $l=1/s^2$ and s^2 is user-defined. The smaller the s^2 the bigger the increase in fitness from one generation to the next. This happens because a small s^2 makes any differences between the optimal vector and the personal vector of signs matter more. As a result, an individual with 1 difference dominates an individual with 2 differences. A big s^2 makes the simulation run without large leaps in fitness, because it is more tolerable to differences between the vectors.

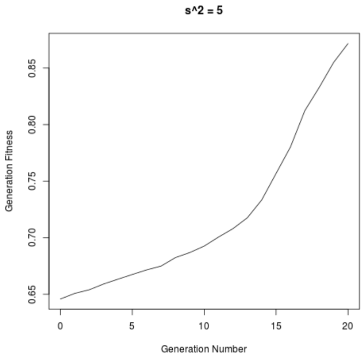
After calculating the fitness of the whole population, the program divides the sum of the individual fitness with the number of individual to get a normalized percentage. The closer to 1 it gets the more adapted to its environment the population is.

The following graphs display the leaps of fitness while demonstrating the effects of different s^2 values.









1.5 Implementation of Forwarding:

Once a population is mature, it starts to create the next generation. There are 3 factors guiding this phase. The number of parents, the fitness of the parents, and which model dictates how the recombinations of the genes of the parents work.

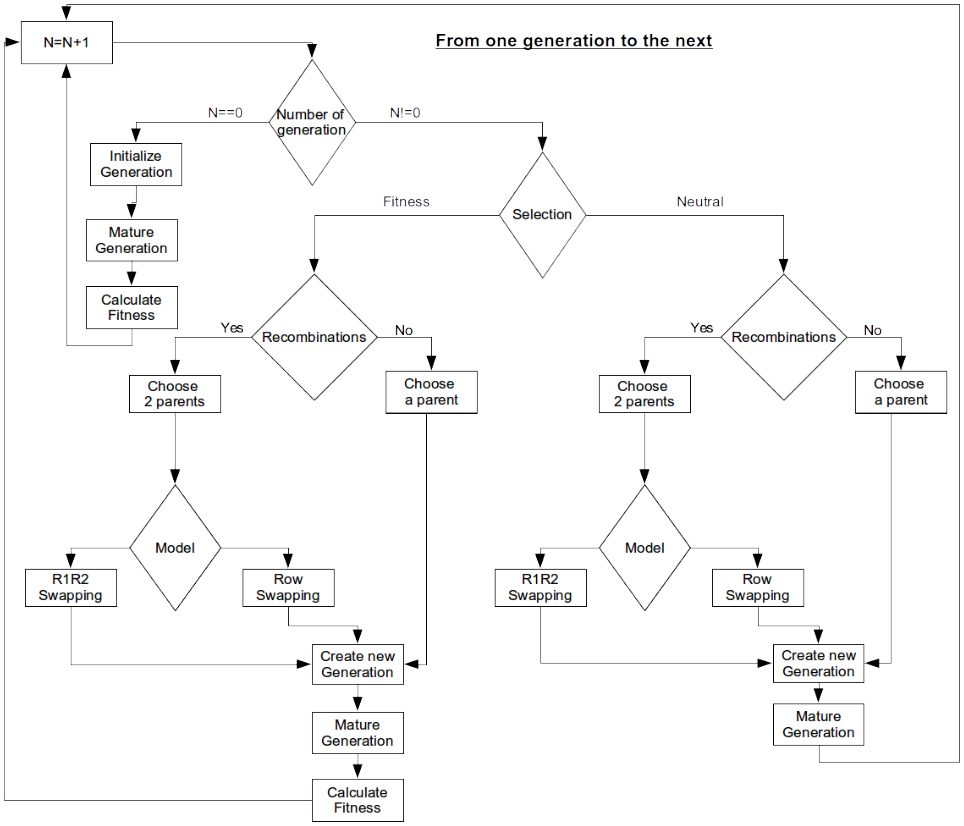
If the number of parents is 1, there are no combinations and the gene interaction matrix passes from parent to child. If a pair is needed to create a new individual then there are combinations of the parents which follow 2 models:

If the model of Dr. Wagner is followed then a random number C is selected from 0 to number of genes. The child inherits from the first parent C columns and from the second parent the remainder columns.

