UNIVERSITY OF
LINCOLN

SCHOOL OF MATHEMATICS AND PHYSICS

YEAR 3 PROJECT

GARDENS OF EDEN IN THE GAME OF LIFE

ROBIN CRITTEN (25677324)

SUPERVISED BY
DR. GED COROB COOK

APRIL 2024

SCIENTIFIC REPORT IN THE 3rd YEAR MODULE MTH3009M MATHEMATICS PROJECT

Contents

1 Abstract

Gardens of Eden are special configurations that can occur within Conway's Game of Life, but only as initial conditions. As the state preceding a known configuration is impossible to determine exactly, all possible precursor configurations therefore need to be considered when proving if a configuration is a Garden of Eden. This paper aims to propose a potential method for identifying potential Garden of Eden configurations, by narrowing the range of possible precursors to a known configuration. More specifically through the mapping of precursor and successor configuration densities to create a reliable method of random sampling that can achieve the same results in a reduced time frame. These results can then be used to create a frame of reference for the range of precursor densities that must be considered when proving whether a configuration is a Garden of Eden.

2 Introduction

Cellular automata are discrete systems imposed upon a graph. In which vertices are all connected by an equal number of edges across an infinite environment, that contains a finite number of non-quiescent vertices. They depend on time, and a mapping function $f : A \rightarrow A$ is applied across the environment to simulate a step forward in time, where A represents the set of possible outputs for the mapping function. This has allowed Cellular automaton to be used to simulate real scenarios such as Stephen Wolfram's work simulating complex physical and biological processes [Hol22].

If a cellular automaton contains at least one pair of mutually erasable configurations, then it must contain Gardens of Eden by the Moore-Myhill Theorem [Myh63] [Moo62]. A Garden of Eden is a configuration within the set of possible configurations a cellular automaton can produce, that can only appear at time $T = 0$. In other words, if a configuration C exists at time $T+1$ that is a Garden of Eden, there would be no precursor that exists at time $T = 0$ that could produce such a configuration.

An example of a cellular automaton that contains Gardens of Eden would be the Game of Life, created by John H. Conway. It was popularised by an issue of the Scientific American soon after creation and is described as "a statistical model designed to mirror simple biological processes, most specifically the birth, evolution, and death of a population" [Gar70]. It is played upon a two-dimensional grid, each vertex, or cell, capable of being either 'alive' or 'dead'. The 'dead' state is considered to be quiescent. Cell states can change during a step in time during which the mapping function utilizing the rule set will consider the states of the surrounding cells to produce an output.

The first Garden of Eden was discovered in 1971 by Roger Banks [Bee22]. Later studies, such as Symmetry in Gardens of Eden [Har+13] have focused on using new techniques to push the boundaries of known Gardens of Eden. In the case of Christiaan Hartman, he utilized known symmetric Gardens of Eden, and optimized search algorithms to prove the existence of new lower bounds for the smallest Garden of Eden [Bee22].

Randall D. Beer has improved the scope of knowledge surrounding Gardens of Eden, by counting the maximum and minimum number of precursors as a function of density, for varied sizes of bounding box [Bee22]. I aim to create an algorithm that maps the density of precursors against the density of their successors. This will expand upon Randall's work by giving further data points to narrow the search scope when looking for Gardens of Eden. I will also be aiming to consider expanding this data set into multiple dimensions, to see if trends in the second dimension can be replicated, or how they differ.

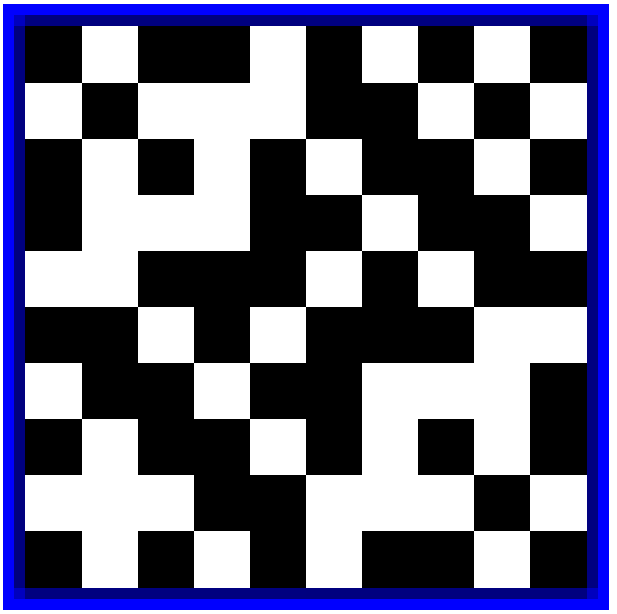

Figure 1: 10x10 Garden of Eden on an infinite environment with the least non-quiescent cells, where white represents non-quiescent cells and black represent quiescent cells. The blue border represents the border of the configuration, the infinite environment extending beyond the border. [Bee22]

3 Preliminaries

3.1 Cellular Automata

All automata are governed by a set of rules. Examples of types of rules that can be used to dictate cellular automata include cell states. All automata have a set of states, A, that cells can be designated at discrete times T. As part of this set, all states bar one are considered to be non-quiescent, or 'alive'. Any remaining cells are therefore quiescent, or 'dead'. Furthermore, a passage of time from T to $T+1$ will change the configuration of the environment in most cases. This is because all cells both quiescent and non-quiescent will be subject to rules that determine their future state. For example, if at time T a certain number of cells of state X are within a set distance, denoted by the rules, from the cell of state Y being inspected, then the state may change to another within the alphabet A. The term used to denote the area that can influence the future state of a cell being inspected is referred to as the neighborhood. Not all neighbourhoods will necessarily be equal in length for each axis branching from the cell being inspected depending on the model. However the distance from the inspected cell will likely be measured in the number of vertices.

Formula for calculating the number of possible configurations in an n dimensional grid, where each dimension is of size m. a being the number of unique identifiers with the set of cell states.

$$a^{m^n}$$

Von Neumann is credited for the invention of cellular automata. He devised the premise in 1948 and later coined the Von Neumann machine [Hol22]. A machine, or cellular automaton, that can replicate itself using the above definitions provided. This can be said to have inspired later works that are the subject of interest to this paper, such as Machine Models of Self-Reproduction by E.F. Moore [Moo62]. This notion is further established by Moore's refer-

ence to Neumann's previous work on cellular automata [Von49]. Moore also refers to cellular automata as tessellations or tessellation structures. For simplicity, I will be replacing any use of tessellation with cellular automata, because, through my research into cellular automata, I have concluded the terms are interchangeable. Cellular automata being the preferred term in modern research papers.

3.2 Game of Life

John H. Conway created the Game of Life (GoL), also referred to as Conway's Game of Life in 1970. It is described as "a statistical model designed to mirror simple biological processes, most specifically the birth, evolution, and death of a population" [Hol22]. Soon after creation, the game was published and popularised by an issue of Scientific American [Gar70].

GoL is played on a two-dimensional grid using two cell states. These are represented by black and white cells, in my simulations the black cells are considered to be quiescent and the white non-quiescent. Traditionally the cell neighbourhood is a 3 by 3 grid with the centre being the inspected cell. This type of neighborhood is called a Moore neighborhood. The initial state of the game, time $T = 0$, is manually created by the player, all succeeding states are therefore a result of the following rules.

- Any non-quiescent cells will remain non-quiescent if they have two or three neighbors, else they will transition to quiescent at time $T + 1$.
- Any quiescent cells will transition to non-quiescent at time $T + 1$ if they have exactly three neighbours

The above is a condensing of the rules, from their original four. Which serves to highlight its beginnings as a model for biological processes. Names for the rules have been given and are respectively 'loneliness', 'survival', 'overpopulation', and 'birth' [Bee22]. As 'loneliness', 'survival', and 'overpopulation' all regard the successive state of non-quiescent cells, they have been combined. 'Loneliness' refers to less than two neighbors existing in the Moore neighborhood, resulting in a transition into a quiescent state. 'Survival' is the continuation of the current state under the condition of there only being two or three neighbors. 'Overpopulation' is when there is an excess of three neighbors resulting in a transition to a non-quiescent state. The final rule 'birth', remains unchanged from the second of the originally stated rule.

A mathematical expression can also be used to describe the rules, one created by Randall D. Beer is [Bee22]:

$$\phi(s, \Sigma) = \delta_{\Sigma,3} + s\delta_{\Sigma,2}$$

Where $_\Sigma$ is the sum of all non-quiescent cells in the neighborhood and s is the current state of the cell, 1 is representative of a non-quiescent cell. 0 otherwise. $\delta_{i,j}$ is the Kronecker delta function. In summary, this function is a consolidation of the rules such that the output is 1 if and only if the cell in question has three non-quiescent neighbors, or is non-quiescent and has two neighbors. The output determines the state of the cell at time $T + 1$. When applying the formula to a configuration, C, the image of C at $T + 1$ can be calculated by applying the function globally to every necessary element.

$$\sum_{i=l}^{m} \binom{n}{i}$$

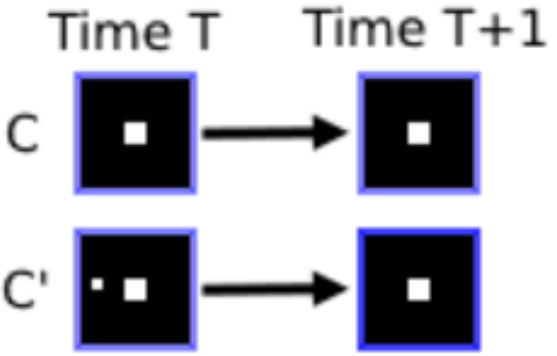

Figure 2: Two configurations C and C'. They are placed within a game of life environment E, that are distinct at time T, yet are identical at time $T+1$. Hence they are mutually erasable.

Is a formula for calculating the number of possible configurations in the GoL between a set of bounds [Bee22]. Where l is the lower bound and m is the upper. n being the size of the grid.

3.3 Moore-Myhill Theorem

In machine models of self-reproduction, published in 1962. Volume 14 contained a theorem written by Edward F. Moore [Moo62].

Theorem 2. *For a tessellation structure for which there exist erasable configurations there exist Garden of Eden configurations [Moo62].*

A tessellation structure is, from my understanding, interchangeable with cellular automata. So in future, I will be referring to them as such unlike in Moore's original work. Mutually erasable configurations can be defined as two or more distinct structures located within a cellular automata environment at time T, such that when the mapping function is applied their states at time $T+1$ are identical. See figure 1.

If a configuration is bound by a grid of size n in d dimensions at time $T+1$, then at time T the boundary of the preceding configuration must be considered to be size $n+2$. This is what I am going to refer to as the boundary condition, and holds as long as a Moore neighborhood is being used in the cellular automaton's rule set. Changes in the neighborhood used will affect the boundary condition. For example, instead of a three-by-three neighborhood with the center being the inspected cell, the neighborhood used was five by five. The boundary condition would instead be $n+4$.

The following is Moore's proof, of his previously stated theorem, extrapolated into d dimensions [Moo62]. Consider an equivalence relation that holds when distinct n^d configurations are either identical or a mutually erasable pair. Where n is a sufficiently large integer

such that an n^d configuration can hold a mutually erasable pattern. Assuming there is at least one mutually erasable pair, there is therefore a maximum of $a^{n^d} - 1$ that the equivalence relation can divide the set a^{n^d} configurations into. Where a is the positive integer representing the amount of possible cell states, given by the set of cell states. $a^{n^d} - 1$ can also be described as the minimum possible loss due to erasure. Erasure is when multiple unique configurations map to the same successor state.

Consider a positive integer k, such that there is a grid of length kn in d dimensions, resulting in k^d n^d sub-configurations within this configuration. If there are two $(kn)^d$ configurations c and c' they will have the equivalence relation R* if and only if each of the n^d sub-configurations of c have the relation R to their counterpart in c'. If R* is an equivalence relation the number of equivalence classes is at most $(a^{n^d} - 1)^{k^d}$.

Using the previously defined $(kn)^d$ grid, loss resulting from the boundary condition can be defined as $a^{(kn-2)^d}$. For Moore's theorem to hold the loss due to the boundary condition needs to be less than the loss due to erasure. Therefore need to prove that there exists a sufficiently large positive integer k, such that the following inequality holds.

$$(a^{n^d} - 1)^{k^d} < a^{(kn-2)^d}$$

This can be proven under the following conditions; $a > 1$, $n > 1$,$d > 0$, $a^{n^d} > a^{n^d} - 1 > 0$, therefore $a^{n^d}/(a^{n^d} - 1) > 1$. This means that k can be an integer such that.

$$k > (2\binom{d}{1}(n^{d-1}))/\log_a(a^{n^d}/(a^{n^d} - 1))$$

As a, d, and n are all fixed constants, k is therefore a positive integer in this inequality. Multiplying by the logarithm and dividing by k then gives.

$$\log_a(a^{n^d}/(a^{n^d} - 1)) > (2\binom{d}{1}(n^{d-1}))/k$$

Taking away both the left-hand side and right-hand side of the inequality from both sides results in the following.

$$\log_a((a^{n^d} - 1)/a^{n^d}) < (-2\binom{d}{1}(n^{d-1})(k^{-1}))$$

As the right-hand side is greater than the left, adding a positive constant to the right-hand side will not change the validity of the inequality.

$$\log_a((a^{n^d} - 1)/a^{n^d}) < (-2\binom{d}{1}(n^{d-1})(k^{-1})) + \sum_{r=2}^{d}\binom{d}{r}(n^{d-r})(-2)^r(k^{-1})^r$$

The right-hand side can then be rewritten to create.

$$\log_a((a^{n^d} - 1)/a^{n^d}) < \sum_{r=1}^{d}\binom{d}{r}(n^{d-r})(-2)^r(k^{-1})^r$$

By setting both sides of the inequality as exponents of the base a it simplifies to.

$$(a^{n^d} - 1)/a^{n^d} < a^{(\sum_{r=1}^{d}\binom{d}{r}(n^{d-r})(-2)^r(k^{-1})^r)}$$

Then multiply both sides by a^{n^d}

$$(a^{n^d}-1) < a^{(n^d+\sum_{r=1}^{d}\binom{d}{r}(n^{d-r})(-2)^r(k^{-1})^r)}$$

The right-hand side can then be simplified to make

$$(a^{n^d}-1) < a^{(\sum_{r=0}^{d}\binom{d}{r}(n^{d-r})(-2)^r(k^{-1})^r)}$$

Which can then be rewritten as

$$(a^{n^d}-1) < a^{(n-2/k)^d}$$

Then raise both sides by the power k^d

$$(a^{n^d}-1)^{k^d} < a^{(kn-2)^d}$$

Therefore there is a sufficiently large k exists such that the loss due to erasure is greater than the loss due to the boundary condition. Hence the inequality holds true and therefore Gardens of Eden must exist.

Myhill later created the converse of Moore's theorem in 1963, completing the Moore-Myhill theorem. The following is his proof extrapolated into d dimensions [Myh63]. Working under the assumption that there is one GoE and all other pairs of n^d configurations, placed upon a $(kn)^d$ grid, are distinguishable at time $T+1$. Assuming that they are placed at time T. Therefore there must be at least as many successor configurations of the $(kn)^d$ grid as there are $a^{(kn-2)^d}$ in total. Furthermore as there is exactly one GoE configuration there cannot be more than $(a^{n^d}-1)^{k^d}$ configurations in the $(kn)^d$ grid.

If v is said to be the actual number of successor configurations, this leads to the following inequalities.

$$a^{(kn-2)^d} \leq v$$
$$(a^{n^d}-1)^{k^d} \geq v$$

These can be combined to create the inequality.

$$a^{(kn-2)^d} \leq v \leq (a^{n^d}-1)^{k^d}$$

This contradicts the inequality, $(a^{n^d}-1)^{k^d} < a^{(kn-2)^d}$, for large k. Therefore proving that Mutually erasable configurations must exist for GoEs to exist, for cellular automata in d dimensions.

4 Method

4.1 Game of Life Simulation

For creating a simulation of the GoL Netlogo was used as a programming language. The reason for this choice lies primarily in that I have little experience with creating dynamic graphical user interfaces. The only experience I have had in the past was with Netlogo, so my familiarity with it led to the ultimate choice. However, it also comes with the benefit of being designed around the creation and simulation of artificial intelligence, of which individual cells within a cellular automata environment can be described as simplistic AI. Therefore lending further credence as to why Netlogo is a beneficial choice for hosting my simulation. If I was to choose a different coding language it would have likely been Python, as I am aware

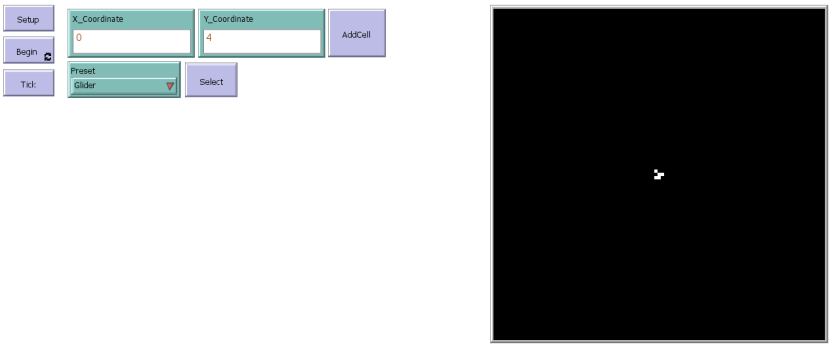

Figure 3: Graphical user interface of the GoL simulation, including the display for the GoL.

