Evolutionary L-Systems and the Growth of a Forest

JACK TAYLOR, Queen's University, Canada
MANAS SRINIVASAIAH, Queen's University, Canada
THEO RAPTIS, Queen's University, Canada

ACM Reference Format:
Jack Taylor, Manas Srinivasaiah, and Theo Raptis. 2023. Evolutionary L-Systems and the Growth of a Forest . In . ACM, New York, NY, USA, 22 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 ABSTRACT

The success of forest ecosystems relies on the cooperative relationships between species and their functional roles within a larger system. Trees compete for resources such as sunlight, water, and nutrients, which drive adaptation and niche-filling. This competition leads to adaptation where certain trees fill certain niches. The study of digitally simulated plant life is a concern of both mathematics and biology where computers are used to generate plant-like structures, the properties of which are then studied. In this paper, we explore the evolutionary development of plant-like structures using an evolutionary algorithm and Unity for generating and visualizing 3D plants. The plants are defined using L-Systems, which model their genotype and phenotype. We introduce these digital plants into a simulated environment with limited resources and assess their fitness based on sunlight absorption and growth. By applying an evolutionary algorithm, we aim to gain insights into the optimal plant structure for capturing sunlight and nutrients within our simulated environment, as well as the optimal composition for an overall forest system layout. We also assess the competition between trees by analyzing their shoot systems and crown structures. Our approach combines L-Systems and evolutionary algorithms to model the development of trees within a simplified forest ecosystem, providing valuable insights into the niches and various strategies that shape the structure of real forests.

2 PROBLEM DESCRIPTION

2.1 Introduction

Within a simplified environment, the exploration of the evolutionary development of a tree's architecture - the branch structure - will be our primary focus. In an environment where there is a population of trees and a sun that arcs across the sky, the competition for sunlight will necessitate evolving strategies for tree growth, branch design, and leaf placement. We expect speciation to occur and a variety of different adaptations as a result. As the population of trees develops, each with different survival strategies, the resulting forest canopy is moved closer to the optimum structure to capture sunlight and nutrients. Trees' crowns will fill gaps in the canopy, vary their height, vary the width of their crown, to both compete and exist in symbiosis with others. This resulting canopy structure and its components will be something we analyze as well.

Manuscript submitted to ACM

2.2 Expectations

(1) To gain insight into individual optimality of the best architecture to capture solar rays with resource cost
(2) To gain insight into an optimal composition for an overall system layout for outcome item 1.

2.3 Digital Representation of Trees

The plants themselves will be defined using L-Systems, which are a parallel rewriting systems commonly used to model plant growth. This will be the focus of the genotype, and a way to model the phenotype of the emergent tree structure. L-Systems consist of an alphabet containing a set of symbols, an axiom defining the initial state of the system, and a set of production rules defining the way in which certain symbols can be replaced with combinations of other symbols in the alphabet. These rules are applied iteratively to create fractal-like structures. Each system will form the phenotype of an individual. We seek to introduce these structures into an environment with limited resources that are used for photosynthesis and survival. In addition, growth, performed in iterations, will be constrained by associating a cost with the purely structural part of a digital plant, making it increasingly costly for a plant to grow larger and larger. The combination of resource consumption and acquisition will form the fitness of the generated individuals. Applying an Evolutionary Algorithm will allow us to evaluate the fitness of these structures and evolve fitter structures through parent selection, recombination, mutation, and survivor selection. Upon the completion of our algorithm and simulation, we will attempt to classify the resulting structures and explain their creation.

3 LITERATURE REVIEW

Although L-Systems are commonly used within mathematics to construct plant-like recursive structures, few have attempted to combine this with an evolutionary algorithm. On Genetic Algorithms and Lindenmayer Systems[4] covers the implementation of a genetic algorithm and non-stochastic L-Systems to define 2D plant structures. Ochoa sought to emulate natural evolution by including factors studied in biology that are attributed to plant growth and reproductive success.

The algorithm in question uses a crossover function which is heavily inspired by the Lisp sub-trees used in the initial implementation of genetic programming. The mutation strategy was to randomly manipulate either a single symbol within a chromosome, or to mutate an entire block in the chromosome by a random (syntactically correct) string. The algorithm uses L-Systems represented in a tree structure rather than the binary fixed-length string encoding of a standard genetic algorithm. Ochoa also uses steady-state selection, replacing 1/5 of the least fit individuals in each generation. The fitness function considers five factors when evaluating the fitness of a plant, attempting to emulate natural evolution closely.

(1) Positive phototropism, where plants with a high maximum y coordinate are considered to be growing towards light and receive a higher fitness value as a result.

(2) Bilateral symmetry, which considers whether a structure is symmetrical or not. Structures that are lop-sided are given lower fitness.

(3) Light-gathering ability. To measure this, the surface area of the plant which is exposed to light and not shadowed by other structures is considered, with plants that have a large surface area exposed to light being given higher fitness.

(4) Structural stability. This factor is considered by counting the number of branching points in a plant, with those possessing too many being given lower fitness.

(5) The proportion of branching points. The number of branching points is taken in proportion to the total number of branches in a structure, with the assumption being that plants with a high number of branches are better at gathering light and dispersing seeds.

Each factor was given an associated weight within the fitness function. The algorithm was run over 100 generations with a population size of 50, a generation gap of 20% and a chromosome length of between 7-30 characters. The resulting structures were sometimes plant-like in nature, but also included structures that resembled bilateral symmetric organisms. Ochoa concluded that L-Systems were a suitable encoding for studies in artificial evolution.

Mock also made an attempt at creating 2D plant structures in Wildwood: The Evolution of L-System Plants for Virtual Environments using a genetic algorithm [3] The crossover function selected random substrings from two parents to create two children. The mutation operator selected a substring and replaced it with a random new substring. Interactive selection was initially used to select parents, with Mock choosing suitable plants based on his own visual examination of the structures produced by the algorithm. However, the algorithm was also modified to run autonomously using a fitness function that considered plant height and width. Parent organisms were chosen using fitness proportional selection. The autonomous version of Mock's algorithm tended to produce plants that were short and wide. The author also noted that the population quickly converged.

Another paper is Artificial Life Simulator using Genetic Algorithms[2]. The authors similarly chose to use a genetic algorithm approach to evolve structures using L-Systems. Fitch, Parslow, and Lundqvist chose a 'cut and splice' method for a crossover where a randomly selected section from each parent was chosen to comprise a child's chromosome, much like Mock. Mutation was applied by random reassignment. The program provided two possible methods for survivor selection; random selection and roulette wheel selection. In addition, the authors also included the ability to set a maximum age for plants and for the user of the algorithm to randomly 'inject' a brand new individual into a population in place of a pre-existing individual. The authors describe their fitness function as being a "user-defined function" but do not specify which function was used and what factors it considered. Limited results were presented, but one structure included as a figure seemed to resemble a dandelion.

An additional paper by Chen and Ross sought to use a genetic programming approach to L-System based trees along with a deep convolutional neural network (CNN) to evaluate the fitness of the resulting structures [1]. Chen and Ross used tournament selection along with subtree based crossover and mutation similar to that utilized by Ochoa. The results of Chen and Ross' work mostly touched on the use of CNN, but the authors noted the relative complexity of finding tree like structures in such a large search space. One of the primary conclusions they arrived at was that the evolutionary algorithm began to evolve individuals which more closely matched the CNN's model as opposed to those which actually resembled tree structures.