If our model is followed , 2 random numbers R, C are selected from 0 to number of genes. The top-left part of the gene interactions matrix is inherited from the first parent and the bottom-right part of the matrix is inherited by the second parent. The top-right part is filled with the interactions emerging from R1 of the first parent and R2 of the second parent, while the bottom left part is emerging from the R2 of the first parent in conjunction with the R1 of the second parent. Detailed explanation of these interactions at 1.2.1 paragraph.

A big factor for the creation of the next generation is if the fitness of the potential parent plays a role in his selection. If it doesn't, the population's fitness remains almost unchanged no matter how many generations we simulate. If the fitness of the parent plays a role, then a more fitted father has a much stronger possibility of passing his genes to the next generation. So as the simulation progresses we see that the fitness of the population is nearly 1. It can never reach it, due to unfavorable mutations and the combinations but it hovers close enough.

The following diagram displays a typical run of the simulation:



1.6 Mutations:

After the creation the new generation and before its maturity some mutations represent themselves. Those mutations affect the inherited R1,R2 arrays of random members of the population. The number of mutations per person is a Poisson distribution using as variable the mutation rate given by the user. Each mutation affects an cell in the R1 array or the R2 array. The selected for mutation cell contains an integer which is then made binary. A random selection is subsequently made for the bit that will be mutated. The mutation is represented by changing the bit to 1 if it was originally 0 and 1 if it was 0. The last bit of the binary integer represents the interaction of this gene with the GRN. Due to that fact, the chance for the last bit to change is 1%. The remaining 99% is distributed equal to the the other bits. The resulting mutated integer is then placed back at its original place.

1.7 Events:

There are 2 types of events: increase of population and a decrease in population. The decrease of population is implemented without any difficulty by removing the groups needed in the end of the doubly-linked list. Due to the randomness of the individual in each group, the general fitness of the population, if calculated, does not change.

The increase of the population is handled by creating new groups of individuals, by the last generation, and adding those groups at the tail of the list. The next step is to mature those new groups. Then if fitness-based selection is active then the fitness of the population improves and we need to recalculate this fitness when all the new individuals have been created.

1.8 User's guide:

The program at the end of the run outputs data to text files. Each text file contains different kind of data from the simulation and for readability and consistency purposes follows a simple prototype. At the beginning of each new generation's data, the number of the generation, the number of individuals and the number of genes are reported. After that, follows the personal data of each member of the population. It's worth noting that, when the fitness of an individual doesn't affect the possibility of leaving offspring, the fitness text file is empty. Also, take in account that both R1R2 and counts text files do not differentiate between different individuals of the same generation, but since the number of genes is known we can easily find out the data belonging to each one.

1.9 Limitations:

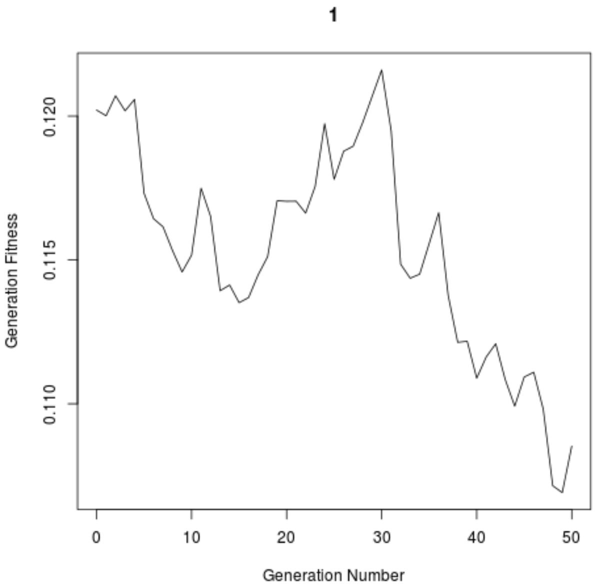
There is a hard limit to the number (10) of genes each person can have. This was selected because for more genes, the personal maturity array had to have a size of 2^N of genes. As a consequence of the memory requirements, a limit had to chosen. That limit is 10 genes per person. Otherwise there are no limitations as to how big a sample we want to simulate. The use of the double-hop system ensures an acceptable time complexity.