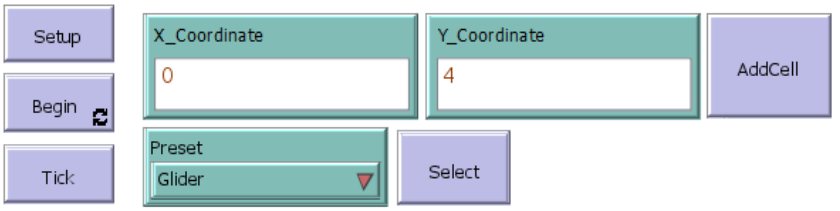

Figure 4: Graphical user interface of the GoL simulation, excluding the display for the GoL.

of multiple Python packages that can be used to create interfaces. However, I have not used them before and the time taken to learn the packages I believed would be better spent coding the simulation.

Figure 3 shows the GUI of the GoL simulation. As can be seen in figure 4 the top leftmost button is the setup button, labelled 'Setup'. What this button will do is initialize both the grid on which the GoL will be displayed as well as several variables important to the running of the code. The variable initialized are the global variables defined at the top of the code and are so due to being used in almost every function defined within the code. As the functions can be called by the user independently of one another through the user interacting with the GUI, they must be defined globally as the functions would otherwise not be able to pass them to one another.

The 'Preset' chooser and 'Select' button widgets are part of a set of two widgets. 'Preset' contains a drop-down list of preset options for the user to choose from and test. Once an option is selected from the list, interacting with 'Select' will then implement the chosen option into the center of the display. However, if an option is selected in such a manner, any configurations currently within the environment will be removed. The reason for this is to prevent conflict with any existing configurations that may have been placed prior. Furthermore, there is a preset option of a blank grid, if a user wishes to remove a previous choice made. How this has been implemented using code is once 'Select' has been interacted with the list containing all current non-quiescent values will be emptied and all patches in the display will have their colour changed to black. The color for quiescent cells. The appropriate patches will then be made non-quiescent having read the current 'preset' value and running the correct if statement. A message will then be printed to the console to inform the user the task has been completed successfully. This section could have been written differently by having the configurations written to CSV files, and then read into the display. While being more complex to implement it would have improved the readability of the code.

Manually adding cells can be done by the user inputting coordinates into

'X_Coordinate' and 'Y_Coordinate' input widgets respectively, before selecting 'AddCell'. The respective cell will be made non-quiescent unless it is not a valid coordinate or is already non-quiescent. In this case, an appropriate error message will be displayed. How the code checks to see if a cell is already non-quiescent is to first check the coordinates are valid for the set of patches. Then compare the coordinates against the list of non-quiescent coordinates to see if they exist in the list. If it isn't a duplicate coordinate the patch at the coordinates will be made non-quiescent.

'Begin' and 'Tick' provide two alternate methods of simulating the GoL. 'Begin' will loop the simulation indefinitely, until one of two conditions is met. Either there are no longer any non-quiescent cells left, or all cells have died. Or the simulation has multiplied exponentially resulting in an excess of non-quiescent cells. I have made the code to terminate in this case, to prevent any issues occurring on the user's end relating to an excessively large number of coordinates being stored in RAM. 'Tick' uses the same code as 'Begin' but only for a single unit of time within the simulation. The equivalent of taking a configuration from time T to time $T+1$. Allowing the user to inspect each distinct moment individually, whilst 'Begin' can be more useful for examining trends in behavior.

How a single tick operates is first the 'Begin' function is called, when either 'Begin' or 'Tick' are selected. The Begin function then calls the iterate function, which initializes the counter variable for how many non-quiescent neighbors a cell has. Furthermore, variables storing lists that keep track of which cells will transition states from quiescent to non-quiescent, or from non-quiescent to quiescent will be reset. As well as the list that tracks quiescent cells within a Moore neighborhood of a non-quiescent cell. Then all quiescent cells are inspected, if they don't have two or three neighbors, those cell coordinates will be added to the list of non-quiescent cells that are going to become quiescent. As this occurs any quiescent cells that are neighbors with the non-quiescent cell being inspected will be added to a separate list, if they aren't already part of it. This list of quiescent cells is then inspected in the same manner as the non-quiescent except if they have exactly three neighbors then they will be added to a list of cells that are going to become non-quiescent. Finally, the state of cells is updated using the created lists, removing cells from the list of non-quiescent cells where appropriate, as well as adding new cell coordinates. After the iterate function has completed checks are undergone by the begin function to ensure it should allow the program to continue running.

Simulating an infinite grid is achieved by coordinates not having to exist on the display grid for them to be counted as valid coordinates. This was achieved by coding in a way that assumes that coordinates are not valid, and only updating the grid to reflect them if they are.

4.2 Precursor Density Mapping Algorithm

For mapping the densities of precursory configurations to their successors I have used Python. As with my choice for Netlogo, it was largely due to familiarity. I have frequently used the matplotlib python package that allows for the creation of plots, that would be necessary for plotting my results. Another is the features supplied by my Python environment of choice, spyder. Plots can be generated without opening separate windows to contain them, making the process of testing code often simpler at times. Due to the relative simplicity of the mathematics behind the GoL, largely any coding language would have been sufficient for the task, however, python is the only language in which I have experience with CSV files and plotting data. Therefore making it the obvious choice for generating results and displaying them.

In addition to the base functionality of Python, four packages have been used. Math was

used to calculate binomial coefficients, as part of a check that was added in later iterations of the code. This check was designed to prevent exceptionally long runtime; The CSV package was used to implement CSV functionality to the program, as results are being written to CSV files; Numpy was used in the creation of arrays; Playsound was also used as a means to send an alert, it has no bearing on the results themselves and simply offers means of acknowledging the program has finished running.

Simulating an n dimensional grid was achieved by creating a one-dimensional list of all possible coordinates, each coordinate being a tuple of length n. First, a function is called that when given the size of the required grid and the number of dimensions will then generate the appropriate variables that can then be fed into the recursive function that will create the coordinates. Each set of coordinates is then appended to a different list. The reason an n dimensional grid is being stored in a one-dimensional list is because a point in Euclidean space has its own unique set of coordinates. Hence storing coordinates in a manner that only serves to reference the coordinates themselves is a pointless endeavour, coupled with the fact that it will only make indexing more difficult in other areas of the code.

The first set of coordinates created is the precursor grid, with $(m+2)^n$ unique coordinate to satisfy the boundary condition. A second list is then made to contain all of the successor's coordinates and is filled utilizing a different function that takes both lists as parameters. The successor list is then filled with all coordinate sets from the precursor list that don't contain a coordinate of the value $m+1$ or 0.

Two different two-dimensional arrays are used to store data. One is a cumulative set of results, recording every data point the code generates. The other records results for each band of precursor density, before writing them to the CSV and then returning to its initial state. The reasoning for this choice is that not all bands of precursor density are being simulated on larger initial conditions due to computational restraints. For example, using the initial condition of $m = 3$ and $n = 3$, would give the total possible precursor configurations to consider $2^{(3+2)^3}$. This has led me to impose the condition that a band of precursor density will not be considered if the number of possible configurations exceeds 500000000. This will in turn reflect on the accuracy of the results obtained as extrapolation will need to be undertaken to produce a complete set in many cases.

The functions combination and combinationUntil were sourced online, and modified to fit the purpose of the code [24]. They work through taking a series of parameters and the combination will use these to generate the remaining parameters for combinationUntil. CombinationUntil is a recursive function that will find all possible unique combinations from a set of values, of a set length. In this case precursor coordinates. Therefore to find all lengths, a combination is called using a for loop that iterates through all possible values. Once a unique set of coordinates of the correct length has been found those coordinates will then be used to find what the successor state will be at time $T+1$. Having found the successor state, results including densities can then be calculated and plotted into the appropriate positions in the results array.

How the successor gird is produced is by first finding out how many non-quiescent neighbors the current cell being inspected in the list of possible successor cells has, whilst simultaneously determining if the cell is in the precursor list. If it can be found in the precursor list that means the cell being inspected is non-quiescent and different rules therefore apply to it. Moore neighborhoods have been extrapolated into n dimensions by considering any cell in which the absolute value of the difference of coordinates for each axis is either 1 or 0. Except all being 0 because then that would be the cell being inspected.

Throughout the process of coding this console application there have been multiple er-

rors throughout the process that have been found and later fixed. One such error went undetected even after generating many complete density maps. The reason being that within smaller boundaries the results were accurate enough that no errors were spotted under human observation during testing. The error itself was related to determining how many non-quiescent cells were within a cells Moore neighbourhood, considering this was a logic error it made it exceptionally difficult to spot that it was occurring and then find the issue and fix it. As a result of this bug all simulations up to this point had to be re-done resulting in multiple weeks of time-loss, due to how long some of the larger starting grids were taking to compute.

4.3 Precursor Density Mapping Algorithm Using Random Samples

By mapping the complete set of all precursor configurations to their successor states and then comparing the densities it runs contrary to the original goal. This is because the only currently known way to prove that a configuration is a Garden of Eden is proof by exhaustion. By utilizing a method that requires the comparison of all possible configurations the algorithm may as well of been finding Gardens of Eden along the way. However this is not an issue for the results I have gained as for Gardens of Eden in two dimensions the smallest known is within a 10 by 10 grid, and it has been definitively proven that no Gardens of Eden can exist within a seven by seven boundary or smaller [Bee22]. To counteract this problem I have created a method of random sampling, as suggested by my supervisor Dr Ged Corob Cook.

Firstly I altered the Precursor Density Mapping algorithm to use a version of random sampling for precursor densities that exceeded the previously set limit. This was to test whether random sampling could produce valid results. The method used was very crude, utilizing a set limit of ten million random samples for each density band that was determined to be too large to compute. This method was tested for a four-by-four grid, and the results given I believe showed promise so I decided to continue to further develop this methodology. Within the acquired results I only found one data point that could be conceivably proven to be incorrect, given previous exhaustive results obtained. 5.

Next, I calculated the minimum number of samples needed to guarantee a 95% chance that at least one of the random samples is within the successor density bound that has the lowest number of precursor configurations mapping to it, for a precursor bound. This was done using binomial cumulative frequency calculations using a Python script that takes the probability of the successor density getting selected as an input. The two sets of data used for this were the two-by-two grid and the three-by-three grid as they have complete data sets and chronological so it would better allow for the determining of patterns, see tables 1 and 2. Additionally given the oddities seen in the results of various dimensions for bounding boxes of size one I excluded these results for the time being. The probability was calculated by hand on a calculator by dividing the smallest number of successor densities within a set of precursor densities and dividing that by the total number of possible precursors of that density, before feeding the results to the Python script to produce results. The probability of a least one positive result using binomial cumulative frequency is then iterated over the total number of combinations for that precursor density until the smallest number of trials that returns a value greater than 0.95 is found. This is then the minimum sample bound that is then recorded. I then plotted the minimum samples against the maximum possible sample size to spot trends within the data and to make a cursory assessment of whether this method would be worth pursuing. If the minimum sample size is within a sufficient close to the possible maximum this method would be useless compared to a more simplistic exhaustive

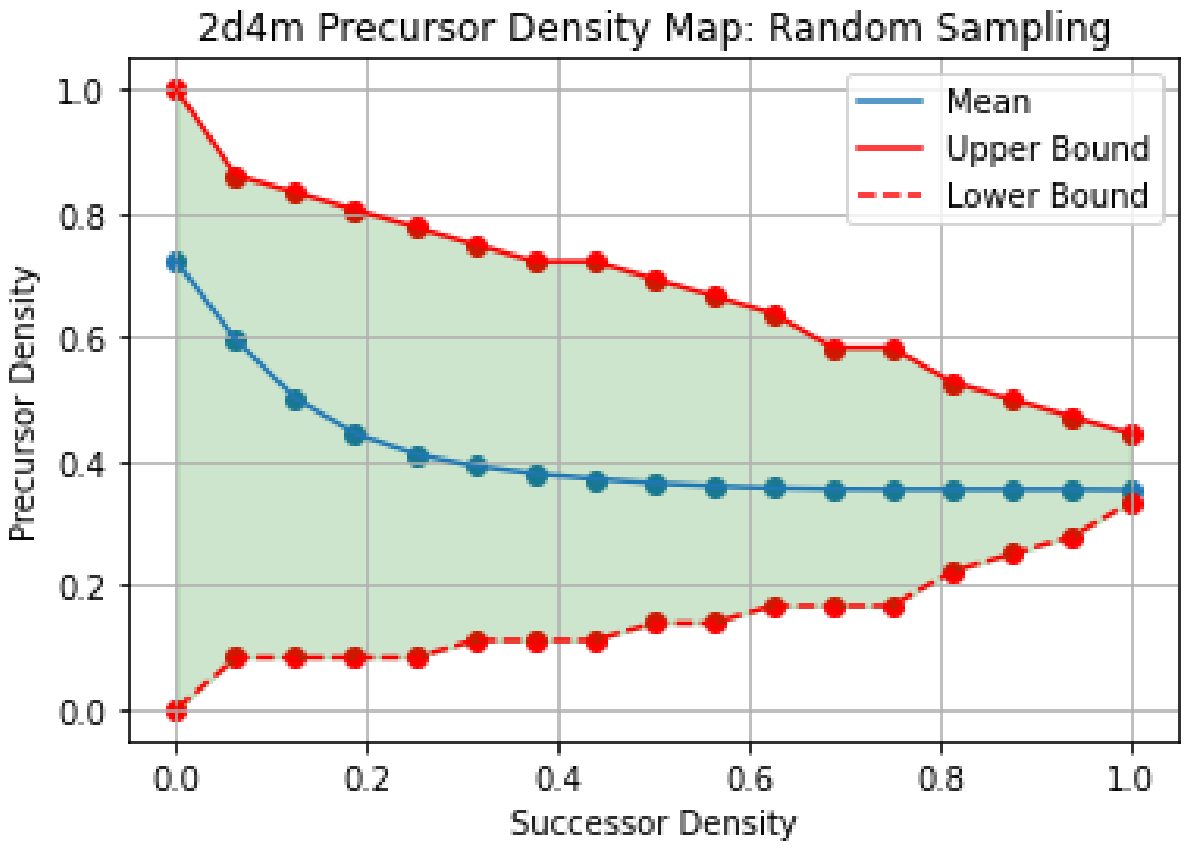

Figure 5: Initial Test for pure random sampling using a arbitrarily chosen number when gathering samples for each bound of precursor density.

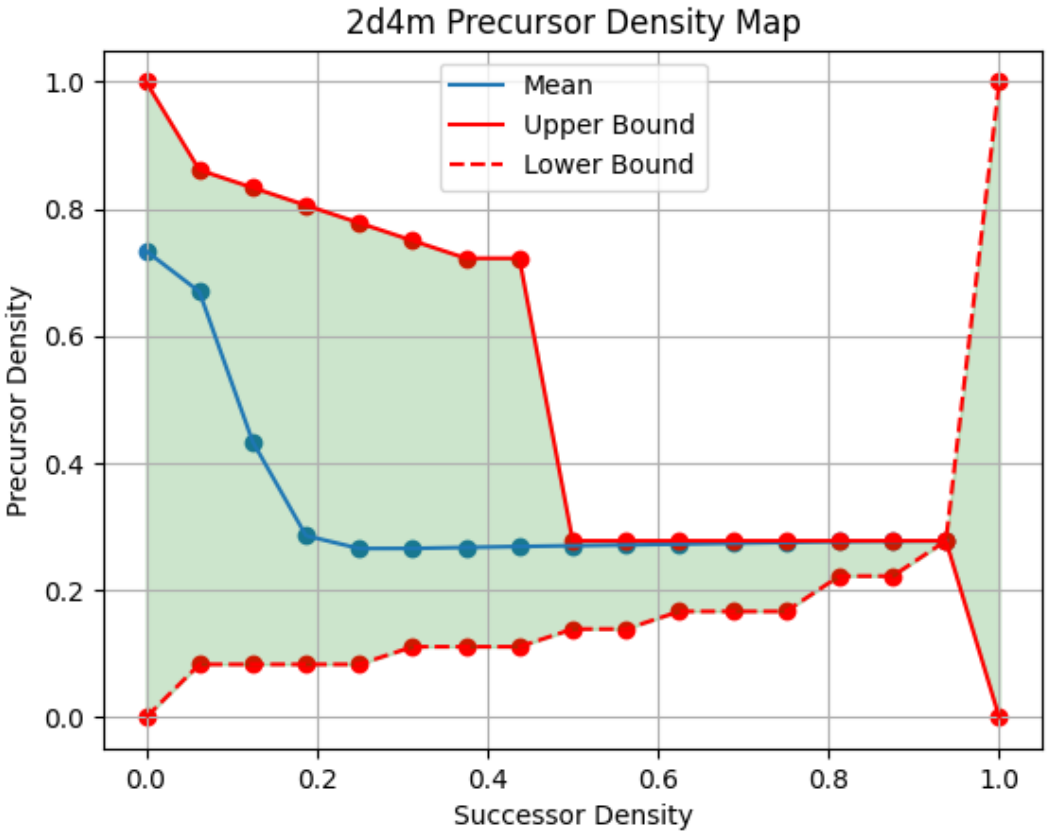

Figure 6: Results Using original Exhaustion method, can be seen that upper bound for densities past approximately 0.45 are completely invalid and unusable. Additionally no data points have been recorded for density 1 resulting in the anomaly at the right hand side of the plot.

search. I have constructed models using both logarithm and normal scalars on the y-axis to better determine any potential trends. See figures 7, 8, 9, 10.