Although all four papers confirm the viability of using L-Systems within an evolutionary algorithm, none included much comment on the resulting organisms or any insight gained into the evolution of plants. Broadly, they all seek to achieve the same goal; create structures which resemble plants by attempting to apply nature-like selection pressure to individuals. Because there are so few papers on this topic, there isn't any consensus on the best way to achieve this goal.

4 EA DESIGN

Environment: The environment for this EA will be a simplified forest floor managed by a SimulationManager. Each day in this environment consists of 12-24 hours of daylight and a night period where evolution is performed on the plant population by our EvolutionManager, as summarized in Figure 3. Water separates chunks of land which creates many areas of localized competition. Plant start locations are randomized to be any area of land above water. Our environment has one resource for the plants to collect: sunlight. We use a Sun object to distribute sunlight to the plants. The Sun arcs across the area above the forest floor in set increments calculated from the total hours in the day and moves from one horizon to the other to simulate the changing angle and amount of sunlight available to plants throughout the day. Each hour 500 raycasts are sent to randomly sampled points on the forest floor, if a plant leaf intercepts this raycast it is given sunlight.

Fitness Function: There are two versions of the fitness function. 1) A simple fitness function that evaluates how much sunlight a tree was able to capture. 2) A weighted fitness function that uses the amount of sunlight collected divided by the number of leaves added to the proportion of branch cells to the total number of cells. We ran into issues using this fitness function as create an asymptotic ceiling to the fitness score, making it harder to achieve greater scores. This affected everything from parent selection, survivor selection, and children creation. It lead to more chaotic "tree" structures.

Phenotype: A tree consists of branch cells and leaf cells. For each tree, we can identify the crown type, height of the tree, number of leaves/surface area, and the total number of cells.

Genotype: The Shoot L-System (branches and leaves), specify rules for growing the plant above ground for sunlight absorption. This L-system can further be broken down into three parts, the axiom, the rules, and the parameters that modify the application of the rules to the axiom.

Survivor Selection: Elitism: We rank and choose the fittest trees of last generation to continue to the next generation. Ageism: We eliminate the oldest 20% of the trees before elitism to simulate the death of old trees/forest fires/natural death.

L-system: A tree's L-system is composed of its *axiom*, *rules*, *stepsize*, and *angle*.

- ***Axiom*** → Refers to what string to replace X with, which is an L-Systems starting state. We randomly select one string from a dictionary of strings of common start axioms from other papers that implement 3D L-Systems. A selection of the start Axioms we use can be seen in Figure 1.
- ***Rules*** → The rules define what to replace different symbols in the L-System string with at each recursive step. Most rules apart from X and F simply replace the symbol with the same symbol. F defines a single branch in the phenotypic expression and has a similar set of possible rules to select from as X apart from being slightly shorter. Giving both X and F many rules to pick from yields a great variety of initial plant configurations as as a starting point for evolution. A selection of the F rules we use can be seen in Figure 2.
- ***Stepsize*** → How long a branch or leaf is, this affects the branch cost calculated in the *grow* step.
- ***Angle*** → The angle that a branch or leaf is offset by at each branching point.

Parent selection: Parent selection is performed with roulette wheel selection and tournament selection with tournament size 5.

Children selection: A child is either created via asexual or sexual reproduction at a rate determined by the variable $crossoverRate$. In the case of asexual reproduction a child simply receives a copy of either parents genome. In the case of sexual reproduction, a single child is created by performing crossover on the genome of two parent trees. There is a 50% chance that the step size and angle of the child's genome are selected from either parent 1 or parent 2. The rules of the child are then determined via a rule crossover function, we have implemented a one point symmetrical, n point symmetrical, and n point crossover function for this purpose. Symmetrical crossover yields more plantlike children. The n-point crossover runs into the issue of creating reed like plant structures Figure 12. This is because, after n-point crossover we need to make sure [and] are not out of sync and thus we add in a push or pop to re-balance the resulting genome. This creates more branches than necessary. A further iteration of this project should look into a Genetic Programming approach to n-point crossover that treats each rule in the L-System as trees and sub-trees to properly preserve tree structure.

Growth of a tree: A tree has attributes that keep track of the total sunlight collected in a day, the current sunlight available for growth, and the total water which is given to each tree equally at a constant rate. We use an attribute to measure how many times a tree has grown so far: $interationCount$. The cost of a a growth iteration is based on factors such as the $iterationCount$, $consecutiveBranchCost$, which is calculated as a exponential weight to consecutive branch symbols, and a basic satisfaction of water collection. The equation is as follows:

(1) The sunlight collected is greater than or equal to the $branchCost$ multiplied by the $iterationCount$ plus the $consecutiveBranchCost$ and
(2) the water collected is greater than or equal to the $branchCost$ multiplied by the $iterationCount$ multiplied by 0.5.

Mutation: Mutation mechanisms take place for both the rules and parameters of our L-Systems as follows:

(1) We mutate the angle parameter that defines the angle between branches randomly with a value between -10 and 10 degrees.
(2) We mutate the step size which decides the length of the branches randomly with a value between -0.5 and 0.5 meters.
(3) We add or remove a character from the rules for the symbols [,], +, -, /, *
(4) We randomly mutate a single symbol within the rules for F or X and mirror the single point mutation across the rules center point to maintain symmetry.

5 EA RESULTS

We can report that there is good variety in initial randomization of plant genomes, with some growing tall and skinny crowns, while others grow wide and bushy crowns as seen in Figure 4. Depending on the simulation parameters (length of simulation, length of hours, mutation rate, etc) you can influence the type of forest that is created. We noticed that in many simulation speciation started to occur where certain plants filled in niches; tall trees that captured the most sunlight, shrubs that capture whatever was left over, and medium sized trees that were co-dominant with one another. Wherever the medium trees grew, the other two could not thrive Figure 5. Examples can be seen with in the 40 day simulation Figure 6 and in the the simulation of elitism Figure 7.

Our environment has islands where plant are spawned. We did not expect this to happen, but we noticed that islands were more likely to be dominated by one species. And over time, this dominant genome spreads out to the other islands. This is shown in Figure 8. Another observation was that, the most dominant plants grew to the where the sun set as it maximized their sunlight capture as shown in Figure 9. In terms of crown type. We were able to observe all the different crown types across our simulation. Some forests favored one over another. Whenever there were dominant crowns, we noticed the development of shrubs underneath or over topped crowns (where the crown of the tree leans to one side as seen in Figure 10. Co-Dominant crowns share the canopy, filling in spaces left by the other trees as shown in Figure 11.

6 MEASUREMENTS

To measure the physical representation of the alleles we will look at the number of leaves, the height of a tree, the efficiency of resource collection, and physical properties that are a result of adaptation (see measurement section for further details). To understand competition, we can look at the width and depth of the leaves and branch system (crown of tree). Different crowns can indicate adaptations to a competitive environment. Dominant crowns indicate trees that grow outward and upward to extend above the surrounding canopy architecture. Co-dominant crowns indicate cooperation where neighbors share access to sunlight or fill in gaps between dominant crowns. Intermediate crowns grow to the lower ceiling of the forest canopy, with thinned-out leaf placement to take advantage of holes in the upper canopy. Over-topped crowns are the result of lopsided growth to one side to take advantage of biased sunlight. The above-mentioned traits are ways to measure the "genetic" expression of a tree's L-system.