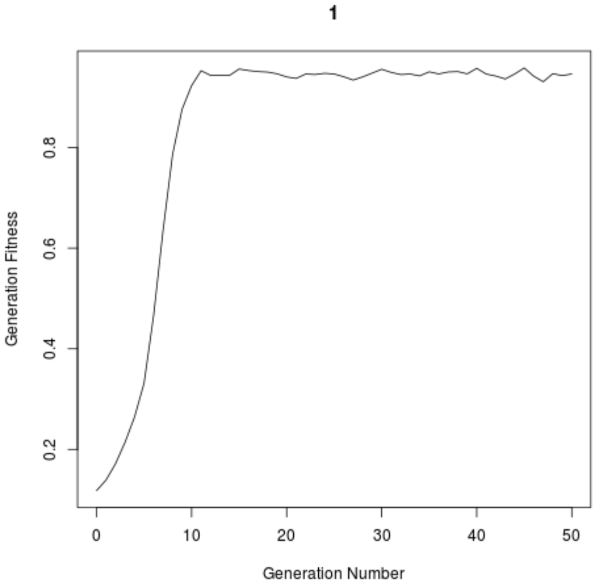
The simulation while running never deletes any data. This is for testing purposes and does not result in memory leaks. But there is the side effect that each run needs large portions of RAM. As an example a simulation for 1.000 individuals and 5.000 generation requires 21 Gigabytes of memory space. The bulk of the memory is held up in the personal records so a simulation of 10.000 individuals for 1.000+ simulations although supported but not advised.

(continue for results)

2.1.1 Comparison between neutral and fitness-based simulations:

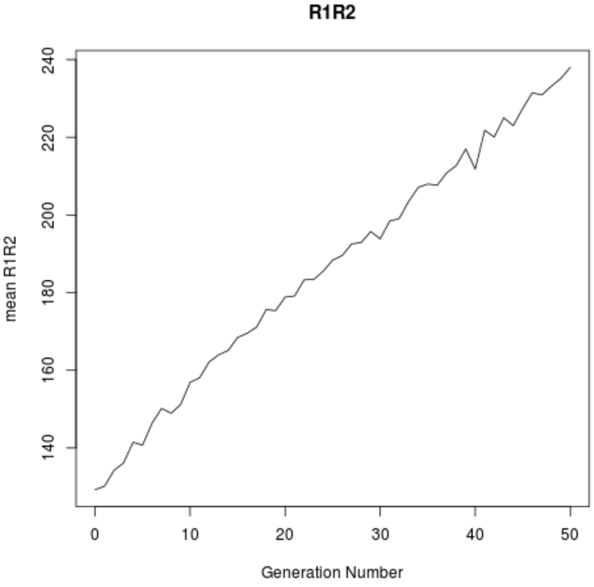
First and foremost, on fitness-based models the population reaches a level of fitness of 0.90 and then it plateaus. The model cannot simulate a generation fitness of 1, due to the existence of random mutations. As expected, simulations run on neutral evolution do not have this trait, instead the general fitness of the population hovers on a level set by the initial random start. The next graphs depict this effect more clearly.