Precursor Density	Number of Samples
0	1
0.0625	1
0.125	1
0.1875	418
0.25	70
0.3125	39
0.375	57
0.4365	213
0.5	2141
0.5625	34
0.625	374
0.6875	57
0.75	1
0.8125	1
0.875	1
0.9375	1
1	1

Table 1: Table used to determine how many samples are needed for each precursor density, for a grid of bounding box size 2 in 2 dimensions, in order to produce accurate results (significance level of 5%)

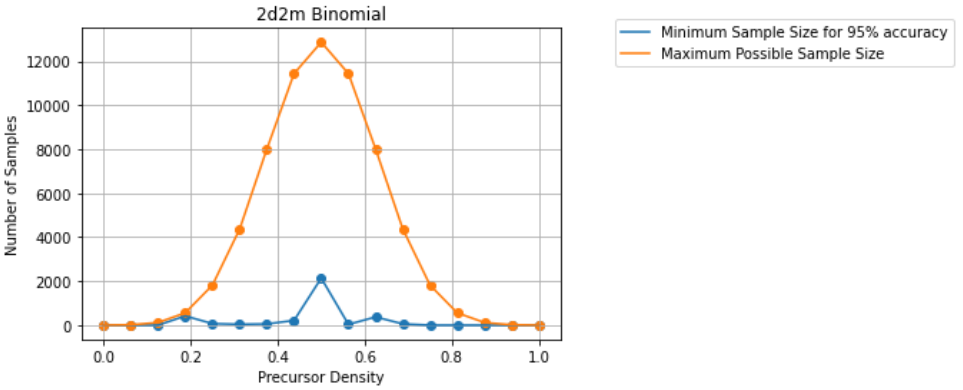

Figure 7: Comparison between the minimum number of samples needed in order to achieve a 95% accuracy with random sampling and the total number of precursor configurations for each precursor density, of a 2 by 2 grid.

Using outliers of the previous results I have created a function that produces results somewhat similar to the minimum sample size curves seen. The idea behind its creation was a curve that returns results closer to the maximum sample size for precursor density values closer to zero and one. However, as density tends to 0.5 the values generated should stray from the maximum. This was achieved by first taking note of outliers within the results to determine the Cartesian coordinates I wish to reproduce using the generated graphing function. This helped me produce a system of equations that could then be solved, resulting

Precursor Density	Number of Samples
0	1
0.04	1
0.08	1
0.12	1147
0.16	9473
0.2	26526
0.24	5766
0.28	3635
0.32	3252
0.36	3449
0.4	4743
0.44	9564
0.48	32727
0.52	229098
0.56	267063
0.6	489615
0.64	23538
0.68	32400
0.72	20000
0.76	2762
0.8	315
0.84	1
0.88	1
0.92	1
0.96	1
1	1

Table 2: Table used to determine how many samples are needed for each precursor density, for a grid of bounding box size 3 in 2 dimensions, in order to produce accurate results (significance level of 5%)

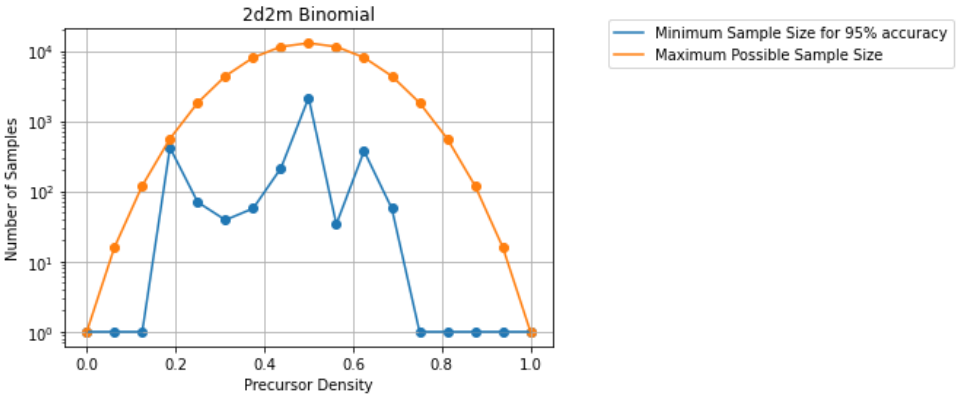

Figure 8: Comparison between the minimum number of samples needed in order to achieve a 95% accuracy with random sampling and the total number of precursor configurations for each precursor density, of a 2 by 2 grid and using a logarithmic scale on the y axis.

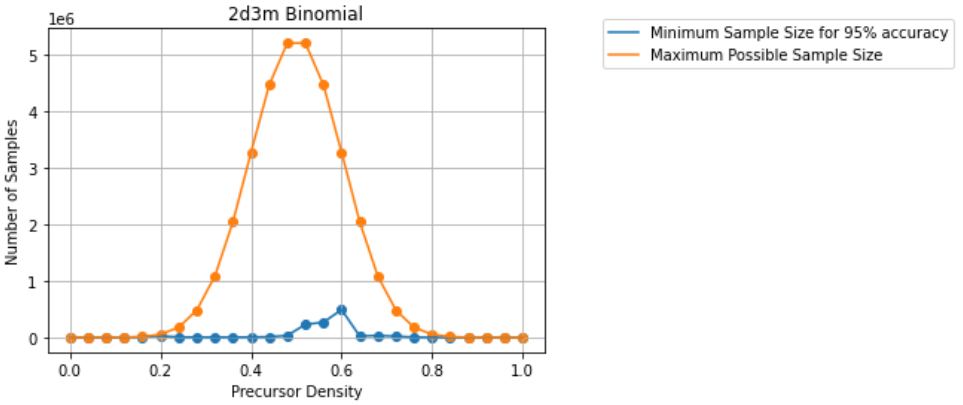

Figure 9: Comparison between the minimum number of samples needed in order to achieve a 95% accuracy with random sampling and the total number of precursor configurations for each precursor density, of a 3 by 3 grid.

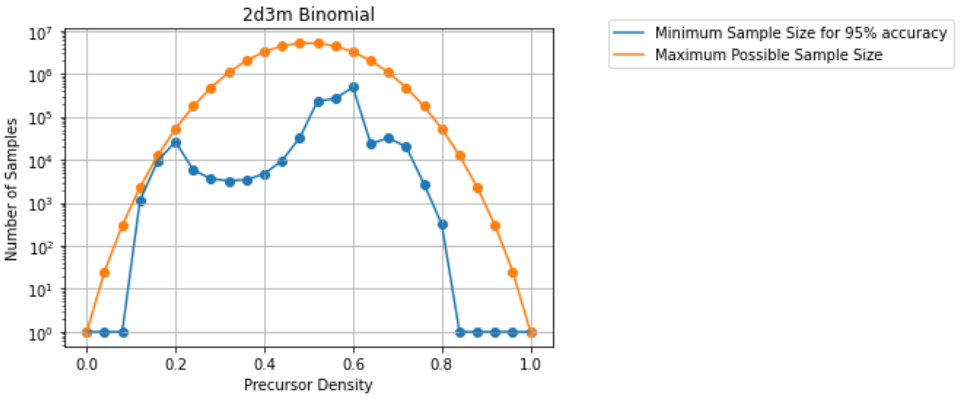

Figure 10: Comparison between the minimum number of samples needed in order to achieve a 95% accuracy with random sampling and the total number of precursor configurations for each precursor density, of a 3 by 3 grid and using a logarithmic scale on the y axis.

in a quadratic that generates values between 0 and 0.8 when given values between 0 and 0.5. To compensate for the fact the density values range from 0 to 1 any density of greater than 0.5 is taken away from 1 to mirror the first half of the equation. The value generated is then taken away from one to produce a multiplier that is used with the maximum sample size to create an estimate of what the sample size of a bound should be. Results generated using the sample size generated using this method will not be completely accurate because whilst larger than the minimum sample all that denotes is that the sample generated will have a probability of greater than 95% to produce valid results. To ensure 100% accuracy all configurations would have to be considered which as previously mentioned would invalidate this method as an exhaustive approach would allow for direct searching of Gardens of Eden. These results can be seen in the figures 11, 12, 13, 14.

The outliers chosen to produce this function where that when precursor density is 0 the output should be 0 to produce a multiplier of 1. This is because there is only 1 possible configuration at densities 0 and 1 for precursor densities. Another outlier chosen was at density 0.16 for a 2-dimensional space using a bounding area of 3 needing a multiplier of 0.75 (rounded up), in addition to a density of 0.5 at density 2 and a bounding area of 2 needs

a minimum of 0.2 multiplier (rounded up). Considering that the result of the equation is being taken away from 1 the actual outputs of the equation are 0.25 and 0.8 respectively. As previously stated these values were then put into a system of three linear equations and solved to produce the following function, where x represents the precursor density.

$$multiplier = 1 - (\frac{15}{136}x^2 + \frac{2101}{1360}x)$$

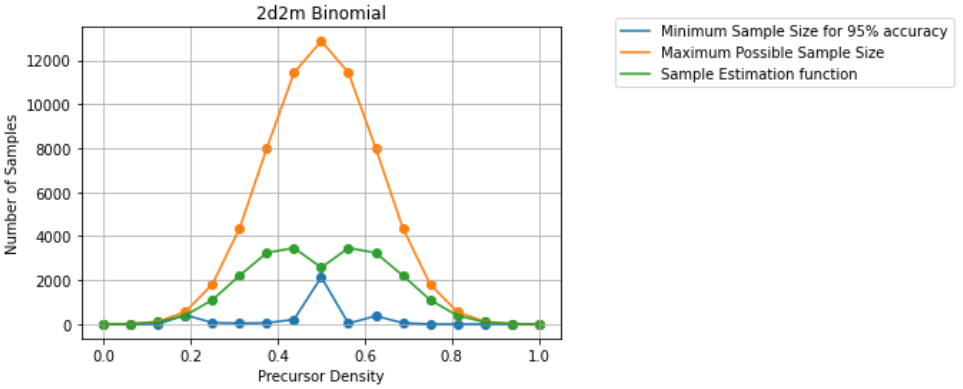

Figure 11: Comparison between the minimum number of samples needed in order to achieve a 95% accuracy, random sampling and the sample estimate. Results are of a 2 by 2 grid.

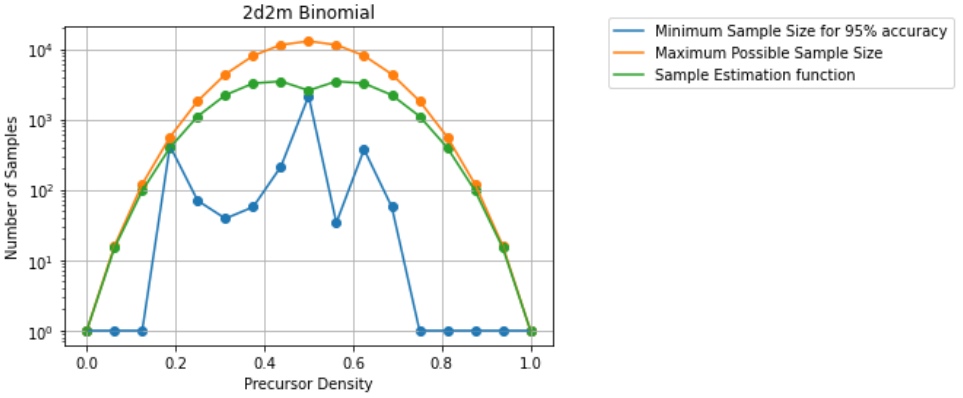

Figure 12: Comparison between the minimum number of samples needed in order to achieve a 95% accuracy, random sampling and the sample estimate. Results are of a 2 by 2 grid and are plotted on a logarithmic scale on the y axis

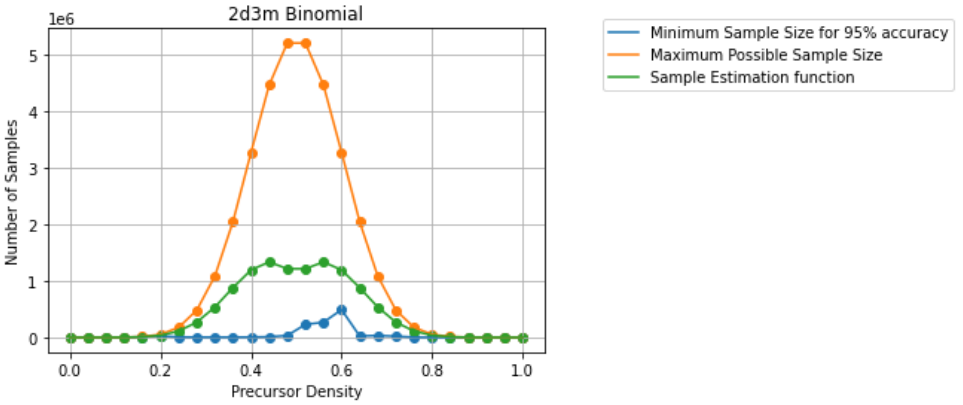

Figure 13: Comparison between the minimum number of samples needed in order to achieve a 95% accuracy, random sampling and the sample estimate. Results are of a 3 by 3 grid.

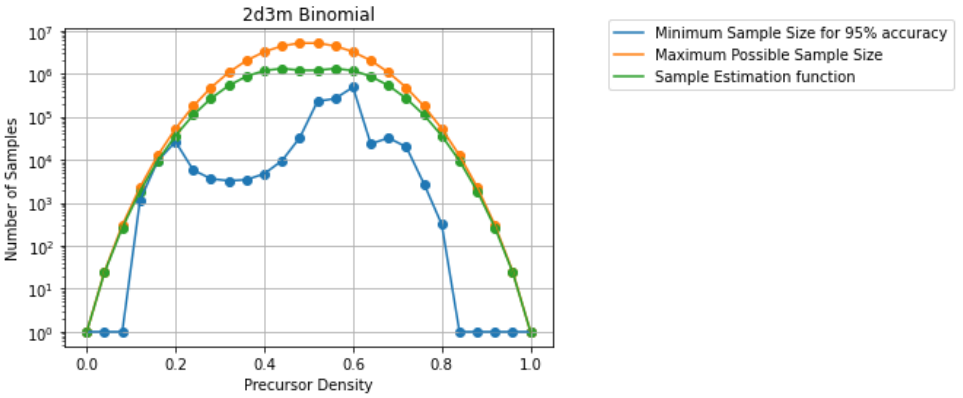

Figure 14: Comparison between the minimum number of samples needed in order to achieve a 95% accuracy, random sampling and the sample estimate. Results are of a 3 by 3 grid and are plotted on a logarithmic scale on the y axis.

5 Results

The naming convention for all figures shown is $n_1 d n_2 m$ which stands for the figure representing n_1 dimensional space for a bounding area n_2 for the successor state. For example figure 15 represents data about a 2 dimensional space for a bounding area of 1 for the successor state. Alternatively a one-by-one space. In this first section of results, I will be looking at figures 15 through 34, as these are the initial results obtained through the exhaustive method used. Any oddities observed will be explained in greater detail later in the results section and are likely due to the exclusion of data due to time constraints.

A great starting point for the results is figure 15. The lower red dotted line represents the lower bound of precursor density that maps to the corresponding successor density. The solid red line on the upper section of the figure is its counterpart, for the upper bound. These two lines are the data points of inquiry in that they can be used to exclude density bounds of precursor configurations from searches for Gardens of Eden. For example, if a person wished to search for a Garden of Eden of density 1 in a two-dimensional one-by-one bounding area, this data would let them know that they would only need to consider precursor configurations between $0.\overline{3}$ and $0.\overline{4}$. However, there is no such Garden of Eden that exists as Randall

D. Beer has proven that no Garden of Eden exists in seven-by-seven boundaries or smaller [Bee22], the previous statement serving as an example of a use case. The mean does not hold any large significance but is a point of potential interest and is useful when compared to random sampling methods to ensure randomness, as a random sample should create an equivalent mean. The translucent green on the figures is purely a stylistic choice to better represent the area between the upper and lower bounds.

As to why figure 15 is a great starting point, given the relatively few data points it gives a great representation of the general trends seen throughout the data set. These trends are lower successor densities having a much wider range of possible precursor densities and a general narrowing of this range as successor density increases. Coupled with this is the tendency for the upper bound absolute value of its gradient to be greater than the lower bound. This appears to be linked with a negative gradient seen with the mean. However, the actual results seen on this graph are hard to interpret due to the lack of them. Additionally, as the two configurations possible at the successor state are the two base components of the Game of Life it stands within reason to assert that no matter how large the dimension a bounding area of one will never be a garden of Eden.

The exact patterns can be observed seen in 15 can be observed in figures 25 and 31. The bounds maintain the same values for the successor density of zero and only differ for the successor density of one. Whilst the starting mean does differ slightly between the three figures, a greater difference can be observed in the ending mean value. In turn, this produces a steeper gradient for the mean, increasing as the dimensions increase. The difference in ending mean value between dimensions of two and three is much greater than the difference between three and four leading credence to the idea that this value may tend towards a 'true' value as the number of dimensions increases. Additionally, this trend can also be observed for both bounds, in conjunction with the range between the upper and lower bound at successor density one shrinking as the dimensions increase.