Number of leaf nodes	The number of leaves the tree was able to generate.
$\frac{Sunlight}{\#\ age}$	Amount resources over lifetime
Number of total cells	The total number of cells. This includes leaves and branches
Height of tree	This measures the length from the base of the trunk to the highest cell.
Width of tree	We want to measure how wide a tree has grown. This can show how dominant a crown has become.
Crown type	Dominant, Co-dominant, Intermediate, or Overtopped
$\frac{sunlight}{\#\ totalcells}$	This looks to measure the ratio of the amount of sunlight collected with the total cells the trees have.
$\frac{sunlight}{\#\ leafnodes}$	This looks to measure the ratio of the amount of sunlight collected with the total leaves the trees have.
Age	How long was a tree able to survive?

Crown type There are various crown types.

- A dominant tree crown reaches over all other plants in the forest, including the crowns of other trees growing outward and upward → extending above the canopy architecture of trees around it.

- A crown that is co-dominant shares access to sunlight with another tree. A co-dominant tree crown receives sunlight at only the tips of its upper branches, as such it doesn't grow to its full potential as a result.
- Intermediate crowned trees grow only about as tall as the lower ceiling of the forest canopy, below the co-dominant tree crowns. As such the crown is shorter and often features leaves with a reduced surface area with thinned-out leaf placement. This helps the tree take advantage of small holes in the canopy.
- Over-topped crowns have smaller crowns as they exist under, or near, more dominant trees. They are often more developed on one side, live in shade, and grow more slowly.

7 COMPARISON

For our simulation we wanted to compare different simulations, as we changed hyper parameters, and their impact on the overall fitness of the population. We choose three factors to manipulate: survivor selection mechanism, the length of simulation (days vs hours), and the rate of mutation.

Using Elitism, only the fittest plants are selected to continue into the next generation. Older trees, who tend to have a higher fitness scores, and will tend to be larger. This size allows them to have superior resource accumulation creating a virtuous cycle of them becoming larger and larger. This limits the growth and viability of new trees, curtailing genetic diversity and allowing smaller trees a chance to compete. We compared Elitism with Ageism. Ageism is a method where we eliminate the oldest 20% of the population every day after the 3rd day before using Elitism.

Trees are given the chance to grow every hour if they have enough resources to do so. At the end of 12 hours, mutation, children creation, and survivor selection takes place. We realized that this structure may favor genotypes that grow quickly and are fast at resource accumulation but may not be optimal in the long run (when they are large). To give slow growth trees a chance, we doubled the hours in a day.

We also compared a simulation of 30 days with that of 60 days. The last comparison we did was doubling the global rate of mutation. This global rate doubles the chance of mutating a tree's dictionary of rules and parameters.

7.1 Comparison Results

(1) Ageism vs Elitism: We realized Ageism only works well when the length of simulation is longer and it necessitates the removal of old trees (> 15hrs and >25 days)
(2) Hours in a Day: The longer the hours in a day, the less of a need to grow outward or taller. It became more economically to create shrub like structures like those found in deserts or dry climates as seen in Figure 14.
(3) Simulation Length: The longer a simulation runs, more chaotic structures are developed and more drastic speciation is as seen in Figure 6.
(4) Mutation Rate. A drastic mutation rate creates nonsense structures like in Figure 13 whereas at 5% mutation rate, the forest starts to become monotone Figure 16. We believe that this is primarily due to the way our mutation methods are created and applied, and perhaps a more gentle mutation function may create more utility.

When using sunlight and branch proportion for fitness, elitism seemed to produce generally fitter individuals than ageism (Figure 17). The average height of the individuals remained close during the simulation regardless of whether ageism or elitism was used (Figure 18), as did the average width (Figure 19. One interesting observation is that the average number of leafs was lower when elitism was employed (Figure 20), the number of branches were roughly the same (Figure 21), but the average fitness was generally higher, suggesting that elitism led to individuals which were more efficient at gathering sunlight.

When using just sunlight for fitness the average fitness values, as seen in Figure 22, were generally close, although elitism still seemed to produce slightly fitter individuals. Elitism resulted in individuals which were generally taller (Figure 23) and wider (Figure 24) than those which were selected using ageism. In contrast to when leaf and branch proportion were both included in fitness, when considering only sunlight elitism resulted in individuals which had substantially more leafs (Figure 25) and branches (Figure 26) than those subject to ageism.

8 DISCUSSION

The greatest challenge we encountered during the development of this simulation was the computational cost being much larger than expected. To effectively run a simulation with approximately 100 plants we needed to cut the Root L-Systems from our initial proposal. While modelling the Roots of plants in addition to their Shoots may yield a more accurate simulation of inter plant competition, we were unable to optimize the generation of the L-Systems to the points where we could feasibly run both a Shoot and Root simulation. We believe the performance issues arise due to the sheer number of leaf and branch objects required to make up a individual plant. Through object pooling we saw significant gains in the number of trees we could support, but simulation of both Roots and Shoots with a reasonable population size would only be possible within Unity if further advances were made in handling the rendering of thousands of object instances. If a similar simulation were made in a program such as Unreal Engine which utilized a technology such as Nanite, there is the possibility for a magnitude greater simulation size.

8.1 Real World Impact

Our project lends insight into the use of L-Systems and Evolutionary Algorithms for the study of artificial life. We demonstrate that plant like structures represented by L-Systems can be placed in a simulation where sunlight is simulated and where features of the plant itself, such as it's ability to receive sunlight are used to determine it's fitness. A more detailed and advanced version of such a simulation may be able to more accurately simulate actual plants, creating a new tool with which to study plant evolution in the real world from within a digital simulation. A simplified version of this could also be used in generative architecture where there is a need to create an "ecosystem" of structures that all use solar energy.

8.2 Comparison With Literature

All of the papers discussed in the literature review sought to prove that L-Systems could be used for studies of artificial life using Evolutionary Algorithms. Our project has satisfied this goal through the successful implementation of an Evolutionary Algorithm capable of evolving a wide variety of digitally represented trees. Three of the papers covered in the literature review pursued the use of a Genetic Algorithm, as we did with this project. Chen and Ross used Genetic Programming instead. As compared to the work of Ochoa, our project uses a fitness function which considers fewer factors in addition to not using the tree based presentation for L-System rules. Chen chose to use a convolutional neural network to identify the structures which most resembled trees, which is different from the arithmetic approach used in our project. Our arithmetic approach towards fitness is heavily inspired by the fitness function used by Ochoa. One key deviation from our work as compared to that of Ochoa, Mock and Chen is the use of 3D structures in our project as opposed to 2D structures. Our project also opts to use a string based representation for the phenotype like Mock, as opposed to Ochoa and Chen who use tree based representation instead. In addition, the use of ray casting to calculate the amount of sunlight absorbed by individuals was not a technique used by any other paper. Our experience with the development of our algorithm, especially in regards to crossover, supports the conclusion that genetic programming

is a more suitable approach to this problem owing to its tree based representation. It would be simpler to develop a crossover function that returns a syntactically correct L-system rule using sub-tree swapping than with the string based methods used in this project.