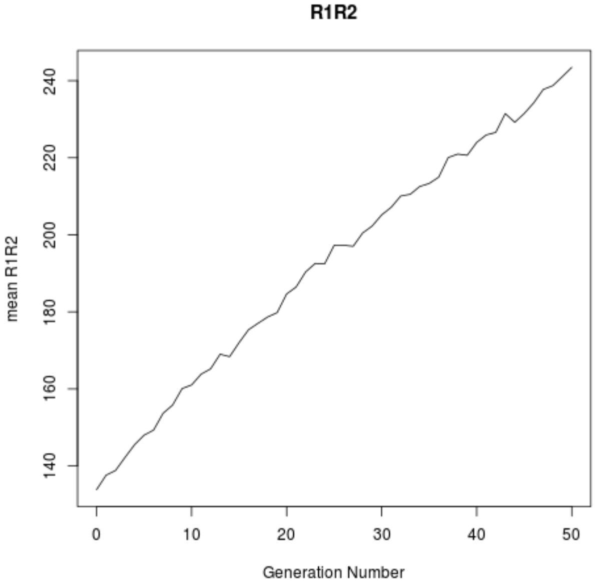




Secondly, the size of the population plays a vital role. Due to the random initialization process followed in all simulations, when the sample from where to draw potential parents decreases, the upwards leaps in generation fitness also decrease. As an example a sample of 1000 individuals reaches 0.9 generation fitness in 1 generation where as a sample of 100 individuals reaches that same percentage after 5 forward generations.

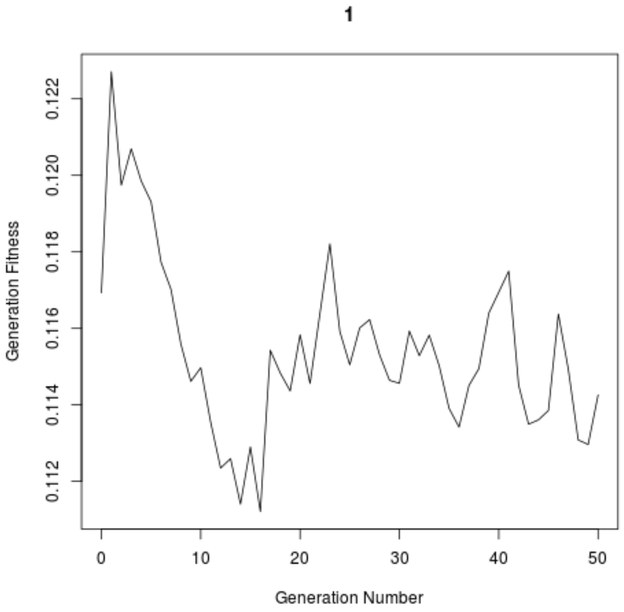
An interesting find follows. Despite using neutral evolution the R1 and R2 regions follows the same pattern with R1,R2 regions following the fitness-based model. The pattern shows that with each new generation those 2 regions show an increase. Those graphs illustrate the canalization of the interaction inside a GRN reproducing Dr. Wagner's work.