Continuing to look at two dimensions, for a bounding area of two, figure 17 continues the same trends highlighted when observing 15. As there are more data points on the x-axis to observe, the mean can be seen to decrease in an exponential manner appearing to tend to an asymptotic point. This point is relatively close to, but slightly lower than the endpoint for the mean for a bounding area of one. Due to the incompleteness of my data for larger bounding and dimensional areas, any patterns for the true value of these asymptotic points, and how bounding area and dimension affect them is purely speculative. As well as any potential value in determining this. However, trends that can be speculated about is that whilst starting conditions that produce a larger number of possible precursor combinations result in a lower ending value for the mean, this is not a guarantee. As can be seen by a bounding area of three and dimension of two as seen in figure 19 is larger than that seen in figure 17.

Another point of note about figure 17 is the peaks and troughs seen in the upper and lower mean. This is reminiscent of the behavior noted in Randall D. Beer's study [Bee22]. This coincidence was of note to me as whilst Beer's data was about the number of precursor configurations against the successor density, the data in question here is precursor density against successor density. Given the similarities I believe there may be some link between the two data sets that could explain this coincidence, however, this trend is not apparent in other figures. Upper bounds in other figures show a more continual decrease compared to figure 17, whilst lower bounds continue to show the observed trend of fluctuating peaks and troughs.

In two dimensions using a bounding area of three, trends seen in figure 17 can be seen

to largely continue. Upper and lower bounds retain their respective gradients within expected limits, however do not continue the trend of peaks and troughs. Additionally, the lower bound, in this case, is likely the weakest example of this behavior of any complete and incomplete data gathered. In turn, this has led me to question if the behavior seen in figure 17 is just a coincidence as all other examples of it in lower bounds to a distinct degree are of incomplete data sets. Another point of note is the continued smoothing of the mean as more fidelity becomes available to represent it, becoming a much clearer example of an exponential curve.

Figures 21 and 23 are the first examples of incomplete data being discussed in this section so far. The most obvious indicator of this is the clear anomaly in the upper bound. Considering that precursor densities that contained more than five hundred million unique configurations were excluded in combination with the binomial distribution a number of configurations compared with density, only the tail ends of precursor configurations are being considered. Consequently, the result is a sheer drop in which only the lower-density precursors could produce successors of a specific density. This is in line with observed trends of higher-density successor configurations occurring more frequently with lower-density precursors. The overcrowding rule results in higher-density precursors only mapping to low-density successors more frequently. To put it into more succinct terms, the lower bound for precursor density to produce a non-zero successor density is closer to zero than upper bound to produce a non zero successor is to one. Hence the incompleteness of data has the appearance of having a greater impact on the validity of the upper bound. This trend can be seen to be mirrored on both 21 and 23, but to a larger degree on the latter. The mean can also be seen to be affected through its deformation from the observed pattern of an exponential curve in figure 21, however maintains the shape of an exponential in figure 23. On the other hand, both are greatly different from the exponential curves observed in figures 17 and 19. They both tend to an asymptotic point and stabilize unlike behavior observed in figures with complete data sets. I believe this gives credence to my previous theory of an asymptotic point eventually being reached in larger datasets with a larger number of precursor configurations, however using incomplete data sets as proof is not in any measure substantial. Furthermore, another key feature of the incomplete data sets is the inversion of the upper and lower bounds. When initializing the arrays in Python to store data the lower bound is initially set to one, and the upper zero. This is so that when a value is recorded in that index it can be compared using a comparison operator and changed accordingly. As a result of this, any successor densities that have no recorded precursors mapping to them have caused the lower bound and upper bound to invert their positions. In retrospect, a more reliable method would have been to have the initial values in the array be null and have a clause for checking if the currently inspected index in an array is a null value, instead of presetting values to then only need one check to be executed.

Similar to figures 21 and 23, figures 27, 29 and 33 are examples of incomplete data sets. They become increasingly unreadable and incoherent as the number of precursor configurations increases due to the relative number calculated decreasing, concerning the total. The effect of this can be seen to be an increasingly steep initial gradient for the mean, as scale increases, before it plateaus. As bounding area and dimensions increase, the point of plateau also decreases. Furthermore, the range between upper and lower bound can be seen to decrease and remain plateau similarly to the mean. For figure 33, this range is small enough that the two lines are indistinguishable. Due to the extremely small portion of total precursor configurations that were calculated for these figures, I believe it to be in my best interest to largely disregard them in any conclusions drawn for this study.

Figure 16 it represents successor density against the number of precursors for each successor density bound. Each line on the graph represents a different precursor density, and their opacity represents the precursor density. The more opaque the line the greater the precursor density. A choice was also made to display the results in a logarithmic format due to results otherwise a large portion of results were congregated into the lower section of the graph in an unreadable manner. This graph in particular displays that the vast majority of precursor density configurations only map to a single successor density value. The only value of precursor density to break this condition is the density of $0.\overline{4}$. I believe that this graph differs from others made in the same style due to the binary output of the successor densities. Other graphs produced much more diverse and recognizable patterns. The exceptions to this are figures 26 and 32, displaying similar patterns to figure 16. I again attribute this to the binary nature of the output resulting in patterns similar to the other figures not presenting themselves.

For figures 18, 20 patterns can be seen to emerge. These can be harder to distinguish in figure 18, likely due to lack of fidelity, however, this can be seen in figure 20. This is a clear distinction between the two groups of precursor density. Lower density largely shows an initial positive gradient, and as successor density increases this gradient decreases, eventually tending to the negatives. For densities of a certain size, this trend changes to an initial negative gradient that further decreases with successor density. There are outliers to this pattern, however, with the vast majority of those being of sufficiently large or small size and in turn only producing successor states of density zero. I believe this to be a great way of visualizing the trends I had noted earlier of lower densities tending to produce a greater number of higher-density successor states. Another key point of note is that in all cases the turning point for which an initial positive becomes an initial negative is below the 0.5 point for precursor density, as to why this is I am not sure, however, it likely has a connection with the overcrowding rule.

Despite missing large portions of data figures 22 and 24 serve to represent an exaggerated version of the trends highlighted. The difference between the two distinguished groups is greater, due to the missing precursor densities that were excluded due to containing an excessive number of configurations to calculate. The remaining figures (28,30,34) highlight the limitations of the design choices for this figure whilst also furthering my point of the unreliability of the incomplete data sets. Firstly, of the few precursor densities calculated compared to the total number the initial few are of such low density that they are barely visible on the graph and in some cases unnoticeable. In turn, the remaining high-density precursor only maps to the successor density of zero. Other design choices were considered but this one gave the best results for the complete data sets so it was chosen as the final design choice. The illegibility of the larger densities and bounding areas is of less importance due to their incompleteness.

Using the results gathered a model for generating results using random sampling was then created. The results for a single implementation of this random sampling method can be seen in the figures 35 and 36. Due to the relative accuracy of the method, the figures are difficult to distinguish from their exhaustive counterparts. In the case of the upper and lower bounds, all data points are accurate. However, deviations can be found when observing the mean. These slight differences can be ascribed to the random sampling used, and slight deviations can be expected. Additionally, the random sampling method was run one hundred times for each bounding area. The results showed that in two dimensions for a bounding area of two each data point on the lower bound has a 99.2% chance of being accurate whilst the upper bound has a 99.6% chance of being correct. Additionally in two dimensions for a

bounding area of three the lower bound has a 99.4% chance of being correct (to 3sf) and the upper bound has a 100% chance of being correct, for each data point. As expected the upper boundary has a greater chance of being correct than the lower, this is because the function used to determine the number of precursor configurations that need to be randomly chosen is a mirror around the density 0.5. In turn, this means that whilst upper and lower densities are treated the same in terms of the number of precursors considered, the lower half tends to have a greater spread of potential successor densities, as previously highlighted in observed trends. Furthermore, despite the upper bound accuracy of a bounding area of three in two dimensions being 100%, it does not mean it is completely accurate. Additional testing would have to be done to determine an exact value for accuracy, 100 simulations not being inductive to conclusive results in my opinion. However, it is interesting to note the increase in accuracy with the increase in bounding area. It leads me to believe that whilst the function I used, for random sampling, could be improved upon one potential method to do so would be to include the dimension and bounding with the function. On the other hand, to properly implement this would require a much larger and more conclusive dataset.

A potential method for using these results would be in conjunction with other methods for finding Gardens of Eden. For example, once a prospective Garden of Eden has been identified, a graph can be generated using the methods I have used to identify the range of densities for which a precursor of the potential Garden of Eden would be if it exists. The goal is to have a method that allows for narrowing the search for Gardens of Eden, as the only known way to prove that a configuration is one, is through exhaustion. This will be achieved if the total time taken to define an upper and lower limit for a bounding area of a certain dimension in addition to the time taken to compute all configurations within the defined range of densities is less than the time to compute the entire map of precursor to successor configurations. However, the upper and lower limits for precursor density will only have to be calculated once for each specific density and bounding area, resulting in greater efficiency in this method the more it is used. On the other hand, if a range of densities has to be calculated every time a potential Garden of Eden is being considered it will eventually be more efficient to have considered the entire parcel of configurations. In turn, finding all Gardens of Eden for that specific dimension and bounding area.

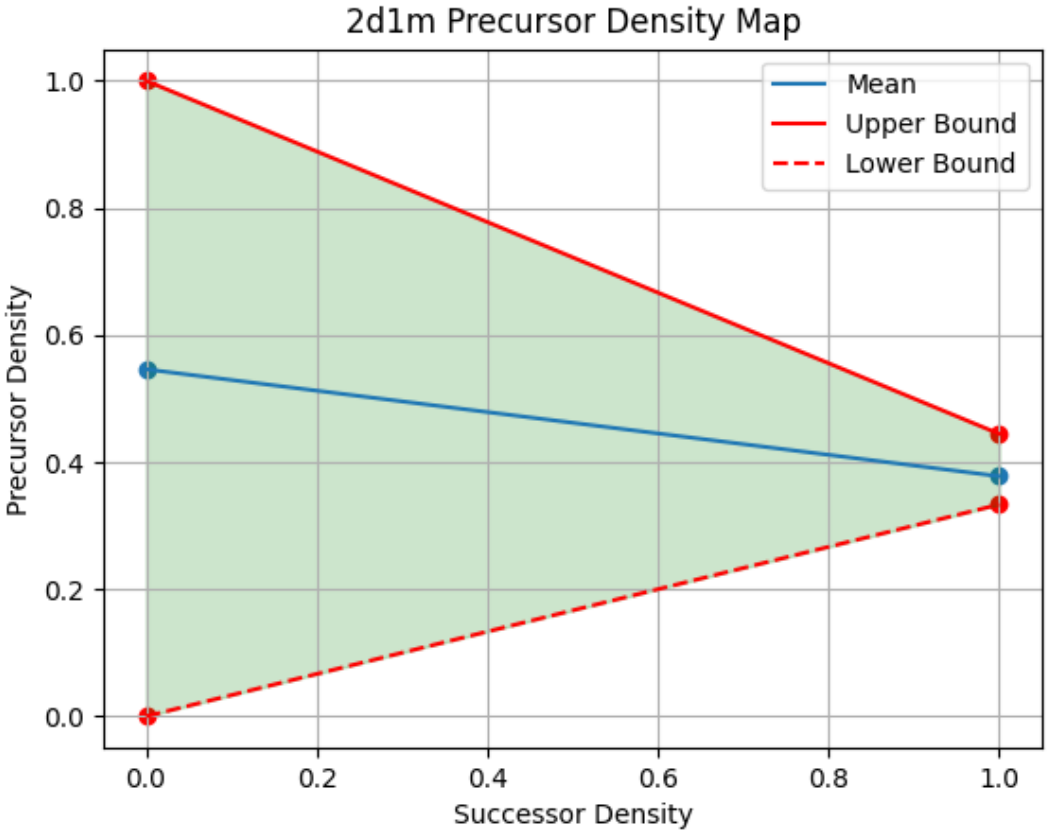

Figure 15: Precursor density mapped against successor density for a 2 dimensional grid with a bounding area of 1.

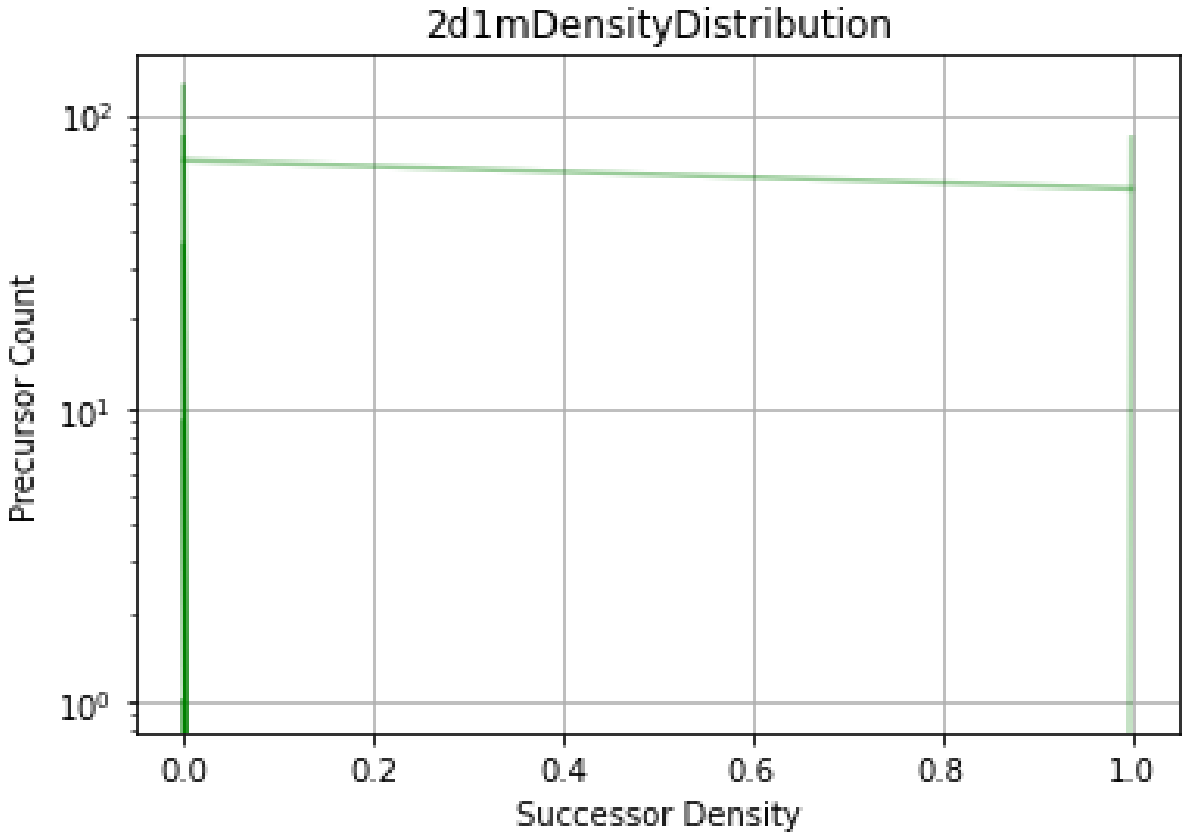

Figure 16: Precursor count mapped against successor density for a 2 dimensional grid with a bounding area of 1. Each line is representative of a density of precursor, and the opacity of the line is reflective of that density.

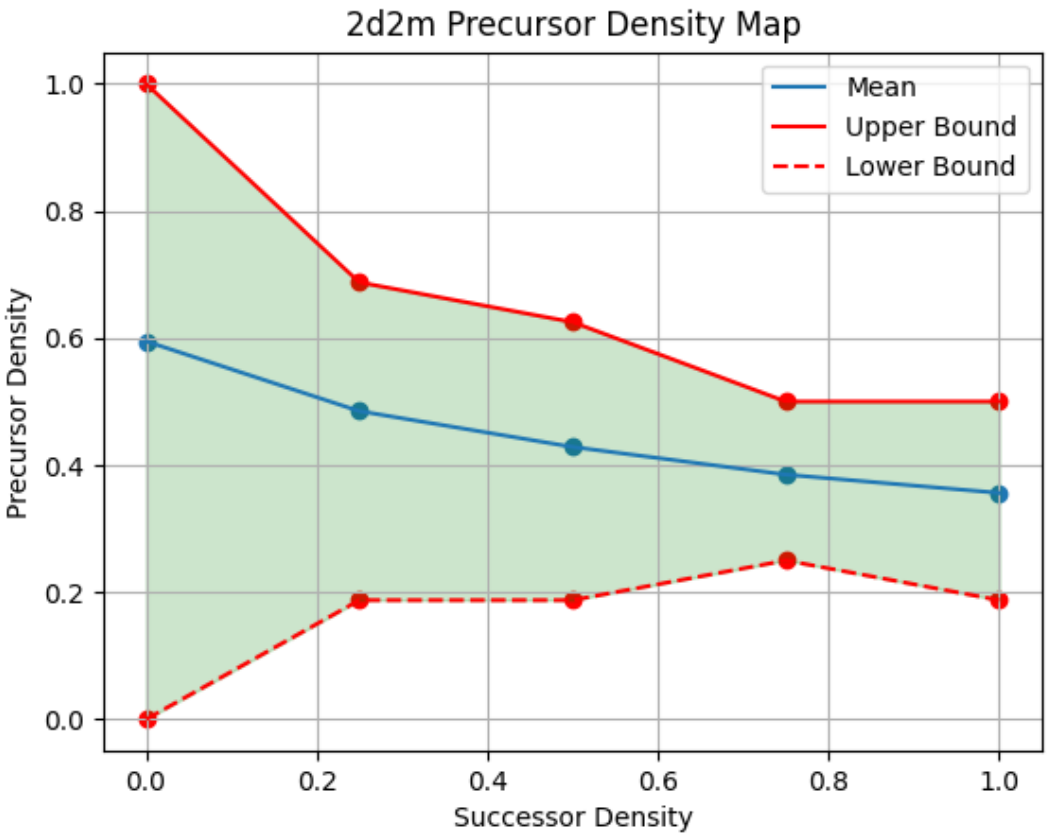

Figure 17: Precursor density mapped against successor density for a 2 dimensional grid with a bounding area of 2.