8.3 Key Learnings

We learned that if too much space was given to for the trees, there was not enough selective pressure for plants to change or adapt. It is important to get the right density of population to area. Tuning hyper-parameters to replicate different environments could be the subject of another entire paper. As outlined in the comparison section, different parameters can create different ecosystems (24hr sunlight creates plants found in deserts). In longer simulations we noticed that midway, none of the plants were cohesive, but as the simulation moved past this "valley" the structures looked more plant-like and more cohesive.

Through experimentation we began to notice that many plants were growing extremely long branches due to many consecutive *F* characters (the symbol for growing a single branch) resulting from crossover. A plant that grew in this manner in real life would not be structurally sound but we do not model gravity in our simulation. Instead we added a exponential cost to consecutive F occurrences by squaring the branch cost the number of times F's occur next to each other in the plants Rules.

8.4 Further Exploration

We believe that there is a lot more than can be explored with our simulation.

(1) Currently only a branch L-system has been implemented, we could get a more holistic view of forest growth if there was a root system incorporated into the genome as well. This would necessitate the introduction of water as a resource, a root L-system, some consideration of how water will be represented in the ground, and its interaction with the growth of the tree.
(2) Further study could look into the interaction of plants and animals with herbivores introduced into the simulation. These animals can eat the plants, allowing the plants to make decisions on whether to grow or repair branches and leaves.
(3) Our simulation has the plants grow with no clear direction. In nature, plants and trees grow towards sunlight. A further iteration of this project could code growth to occur in areas where the plants received the most sunlight - this would allow the tree to "chase" the resource over its lifetime.
(4) In nature, there is often a mycelium net that connects the root systems of trees. This network allows trees to negotiate and trade resources with each other. The next iteration could introduce this to the simulation as well.
(5) Other iterations could also implement a growth factor that is embedded in the genome. We can mutate the growth factors. Regarding rules we can also use a 2-point crossover between 2 rules. We generate two random pairs for the index values, and swap the substrings between the two rules (indicated by those index values).

9 APPENDIX

9.1 Execution of Code

The code repository can be found here: https://github.com/Jackmactaylor/CISC-455-L-Systems-Project
To execute the code you may either run the Demo "L-Systems.exe" in the Demo Build folder (you must download the

entire Demo Build folder, not just the L-Systems.exe) included in the base repository or install Unity 2019.4.40f1 within Unity Hub and clone the GitHub repository. Step by step instructions are provided for clarity.

(1) Download GitHub Desktop and login using a GitHub account
(2) In top left click "Add", clone repository using the link https://github.com/Jackmactaylor/CISC-455-L-Systems-Project
(3) Fetch the project at the top once selected
(4) Download Unity Hub
(5) Install 2019.4.40f1, goto Installs->Install Editor->Archive->download archive and find 2019.4.40f1, it will open back in Unity Hub
(6) In Unity Hub open the project from where you cloned the Repository in GitHub Desktop
(7) It will take a while initially to import and solve everything but it will only happen once
(8) Within Unity you should open the scene "V2 Test" under Assets/Scenes/V2 Test"
(9) All relevant simulation settings can be found by clicking on the "SimulationManager" or "EvolutionManager" objects in the Hierarchy (usually on the left side) and looking in the Inspector tab (usually on the right side)

The Demo build is included for ease of access and novelty purposes (controls are shown in the bottom left, there is no way to close it other than alt+F4) but access to the full range of tune-able parameters used for modifying the simulation is only possible within Unity. It is also important to note that caution should be used when running auto simulation for more than a few days on lower end machines, the memory required to run the simulation is very large and it can sometimes lock up the entire computer. Manually iterating through the hours is recommended unless you are collecting data.

9.2 Figures

X Axioms:

- `X -> [F[+FX][*+FX][/+FX]][F[-FX][*-FX][/-FX]]`
- `X -> [F[+FX][*+FX]][F[-FX][*-FX]]`
- `X -> F[+FX]F[-FX]F[+FX]F[-FX]`
- `X -> [F[+FX][*+FX][/+FX]]`
- `X -> [*+FX]X[+FX][/+F-FX]`
- `X -> [F[-X+F[+FX]][*-X+F[+FX]][/-X+F[+FX]-X]]`

Fig. 1. Axioms of X used during randomized rule initialization

REFERENCES

[1] Ross B. J. Chen, X. E. 2021. Deep Neural Network Guided Evolution of L-System Trees. *2021 IEEE Congress on Evolutionary Computation (CEC)* (2021), 2507–2514. https://doi.org/10.1109/CEC45853.2021.9504827

[2] Parslow P. Lundqvist K. Ø Fitch, B. G. 2018. Evolving complete L-systems: Using genetic algorithms for the generation of realistic plants. (2018), 16–23. https://doi.org/10.1007/978-3-319-90418-4_2

[3] K. J. Mock. 1998. Wildwood: The evolution of L-system plants for virtual environments. *IEEE World Congress on Computational Intelligence* (1998). https://doi.org/10.1109/icec.1998.699854

[4] G. Ochoa. 1998. On genetic algorithms and Lindenmayer Systems environments. (1998), 335–344. https://doi.org/10.1007/BFb0056876

F Rules:

- `F -> FF`
- `F -> [+F][-F][*F][/F]`
- `F -> F+F-F`
- `F -> F[+F]F[-F]F`
- `F -> F[*+F][-F][*F]`
- `F -> F[/F*F*F][/F*F*F]/F`