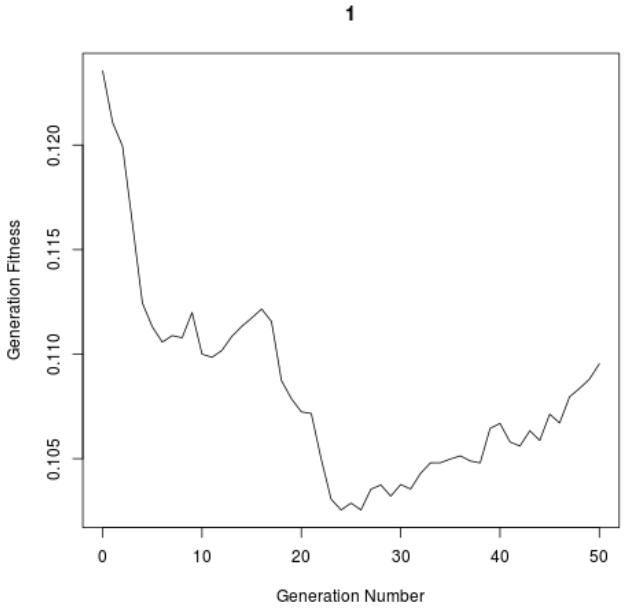




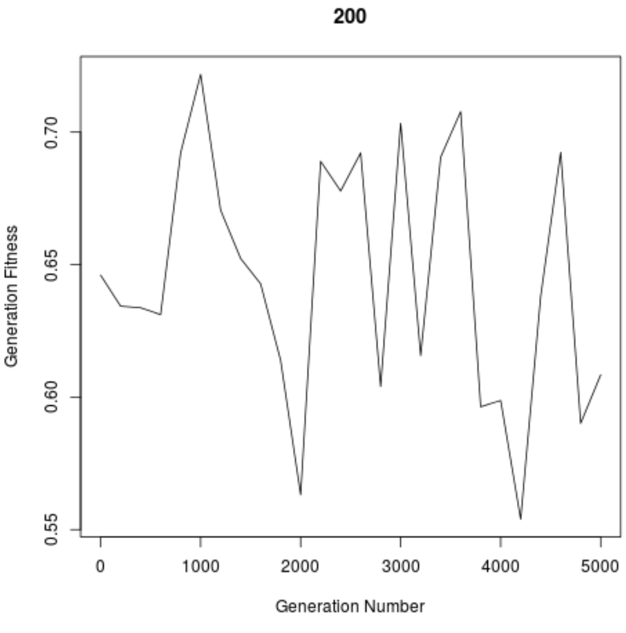
2.1.2 Comparison between sexual and asexual reproduction simulations

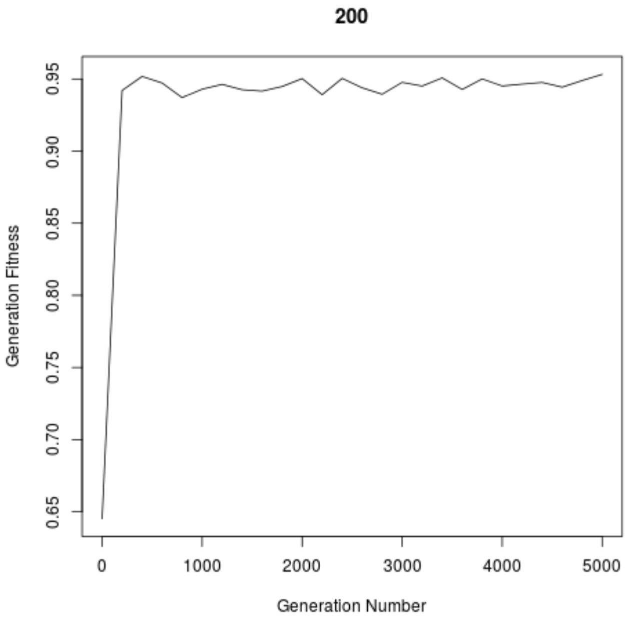
In accordance to Dr. Wagner's research, our simulations also show that asexual reproduction wield the same results when there is neutral evolution. In the simulation the asexual reproductions are represented by having no recombinations when creating the next generation. The graphs were generated with a sample size of 1000 individuals, a mutation rate of 1 and following neutral evolution. The graphs show that despite the difference in the reproduction model the general fitness of the generation continues to stay at the same level despite fluctuating "wildly" between the generations.