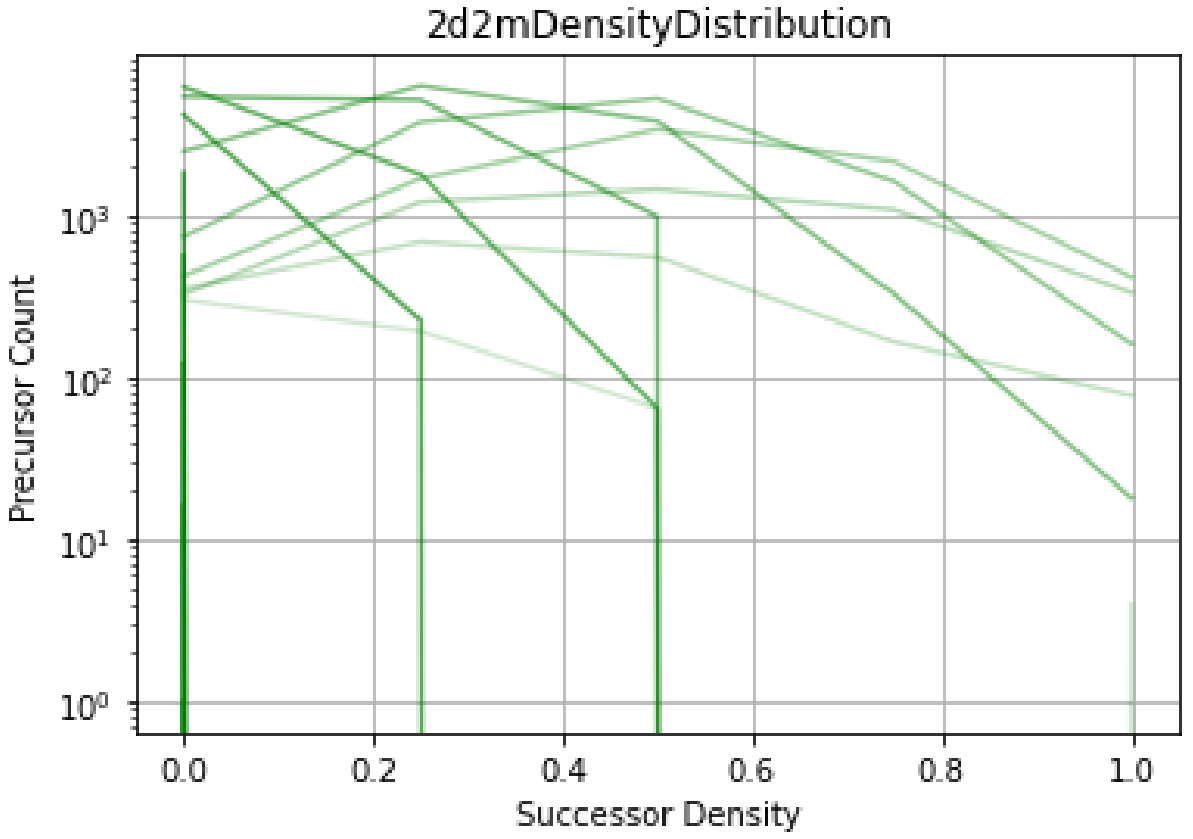

Figure 18: Precursor count mapped against successor density for a 2 dimensional grid with a bounding area of 2. Each line is representative of a density of precursor, and the opacity of the line is reflective of that density.

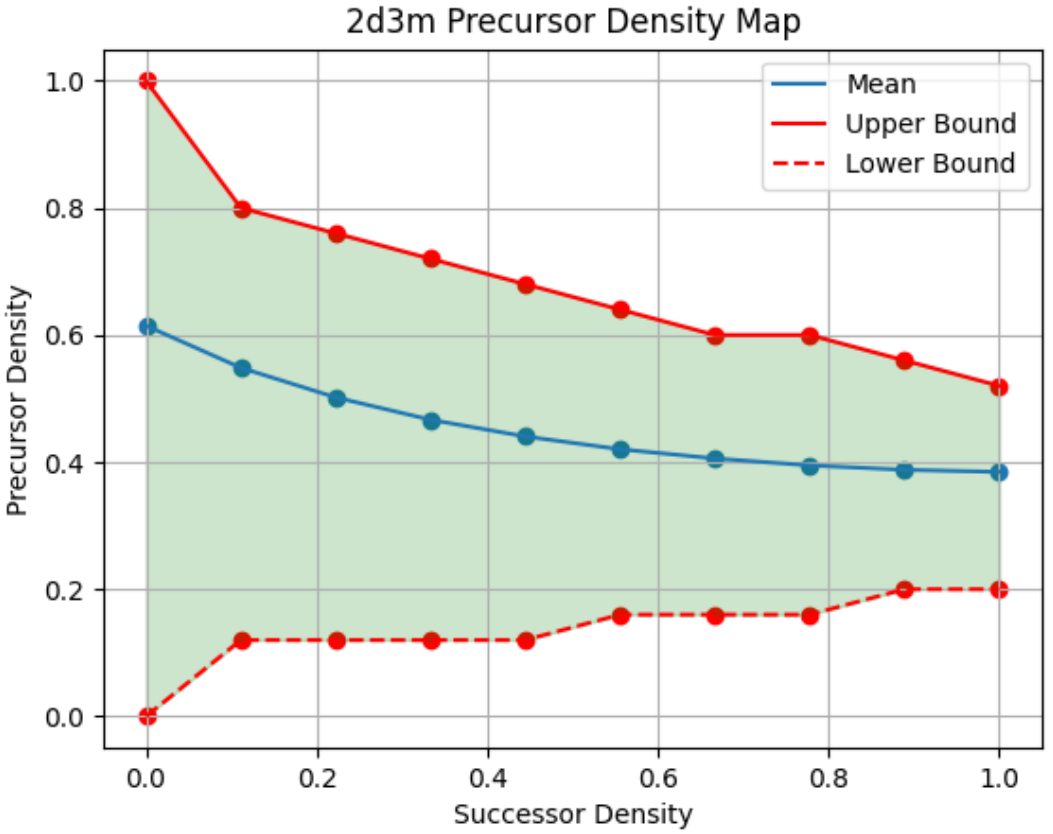

Figure 19: Precursor density mapped against successor density for a 2 dimensional grid with a bounding area of 3.

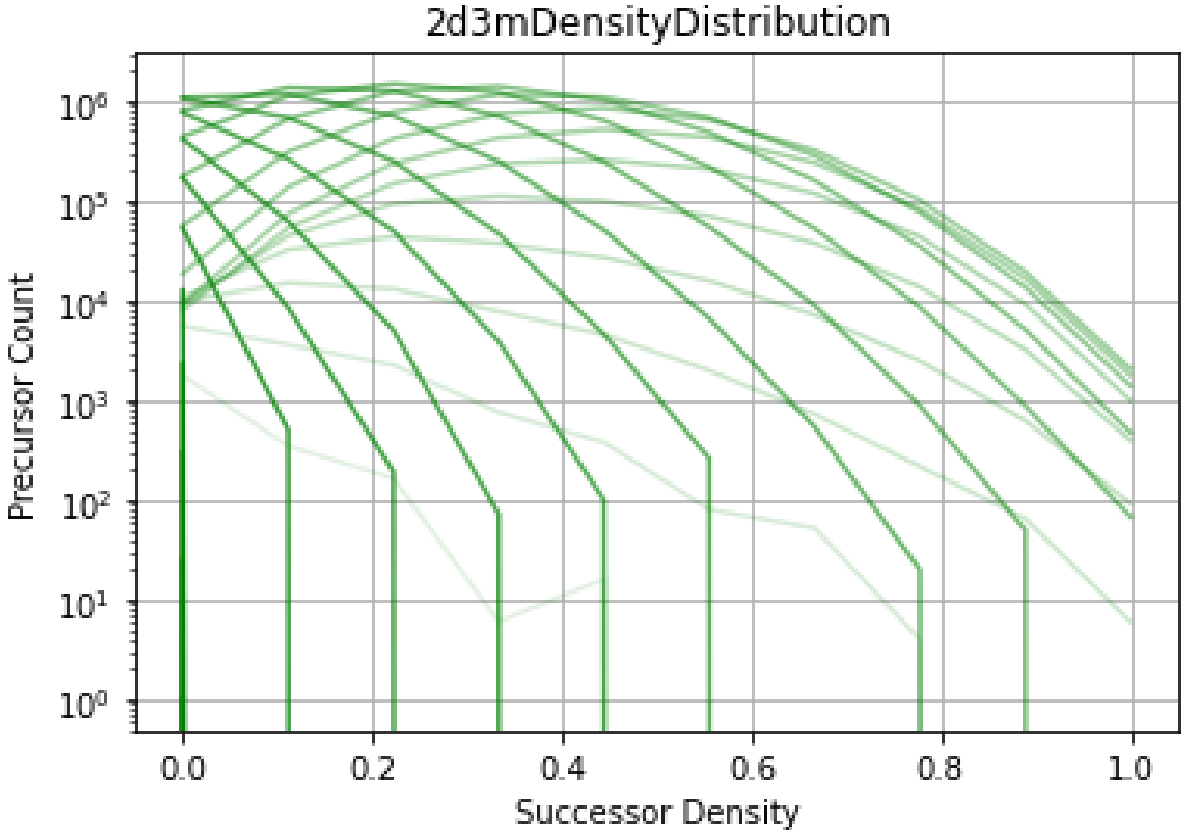

Figure 20: Precursor count mapped against successor density for a 2 dimensional grid with a bounding area of 3. Each line is representative of a density of precursor, and the opacity of the line is reflective of that density.

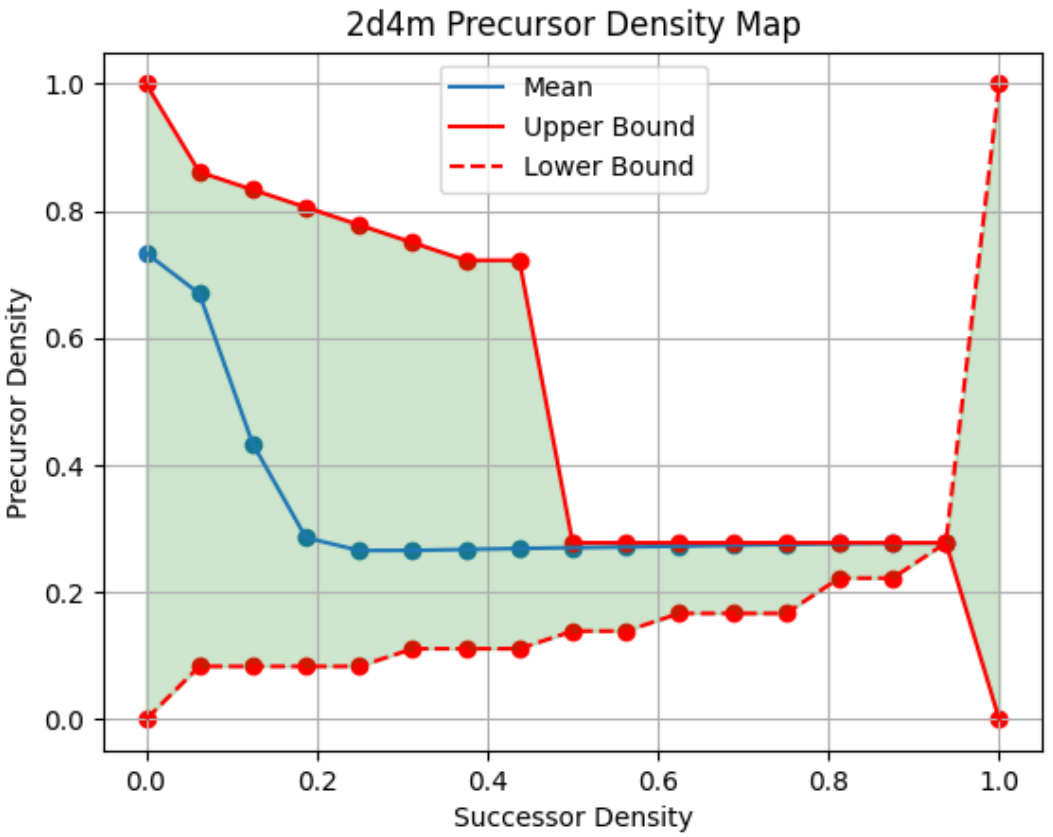

Figure 21: Precursor density mapped against successor density for a 2 dimensional grid with a bounding area of 4.

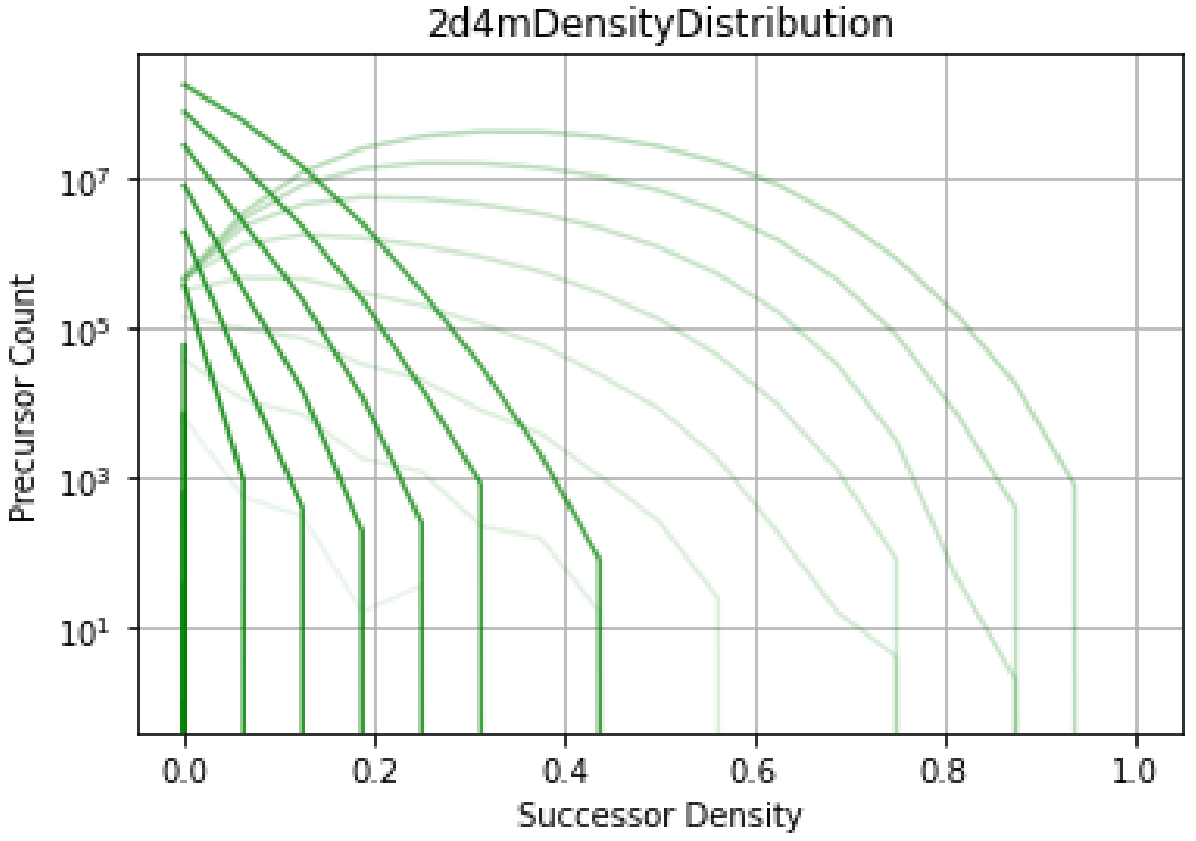

Figure 22: Precursor count mapped against successor density for a 2 dimensional grid with a bounding area of 4. Each line is representative of a density of precursor, and the opacity of the line is reflective of that density.