Fig. 2. F Rules used during randomized rule initialization

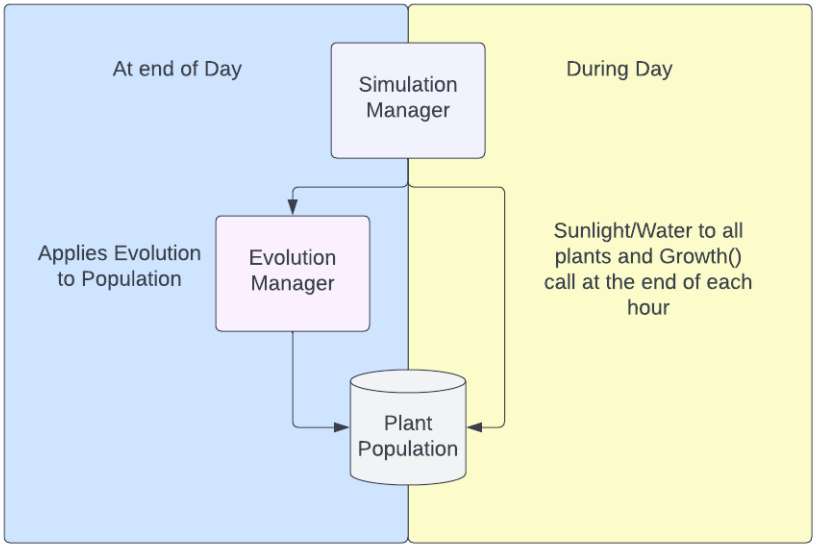

Fig. 3. The high level structure of our simulation

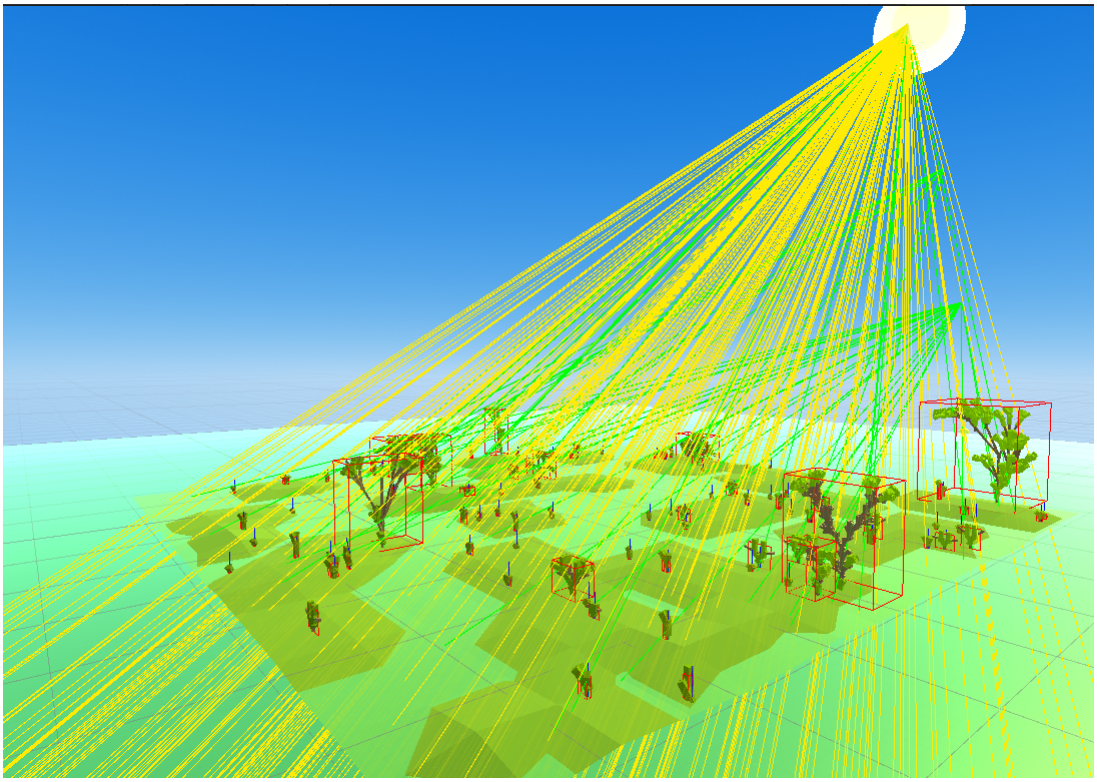

Fig. 4. 5 Plants Genomes randomized and grown based on received sunlight. Yellow rays are raycasts for the current hour while green rays trace successful sunlight raycasts that hit leafs throughout the day