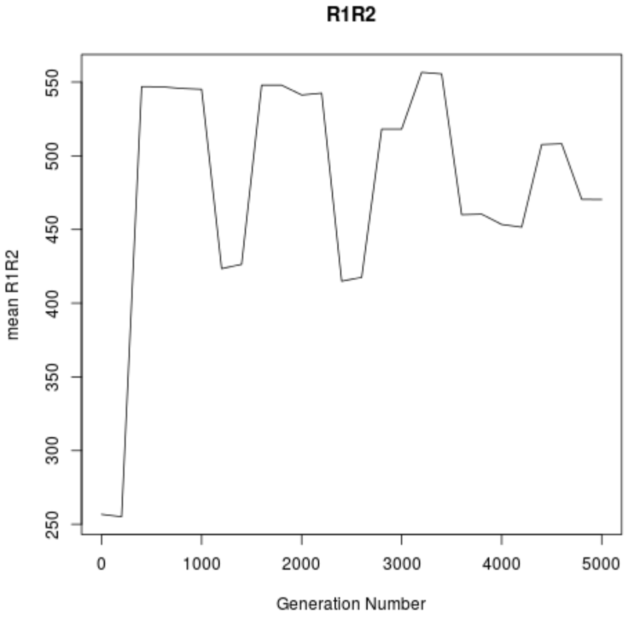


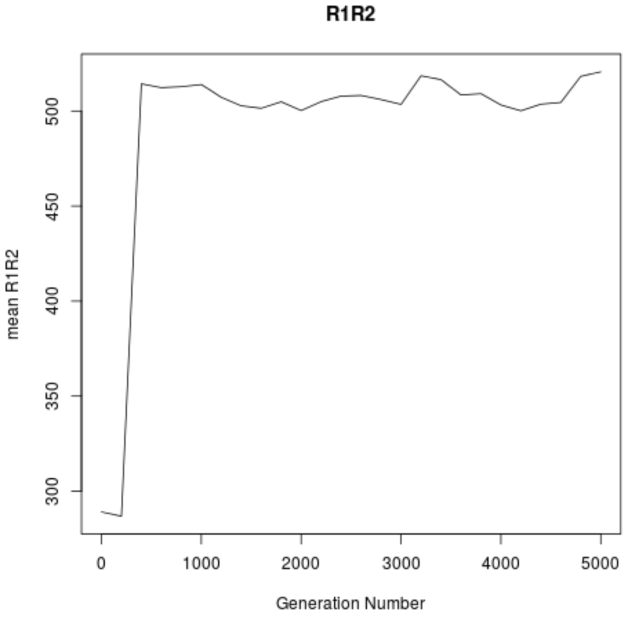
In contrast, when the fitness of the individual plays the dominant role in the selection process, whether there is sexual reproduction or not differentiates the results. This happens because when 2 fitted individuals are needed for reproduction, a shortcoming in one factor from the first parent can be covered from the second parent. If there is only 1 needed parent then those shortcomings may endure, and appear further in the forwarding simulations. The following figures where created with a sample size of 1000, mutation rate 5 and created 5000 forward generations.





Finally a result which may have a little importance but its worth mentioning are the minor differences in the R1,R2 regions when the different methods of reproduction are used. When a pair is not required, there are sudden dips in the mean values of the regions. Furthermore there is a correlation between how small the s^2 is, and how steep are those dips, but the reason behind this is explained in paragraph 1.4 so we will not go into more detail here.





3.1 Possible Extensions:

An extension of the current algorithm would be to lift the hard limit on the number of genes per individual. But, this limit is enough to draw our conclusions and I believe that those results are indicating the behavior for GRNs with more nodes.

Through the schema of the program its very easy to substitute the fitness function with another. Currently the fitness of an individual is calculated by the euclidean distance. This was chosen because of the easy implementation in C and more importantly, the results where intuitive without the use of complex statistics. Both Chebyshev and Manhattan distance are valid candidates

References:

Wagner, A. (1996). Does evolutionary plasticity evolve?. *Evolution*, 1008-1023.

Canalisation (genetics). *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/Canalisation_(genetics)

Phenotypic plasticity. *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/Phenotypic_plasticity

Gene regulatory network. *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/Gene_regulatory_network

Scale-free network. *Wikipedia* from https://en.wikipedia.org/wiki/Scale-free_network