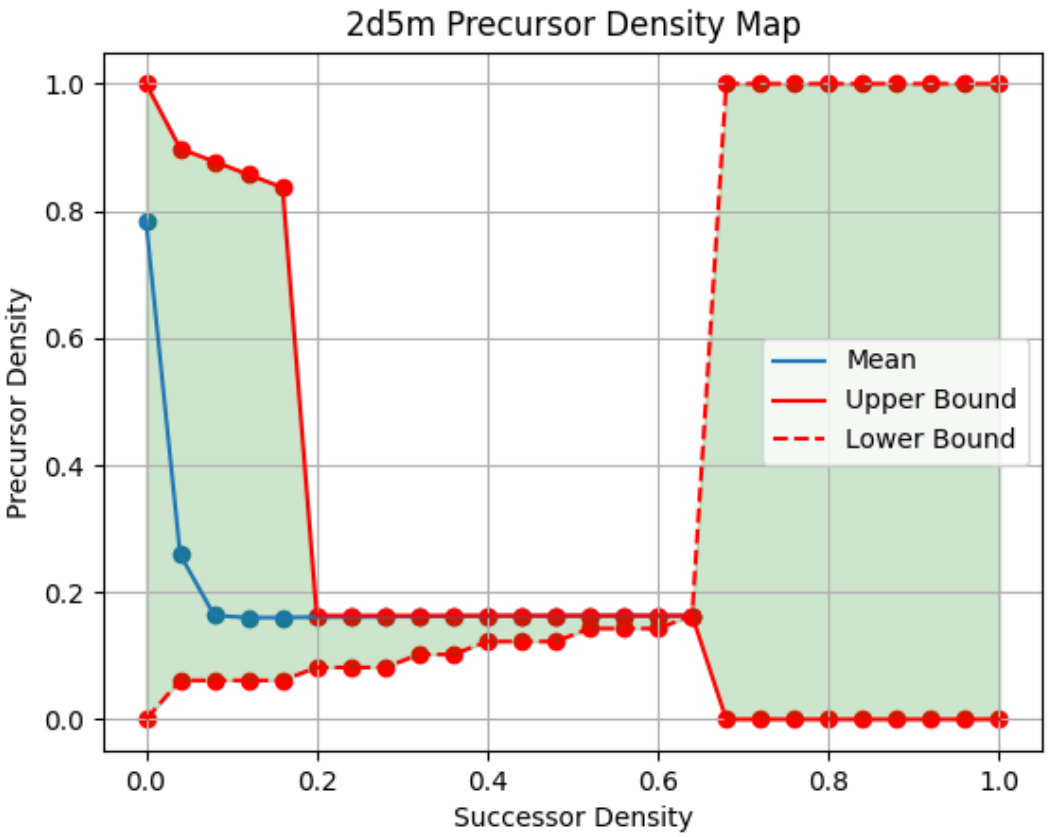

Figure 23: Precursor density mapped against successor density for a 2 dimensional grid with a bounding area of 5.

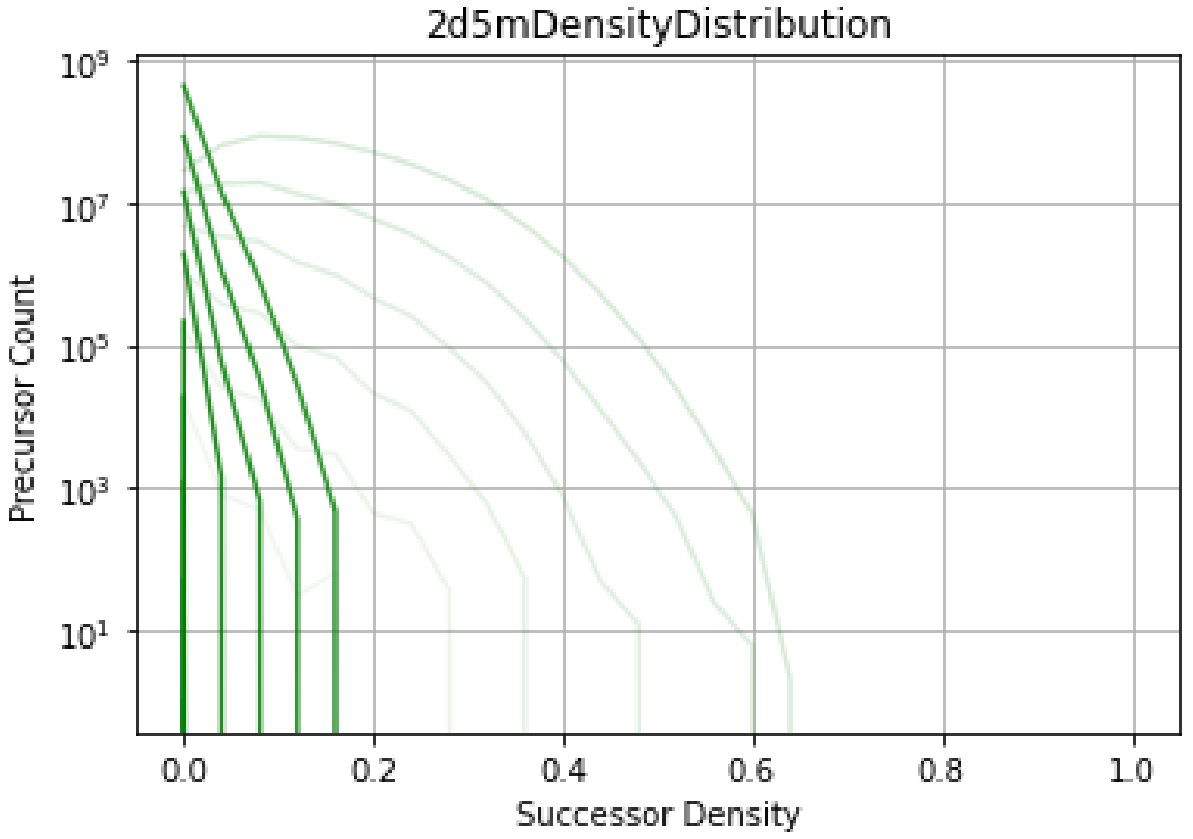

Figure 24: Precursor count mapped against successor density for a 2 dimensional grid with a bounding area of 5. Each line is representative of a density of precursor, and the opacity of the line is reflective of that density.

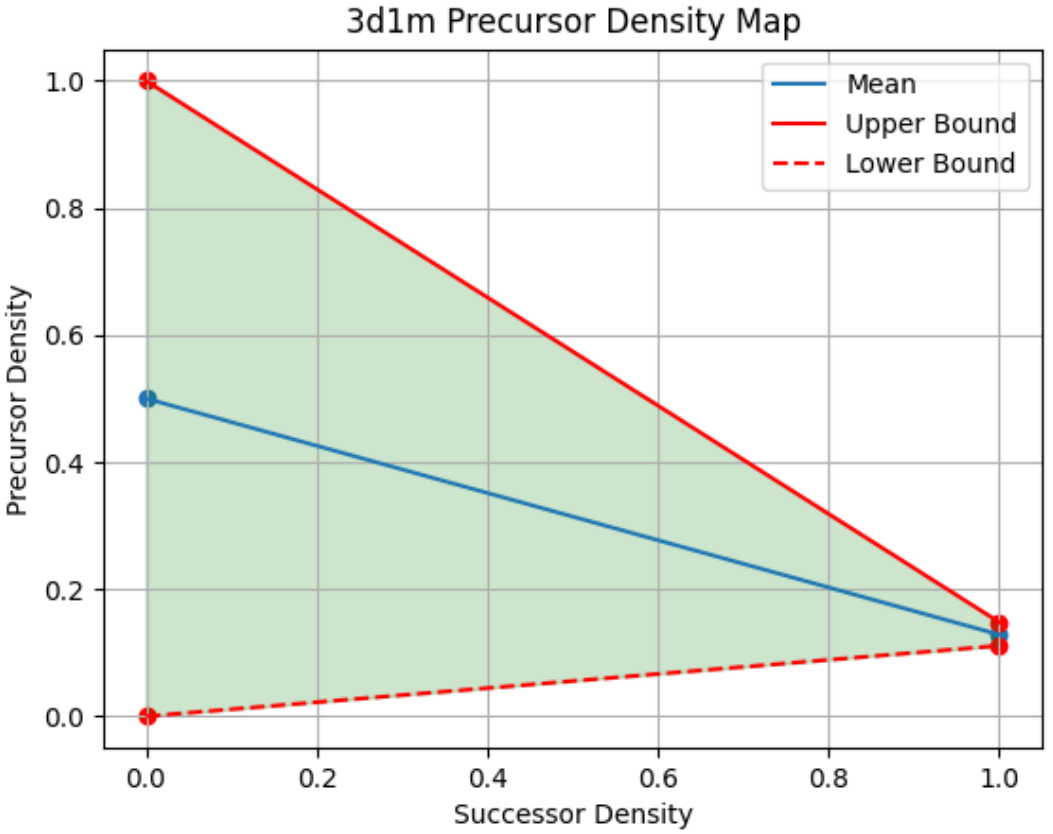

Figure 25: Precursor density mapped against successor density for a 3 dimensional grid with a bounding area of 1.

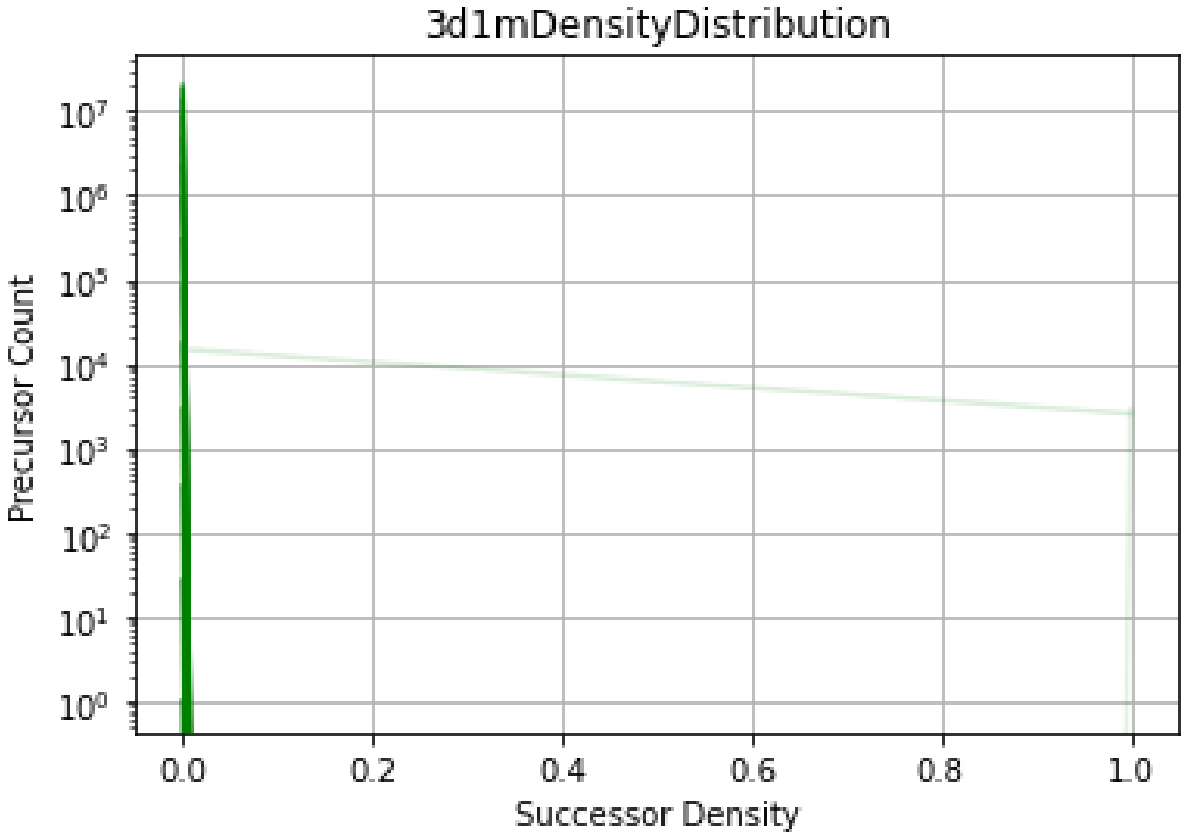

Figure 26: Precursor count mapped against successor density for a 3 dimensional grid with a bounding area of 1. Each line is representative of a density of precursor, and the opacity of the line is reflective of that density.

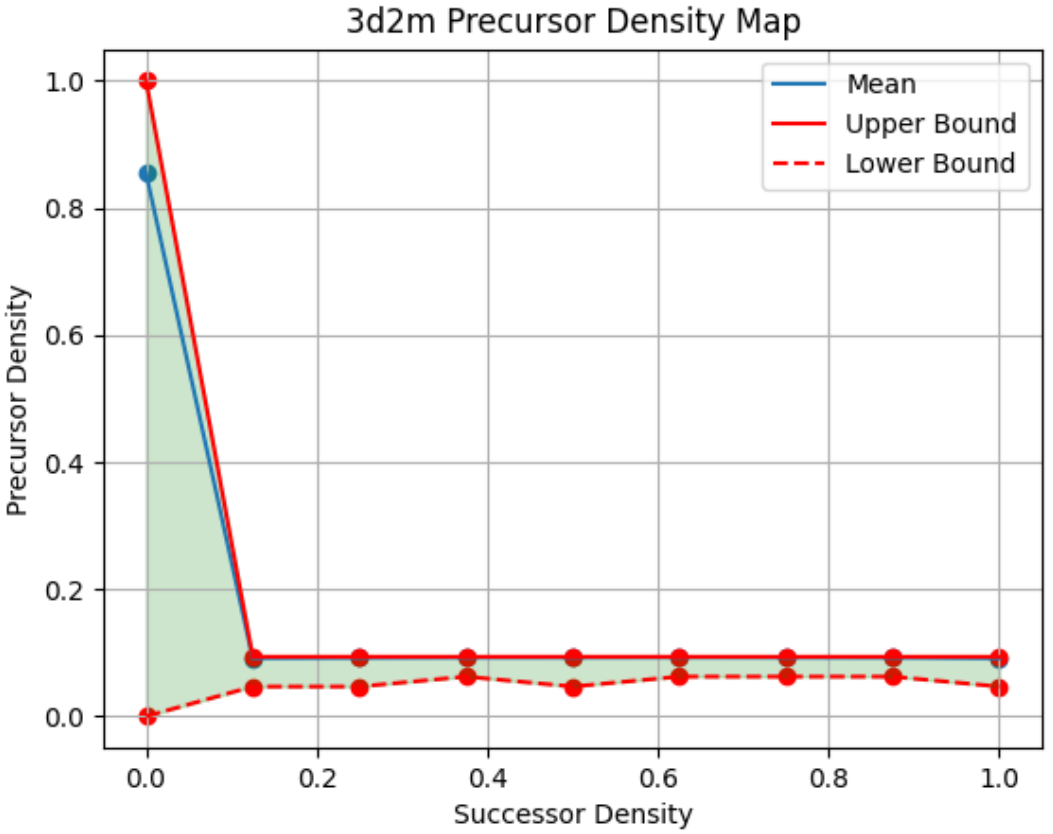

Figure 27: Precursor density mapped against successor density for a 3 dimensional grid with a bounding area of 2.

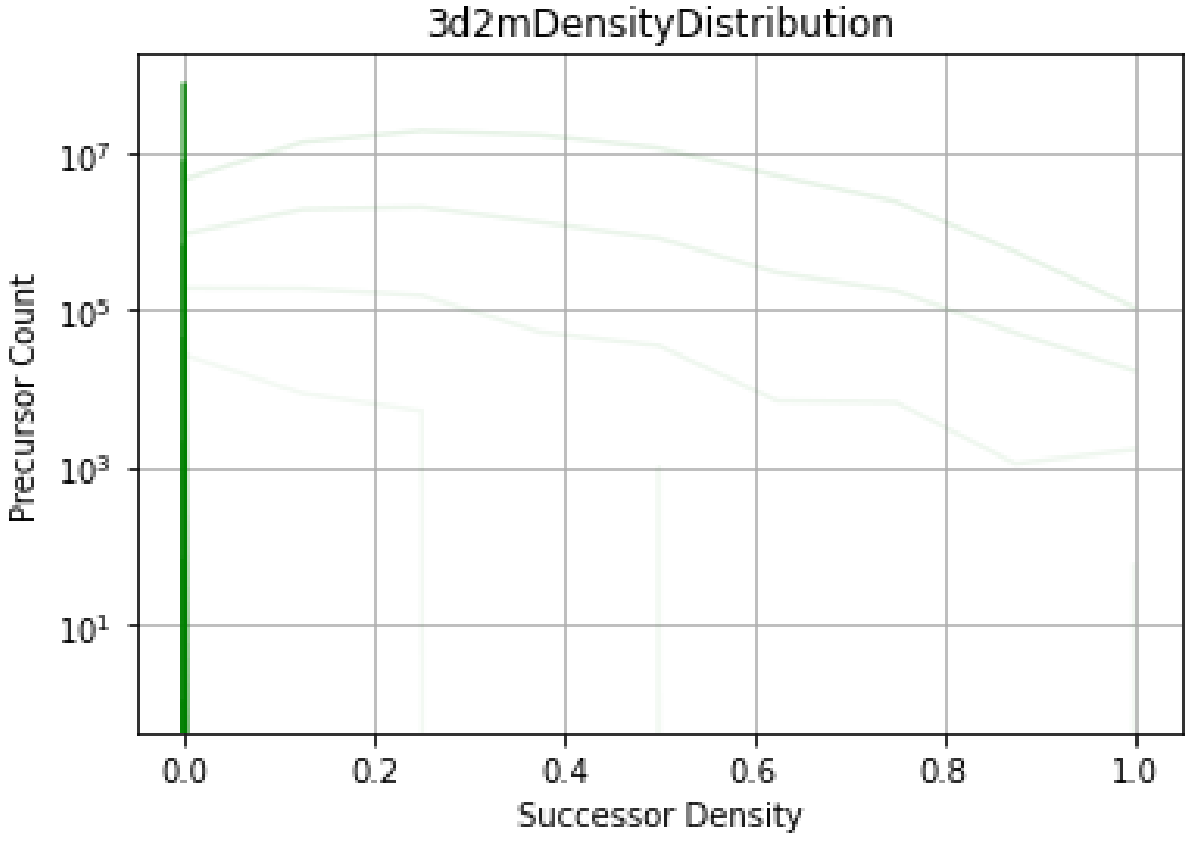

Figure 28: Precursor count mapped against successor density for a 3 dimensional grid with a bounding area of 2. Each line is representative of a density of precursor, and the opacity of the line is reflective of that density.