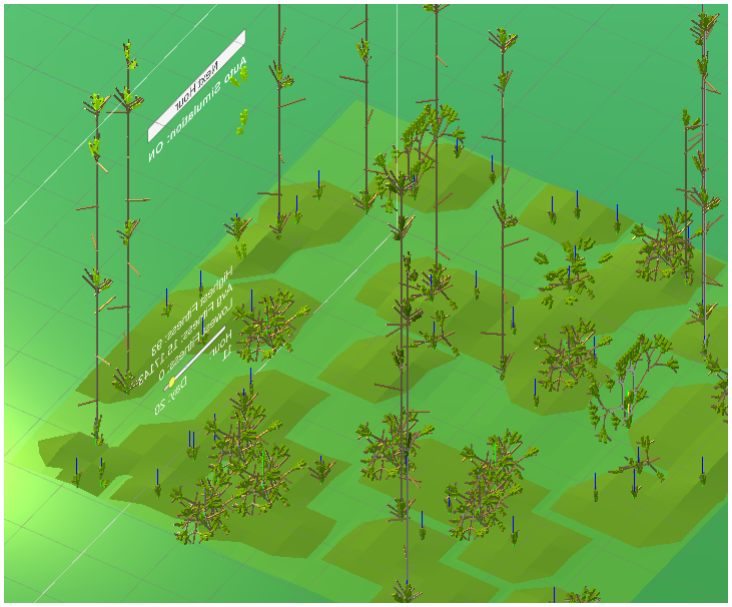

Fig. 5. Speciation in Trees

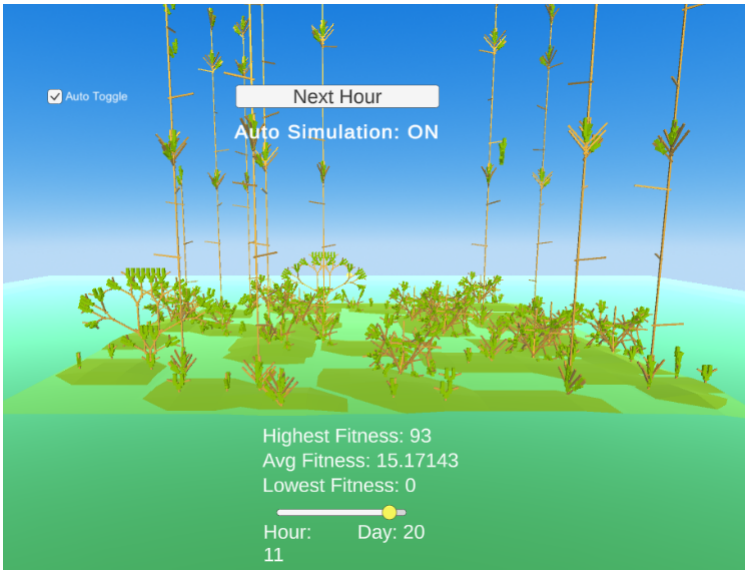

Fig. 6. 40 Days Simulation: Day 20

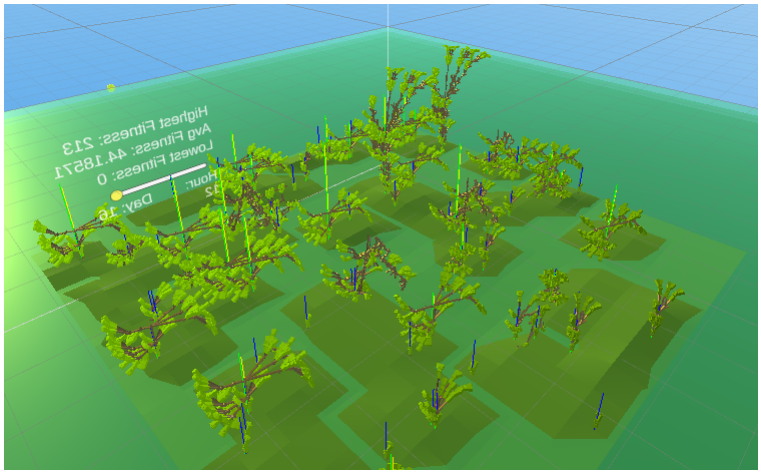

Fig. 7. Sunlight Fitness Function with Elitism

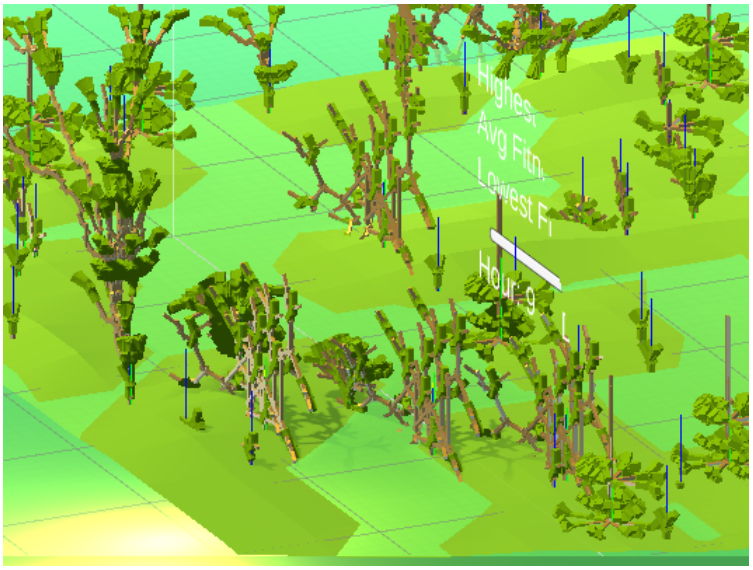

Fig. 8. Island Hopping

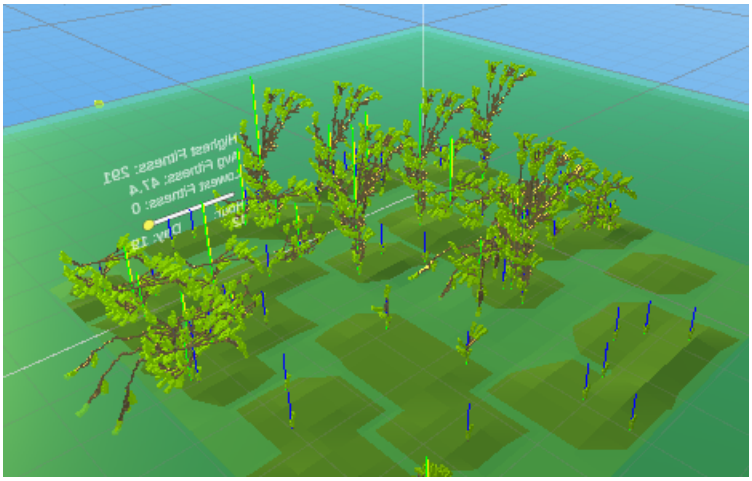

Fig. 9. Plants Growing Where sunlight is most prevalent

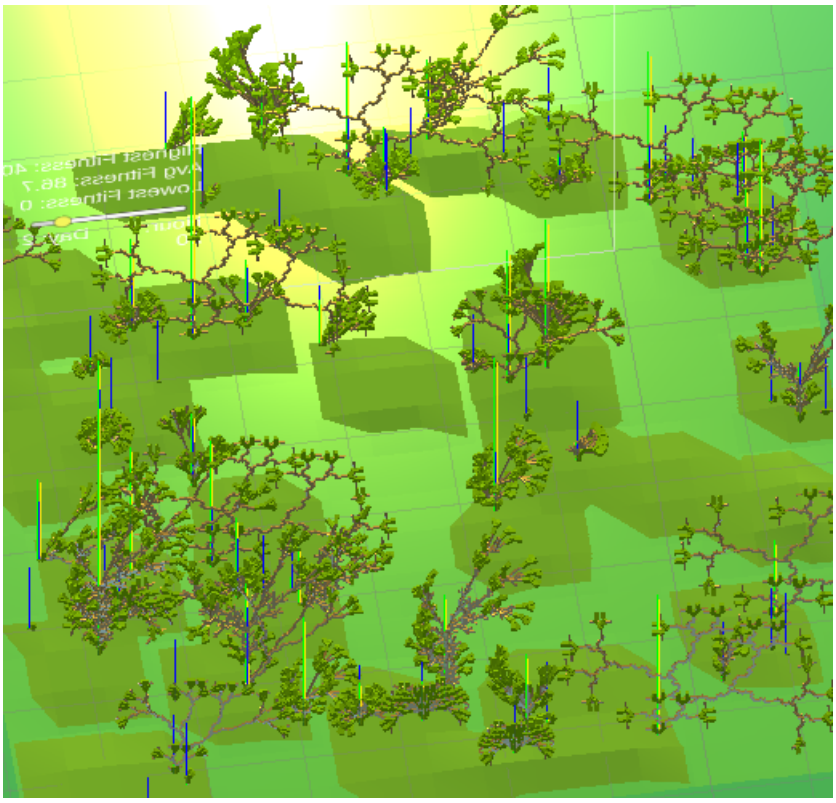

Fig. 10. Overtopped Crowns

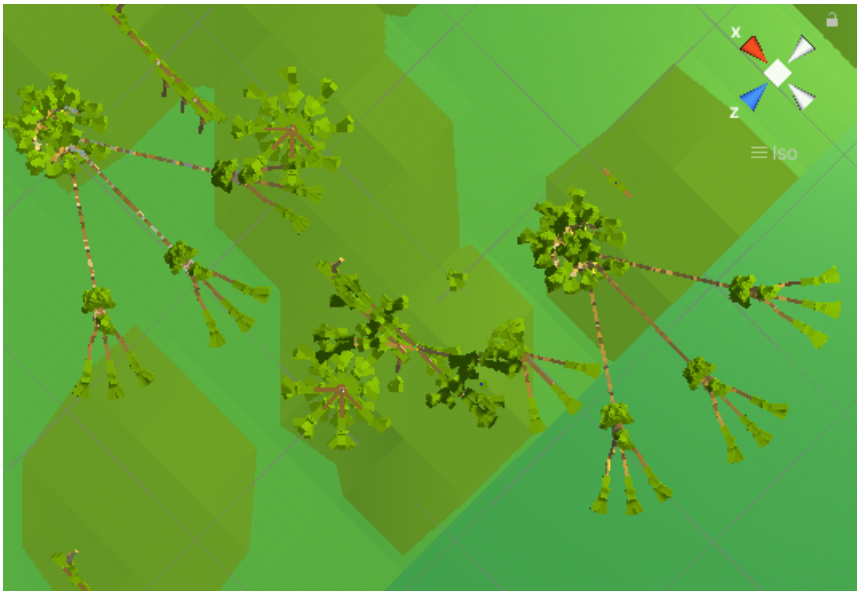

Fig. 11. Co-Dominant Crowns

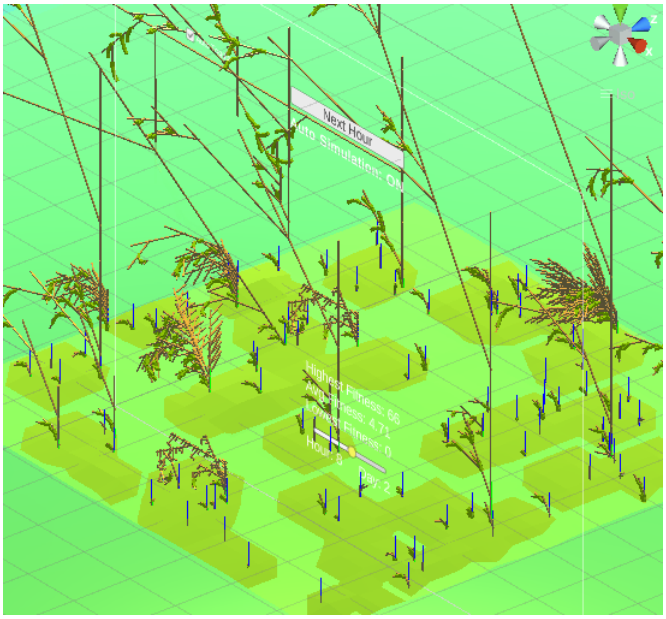

Fig. 12. N-point Crossover creating Reeds

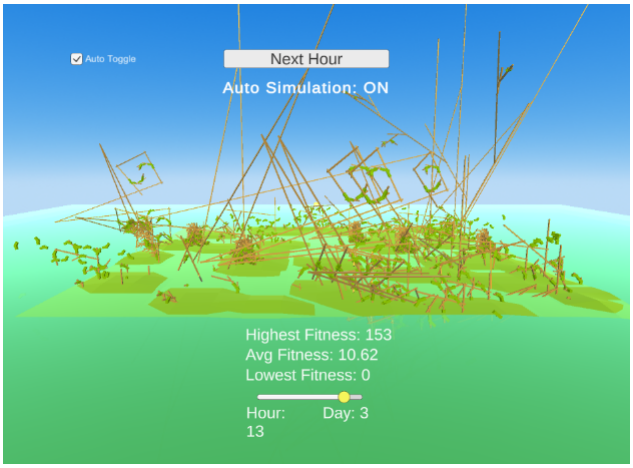

Fig. 13. Mutation Rate at 20%

Fig. 14. 24 hour day with ageism

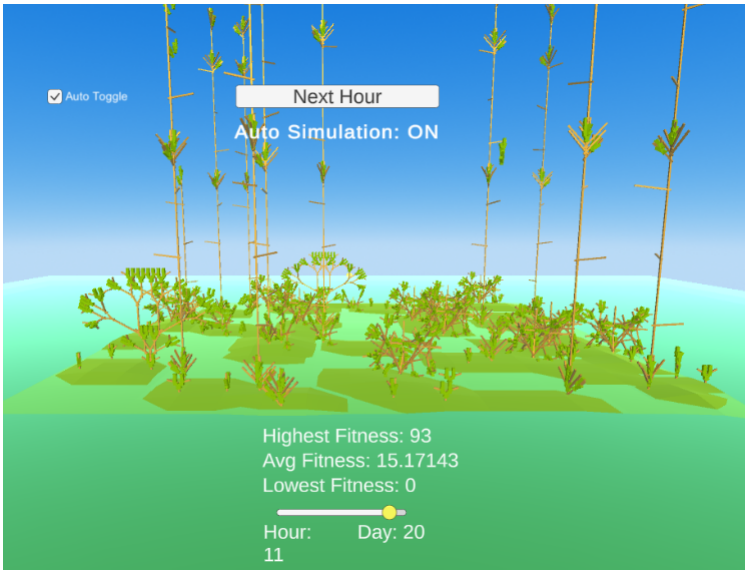

Fig. 15. Simulation with 40 days and Elitism

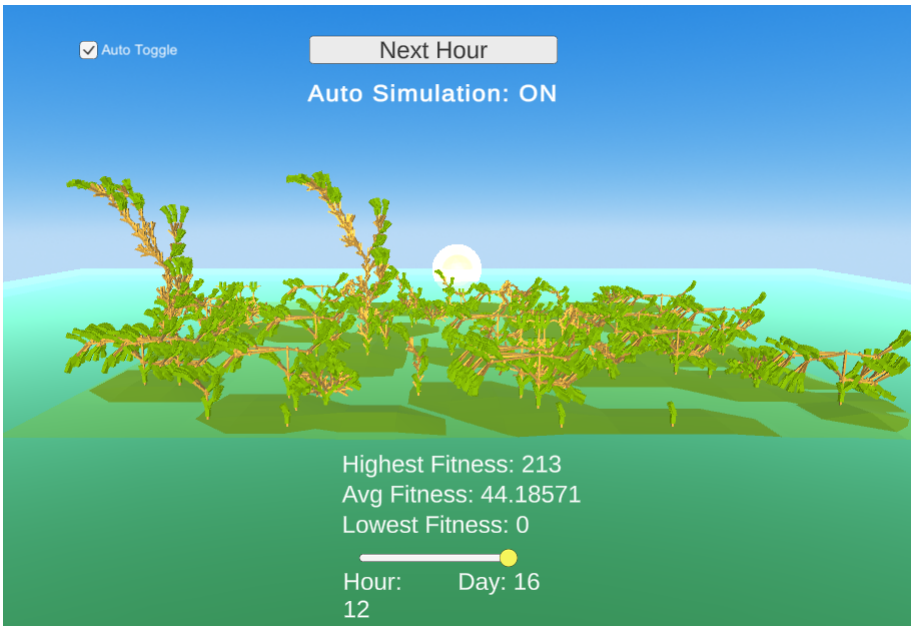

Fig. 16. Mutation rate at 5%

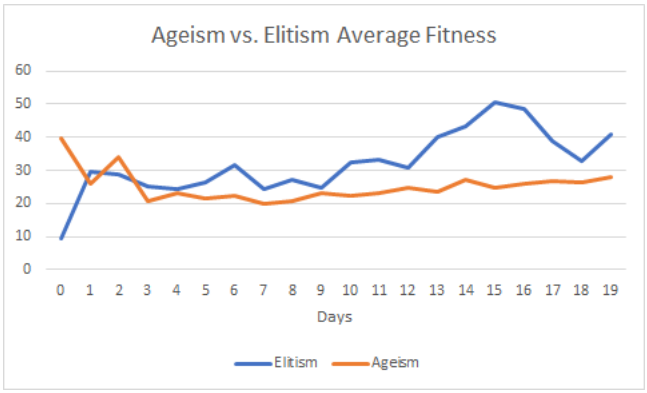

Fig. 17. Average Fitness Using Sunlight and Branch Proportion for Fitness

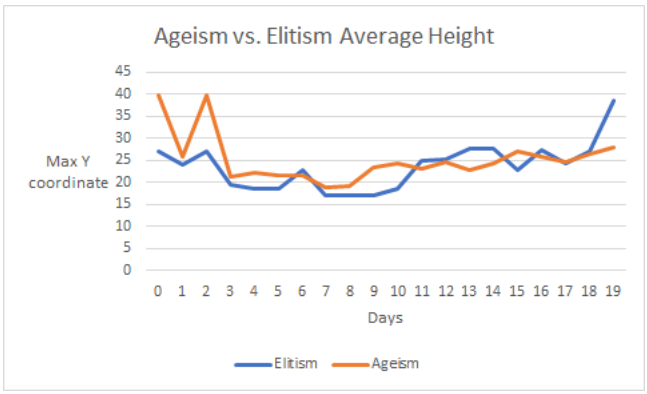

Fig. 18. Average Plant Height Using Sunlight and Branch Proportion for Fitness

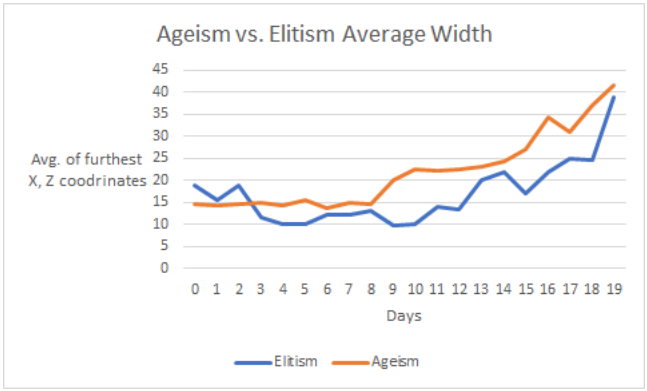

Fig. 19. Average Plant Width Using Sunlight and Branch Proportion for Fitness

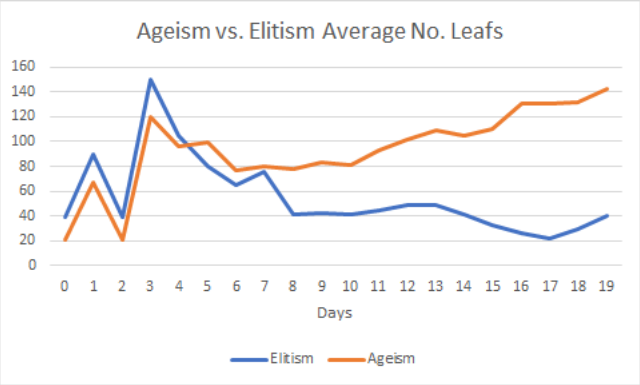

Fig. 20. Average Plant Leaf Count Using Sunlight and Branch Proportion for Fitness