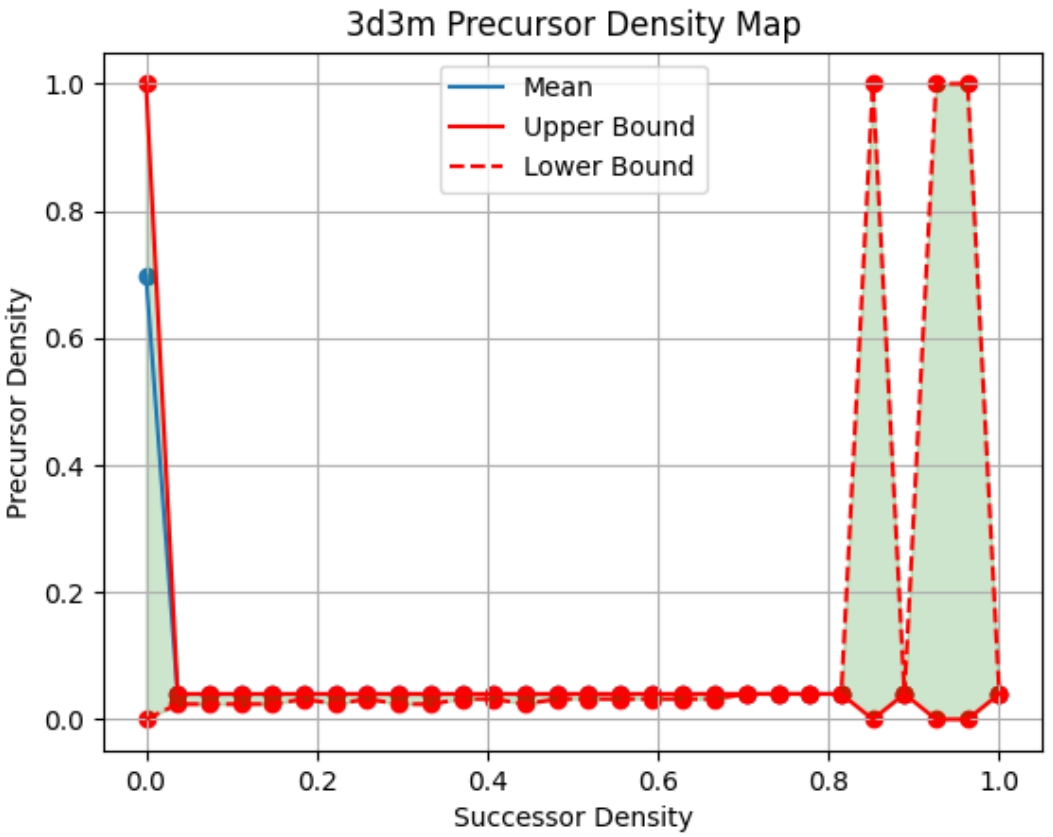

Figure 29: Precursor density mapped against successor density for a 3 dimensional grid with a bounding area of 3.

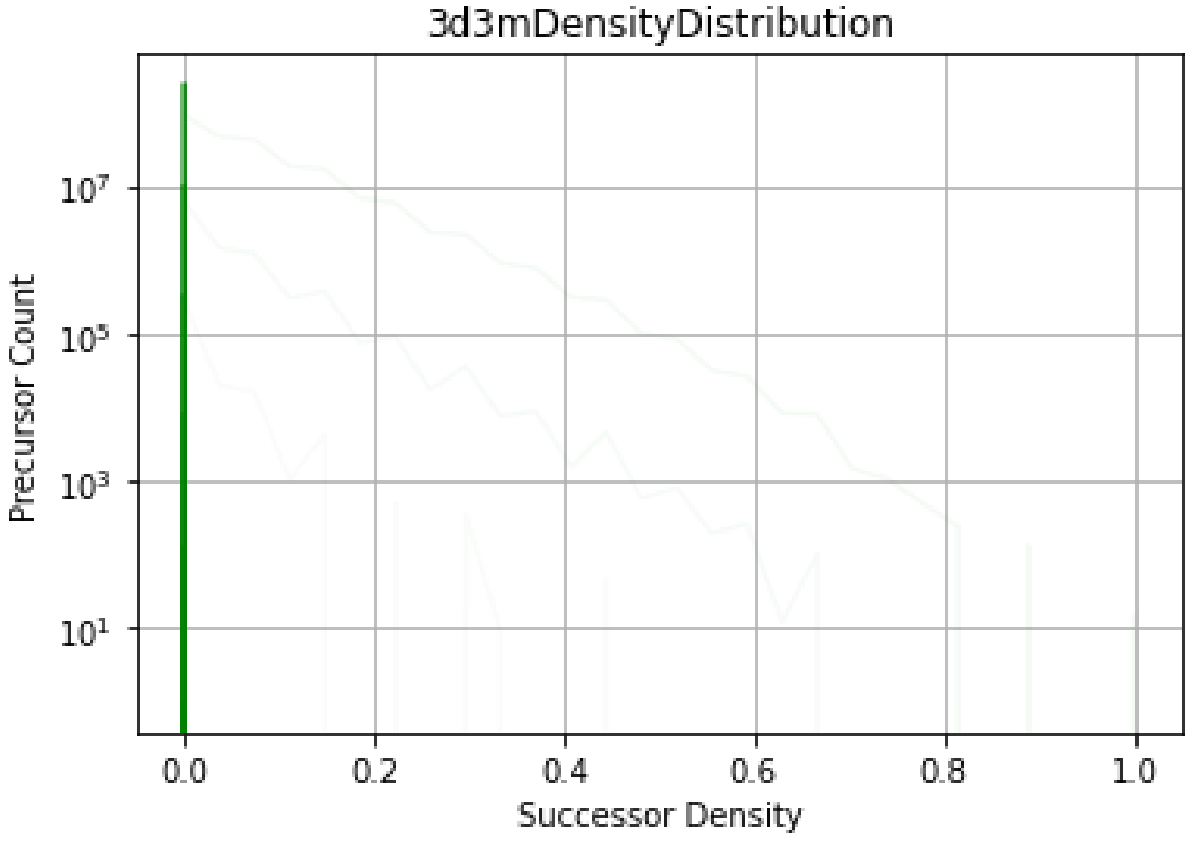

Figure 30: Precursor count mapped against successor density for a 3 dimensional grid with a bounding area of 3. Each line is representative of a density of precursor, and the opacity of the line is reflective of that density.

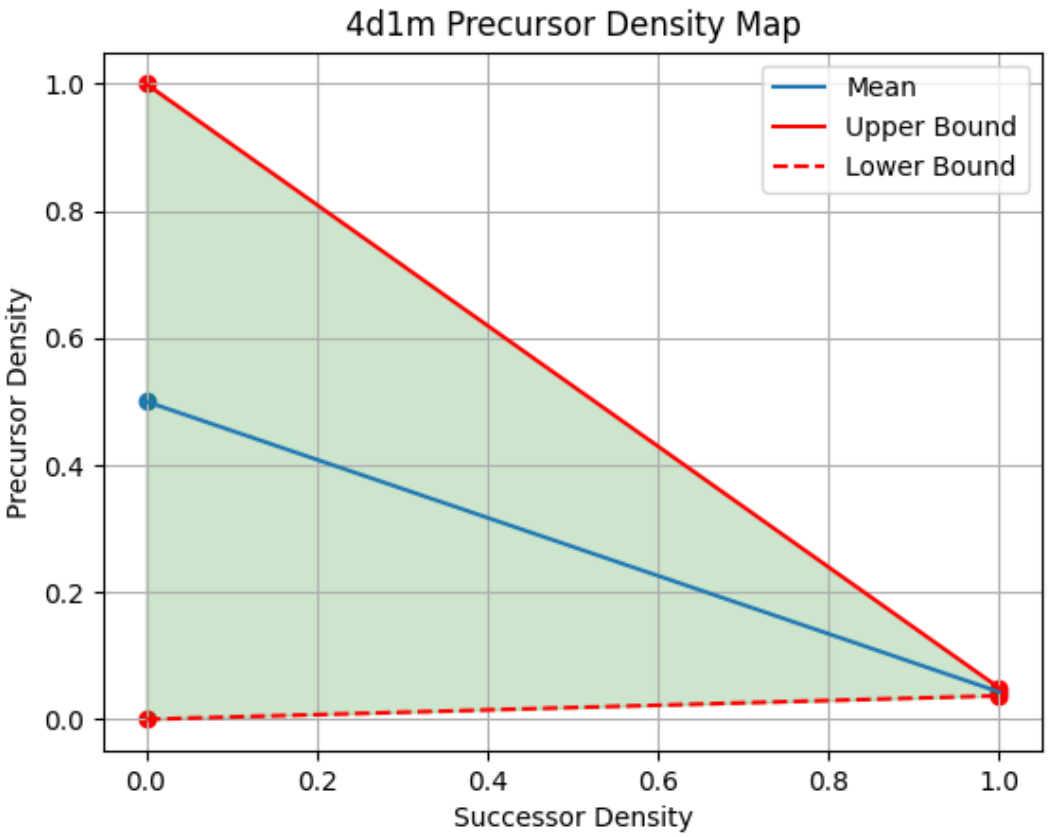

Figure 31: Precursor density mapped against successor density for a 4 dimensional grid with a bounding area of 1.

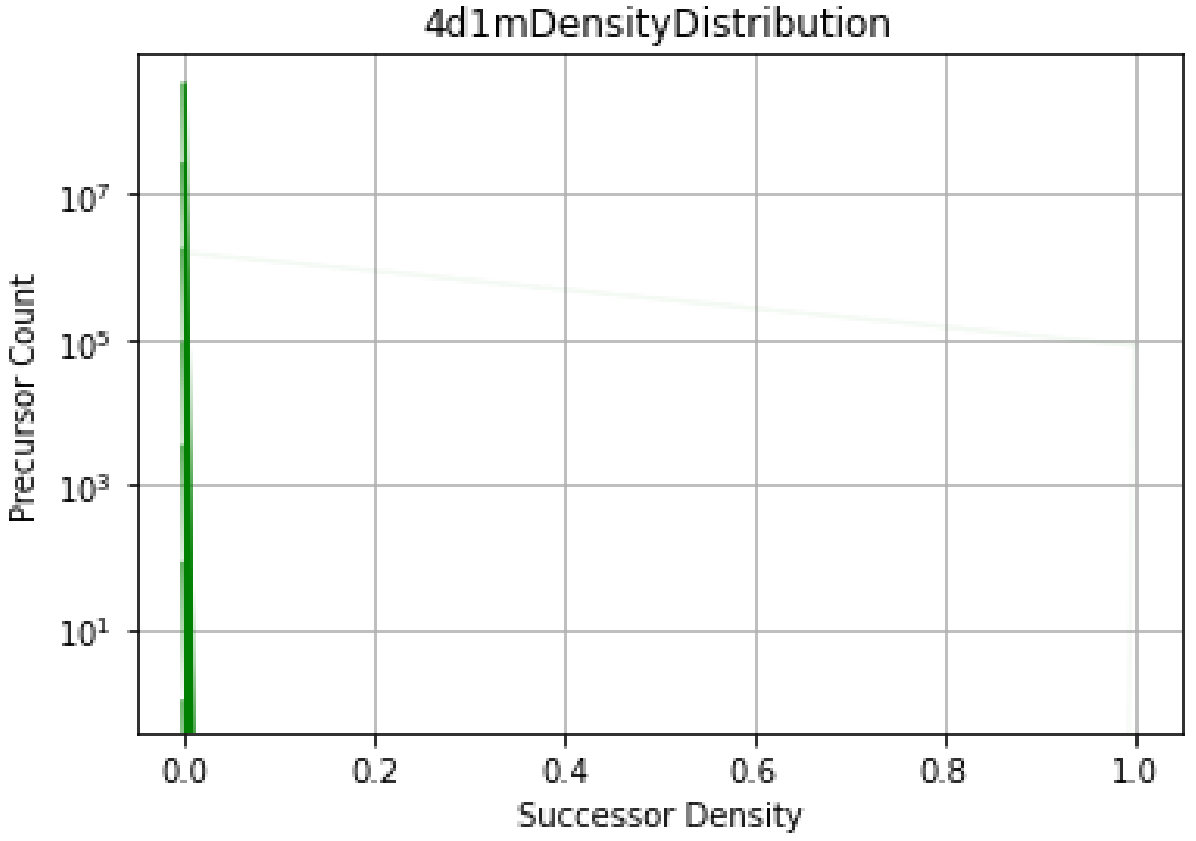

Figure 32: Precursor count mapped against successor density for a 4 dimensional grid with a bounding area of 1. Each line is representative of a density of precursor, and the opacity of the line is reflective of that density.

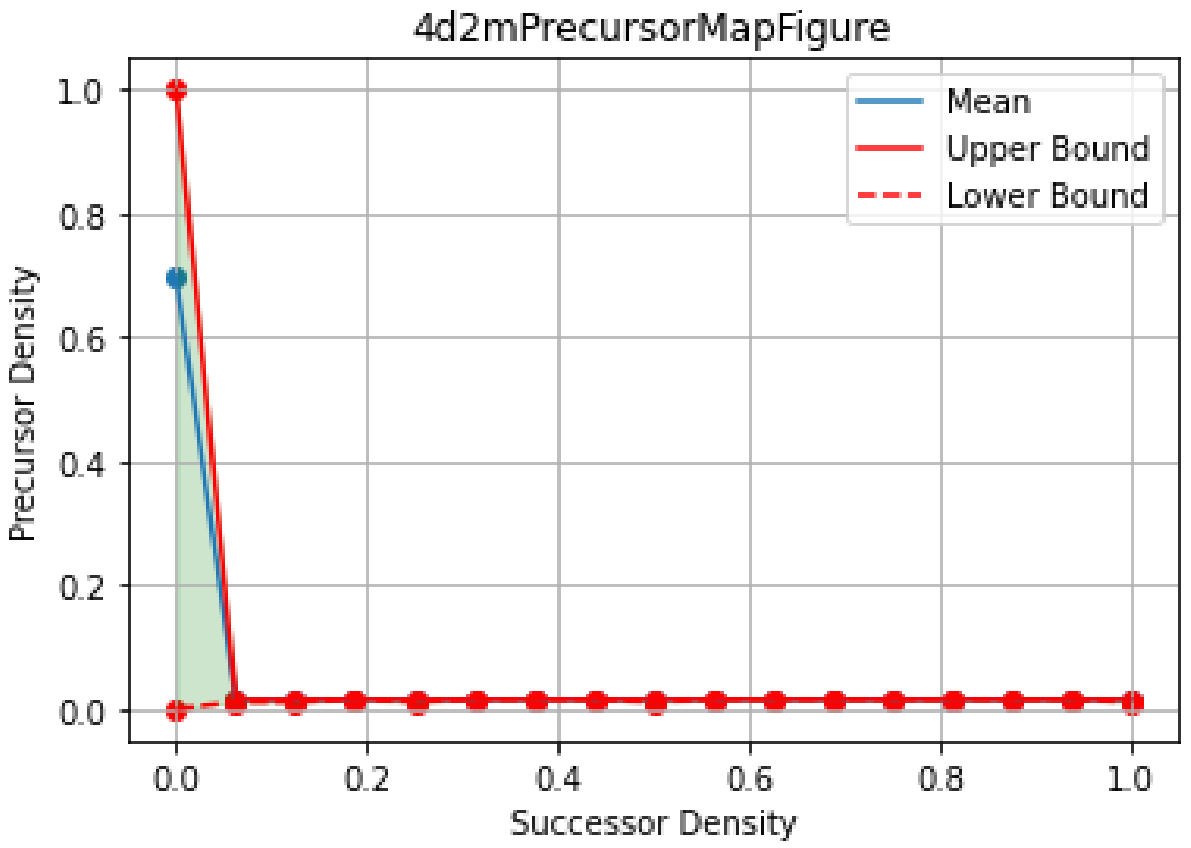

Figure 33: Precursor density mapped against successor density for a 4 dimensional grid with a bounding area of 2.