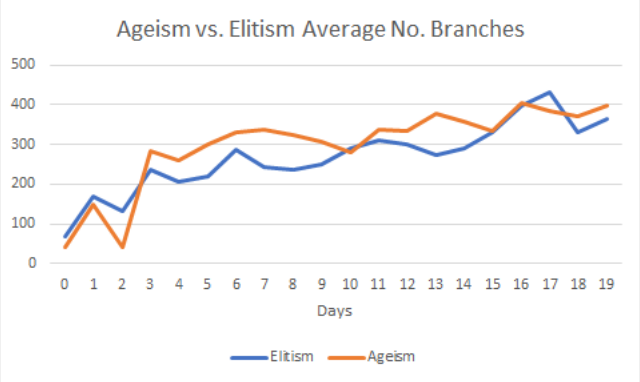

Fig. 21. Average Plant Branch Count Using Sunlight and Branch Proportion for Fitness

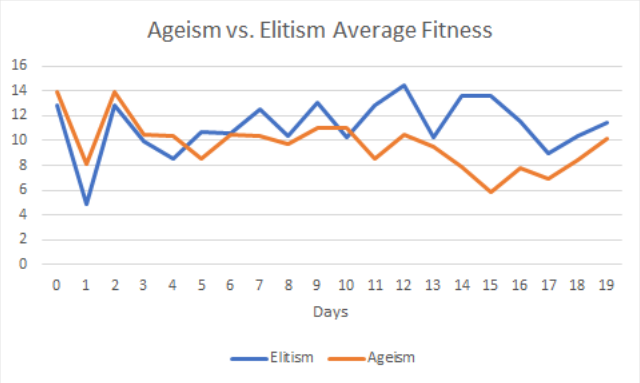

Fig. 22. Average Fitness Using Sunlight for Fitness

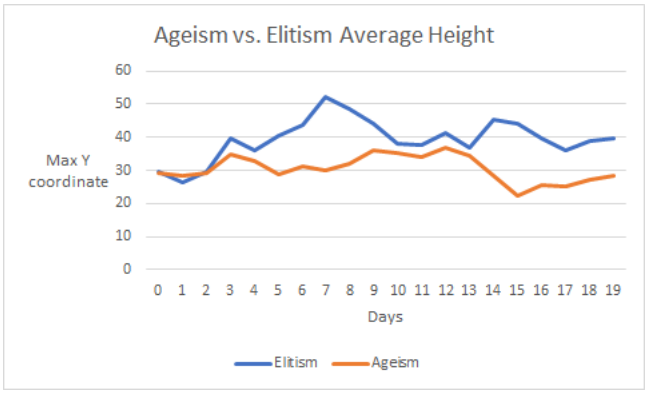

Fig. 23. Average Plant Height Using Sunlight for Fitness

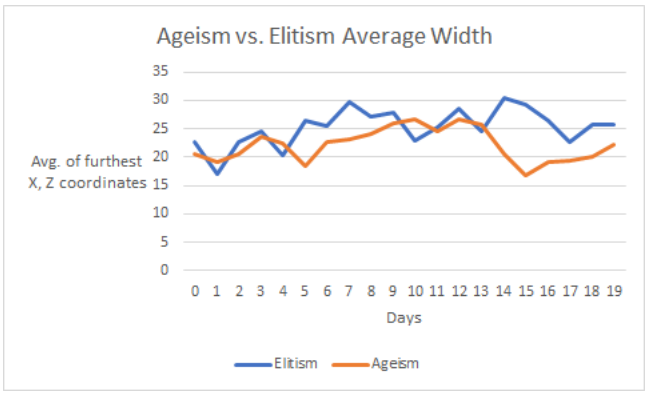

Fig. 24. Average Plant Width Using Sunlight for Fitness

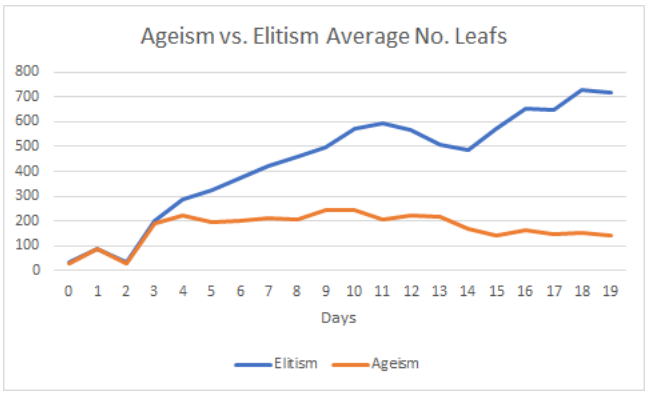

Fig. 25. Average Plant Leaf Count Using Sunlight for Fitness

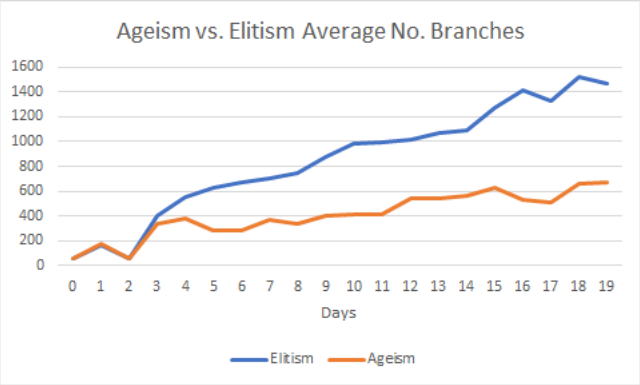

Fig. 26. Average Plant Branch Count Using Sunlight for Fitness