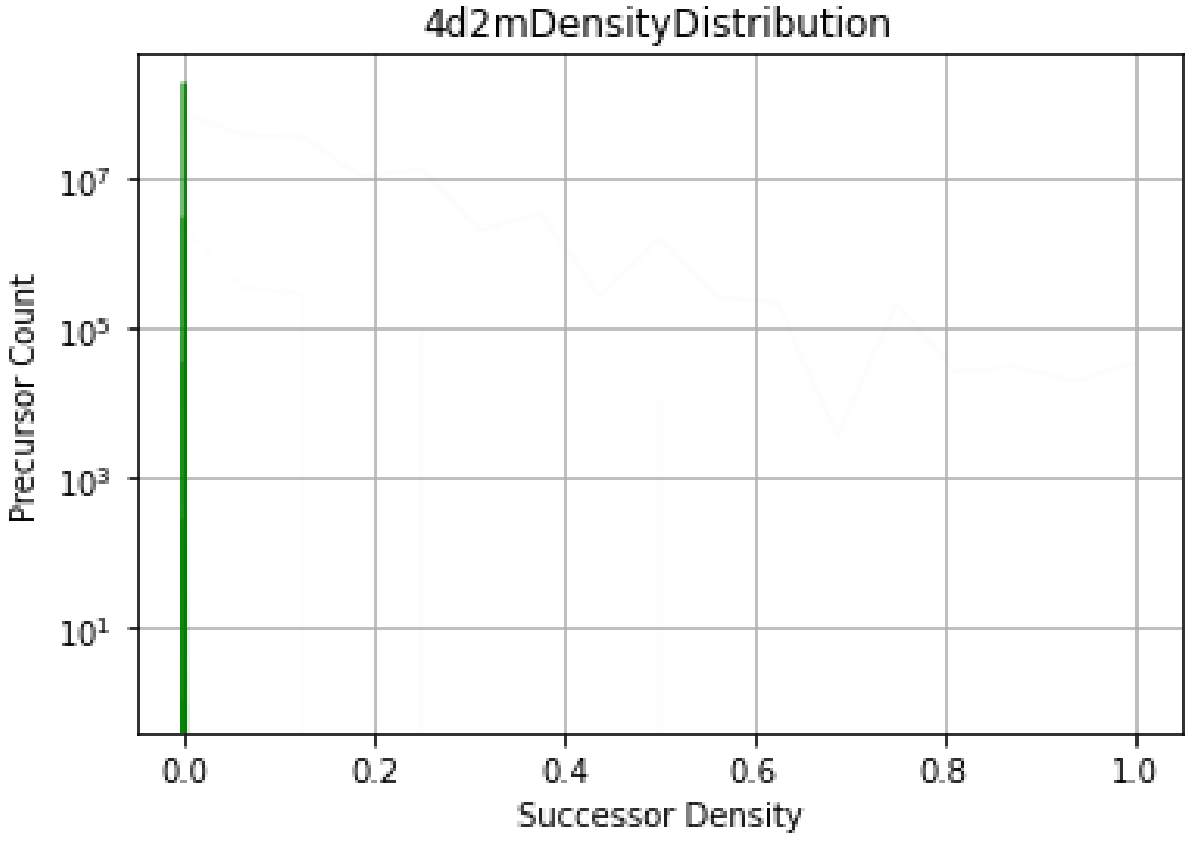

Figure 34: Precursor count mapped against successor density for a 4 dimensional grid with a bounding area of 2. Each line is representative of a density of precursor, and the opacity of the line is reflective of that density.

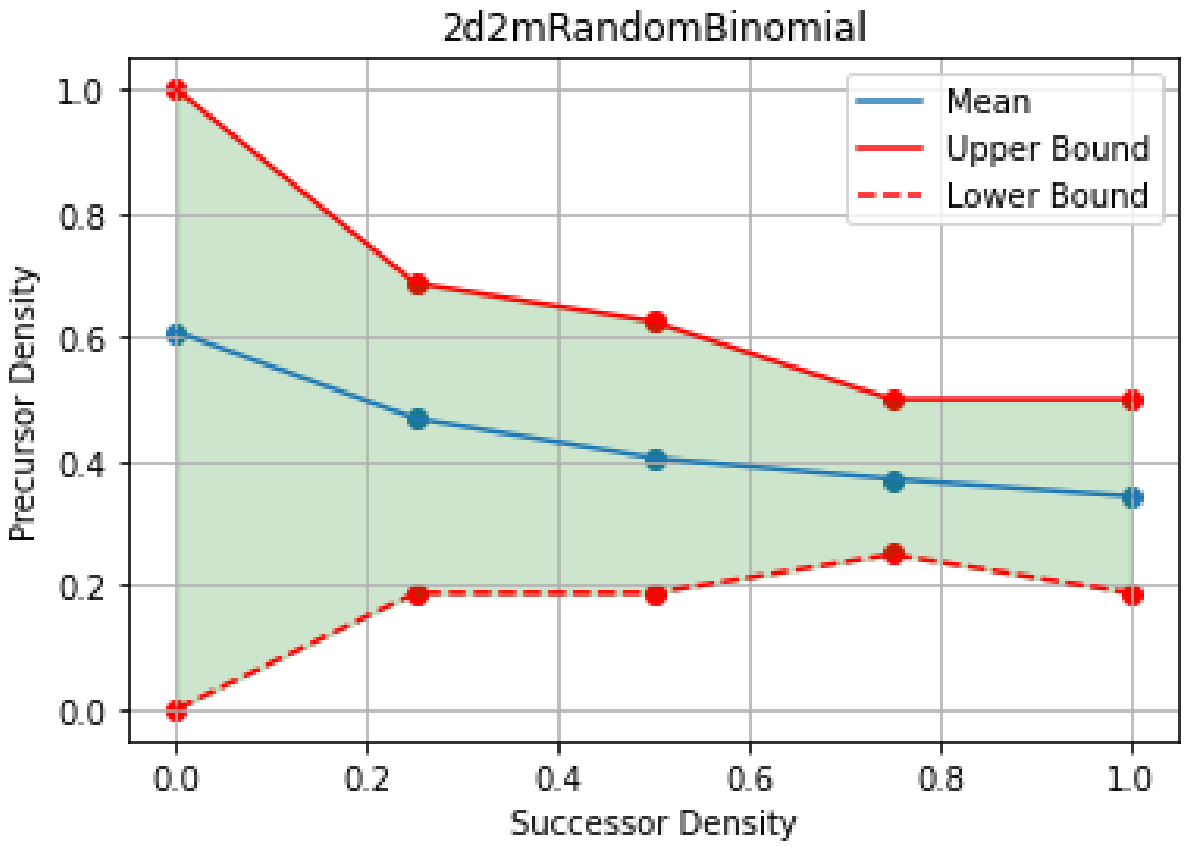

Figure 35: Precursor density mapped against successor density for a 2 dimensional grid with a bounding area of 2, utilizing a method of random sampling.

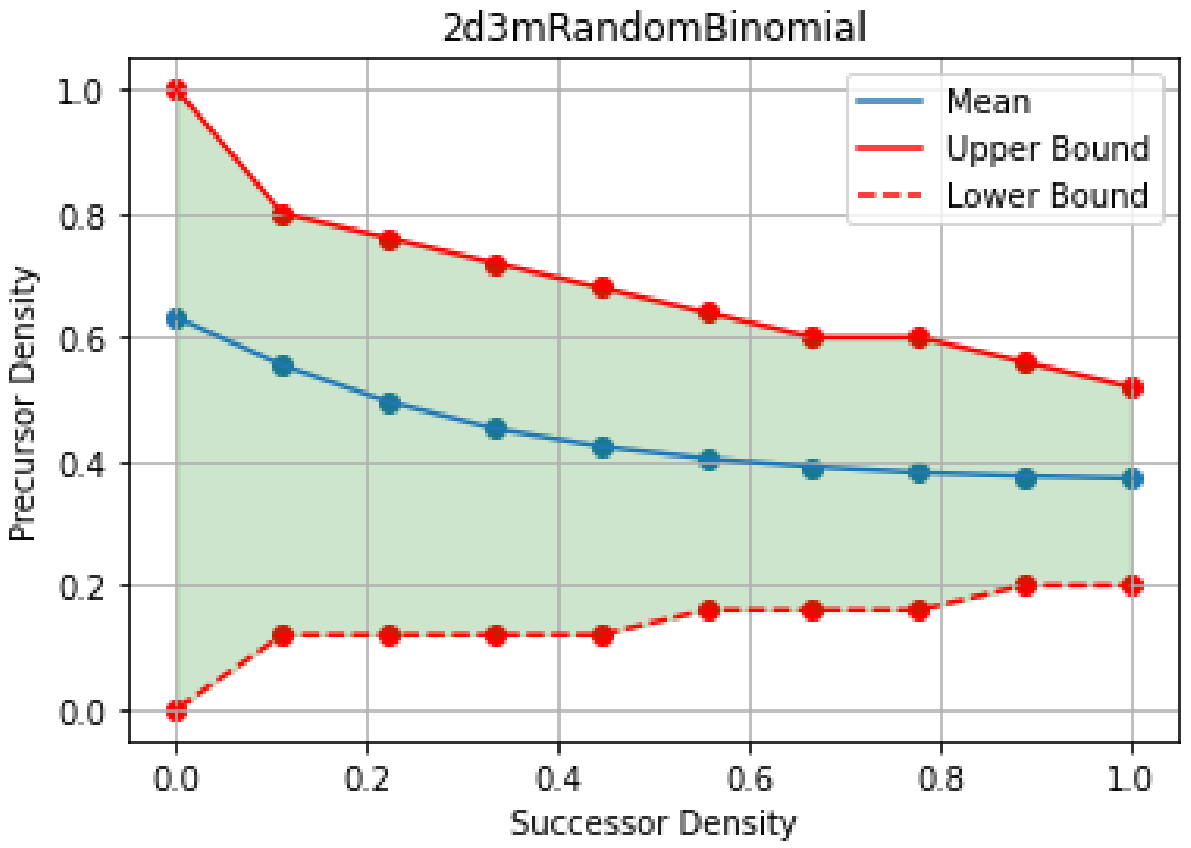

Figure 36: Precursor density mapped against successor density for a 2 dimensional grid with a bounding area of 3, utilizing a method of random sampling.

6 Conclusion

In conclusion, the results I have gathered are largely inconclusive due to larger dimensions and bounding areas having an amount of configurations large enough that computational restraints became an issue. However, I believe there is merit for further study into this subject area. If a more conclusive set of results could determine a pattern in the range of possible precursor densities and successor density it would allow for the greater ease of finding Gardens of Eden. Whilst the eventual goal would be to create complete maps of increasing bounding areas to find all Gardens of Eden, the ability to check for a specific Garden of Eden in a reduced time could be a boon in some circumstances. For example, a more localized, or smaller scale study that does not have access to a large amount of computational power. Alternatively, for finding Gardens of Eden that exist beyond currently exhausted bounding areas, potentially due to current hardware being unable to compute all possible precursor configurations within a reasonable time frame.

7 Conclusion

In conclusion, the results I have gathered are largely inconclusive due to larger dimensions and bounding areas having an amount of configurations large enough that computational restraints became an issue. However, I believe there is merit for further study into this subject area. If a more conclusive set of results could determine a pattern in the range of possible precursor densities and successor density it would allow for the greater ease of finding Gardens of Eden. Whilst the eventual goal would be to create complete maps of increasing bounding areas to find all Gardens of Eden, the ability to check for a specific Garden of Eden in a reduced time could be a boon in some circumstances. For example, a more localized, or smaller scale study that does not have access to a large amount of computational power. Alternatively, for finding Gardens of Eden that exist beyond currently exhausted bounding areas, potentially due to current hardware being unable to compute all possible precursor configurations within a reasonable time frame.

8 References

[24] *Print all combinations | Set-2*. https://www.geeksforgeeks.org/print-all-possible-combinations-of-r-elements-in-a-given-array-of-size-n/. Accessed: 10-02-2024. 2024.

[Bee22] Randall D. Beer. *Cultivating the Garden of Eden*. 2022. arXiv: 2210.07837 [nlin.CG].

[CC18] Tullio Ceccherini-Silberstein and Michel Coornaert. *The Garden of Eden theorem: old and new*. 2018. arXiv: 1707.08898 [math.DS].

[Gar70] Martin Gardner. *Mathematical Games: The fantastic combinations of John Conway's new solitaire game "life"*. 1970.

[Har+13] Christiaan Hartman et al. "Symmetry in gardens of eden". In: *the electronic journal of combinatorics* 20.3 (2013), P16.

[Hol22] David Wason Hollar Jr. "Cellular automata." In: *Salem Press Encyclopedia of Science* (2022). URL: https://proxy.library.lincoln.ac.uk/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=ers&AN=89316912&site=eds-live&scope=site.

[Moo62] Edward F Moore. *Machine models of self-reproduction*. American Mathematical Society New York, 1962.

[Myh63] John Myhill. "The converse of Moore's Garden-of-Eden theorem". In: *Proceedings of the american mathematical society* 14.4 (1963), pp. 685–686.

[Von49] John Von Neumann. "Theory and organization of complicated automata". In: *Burks (1966)* (1949), pp. 29–87.

A Project Research Plan

By Robin Critten and Supervised by Dr Ged Corob Cook

A.1 Introduction

In this project I will be studying Conway's cellular automata, the game of life, and the Moore-Myhill theorem. That Gardens of Eden can only exist within the game of life if and only if a state can be reached by two different preceding finite patterns. I will use various methods in order to prove Moore-Myhill theorem, including but not limited to, creating a simulation of the game of life, mathematical expressions as well as sketches. Further testing of the theorem can incorporate iterations on Conway's rule-set, as well as changing the dimensions of the cellular automata.

A.2 Motivation

When choosing this project, I was first drawn to it due to the fact I already had knowledge of the game of life. Whilst I was not over familiar with the subject I did have an interest in it. Upon further research that interest was only increased. After learning about the various applications of cellular automata, and how they can be applied to real world scenarios. An example of this would be how cellular automata can be used is to create computer systems of animal nervous systems, the first attempts of which were undertaken by Von Neumann [Hol22].

A.3 Literature Survey

Moore's Machine Models of Self Reproduction [Moo62] proposes several definitions to formalise the understanding of Von Neumann's 2 dimensional euclidean spaces, subdivided into squares of equal size. For example calling them tessellations. However, the main interest of Moore's paper for this project is his theorem on finding Gardens of Eden. Within a tessellation, the presence of Gardens of Eden can be proven by finding a pair of differing configurations such that their sequents cannot be distinguished. In Myhill's The Converse of Moore's Garden-of-Eden Theorem [Myh63]. He implies that if a Garden of Eden exists, there must therefore be the existence of twins. Two distinguishable twins that have the same sequent. Therefore forming the Moore-Myhill theorem by proving the converse of Moore's original theorem.

Cellular automata [Hol22] is an article that conveys a lot if information surrounding cellular automata. This includes a brief history of their beginnings in Von Newman's work, in reproducing biological processes on a computer. Other key notes in cellular automata's history are mentioned, such as the Creation and popularisation of Conway's game of Life. Furthermore, the article also gives examples of some of the key uses for cellular automata's. Growth of fauna population's and thermodynamics are both given as examples.

Tullio Ceccherini-Silberstein and Michel Coornaert's The Garden of Eden Theorem: Old and New [CC18], and Randall D. Beer's Cultivating the Garden of Eden [Bee22] both present an overview on topics surrounding the Moore-Myhill theorem and previous works discussing it. Silberstein takes a broader approach, reviewing topics that have since been researched within the field of Gardens of Eden. These include Garden of Eden theorems for both amenable and non-amenable groups. He does not limit the scope of his review to just Conway's game of life, but discusses cellular automata in more general terms. Beer's approach was to write in more detail surrounding a specific aspect of research into Gardens of Eden. He presents a series of results into how computers have been used in order to try and find Gardens of Eden within the Game of Life, the implications and limitations of their use, as well as how different conditions have effected have an effect on the results.

A.4 Equipment, Facilities ans Supervision

I will require no access to any labs or equipment for the duration of my project, from my initial assessment. The coding part of the project I will be able to do from my laptop using NetLogo as my planned coding language. Current plans with my supervisor, Dr Ged Corob Cook, is to meet every Tuesday at 2pm for 15 minuets This will be extended to a bi-weekly meeting for 30 minuets after the first 6 weeks.

A.5 Action Plan (Gantt Chart)

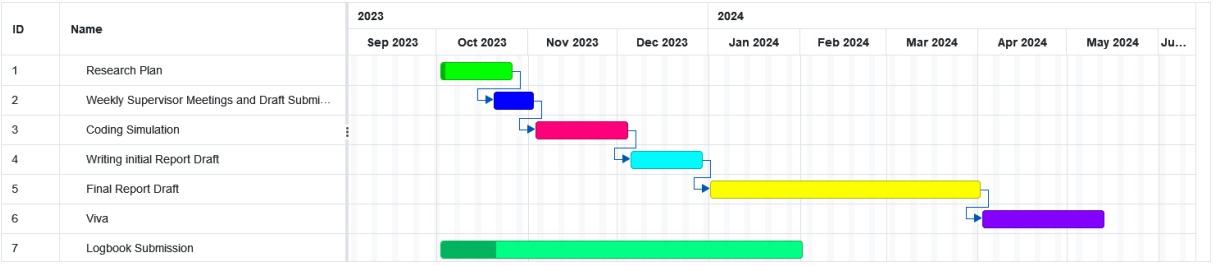

Figure 37: Gantt Chart for Project

B Ethics Documentation

Application Details

Ethics Reference: $UoL2023_16497$

Title of Project: Gardens of Eden in the Game of Life

Lead Researcher: Robin Critten

Academic Supervisor (if applicable): Ged Cook

Date of Letter: 31 October 2023

FAVOURABLE OPINION

Thank you for the submission of your Project Registration Form (PRF), on behalf of the committee and I am pleased to confirm a favourable ethical opinion for the above research on the basis described in the application form and supporting documentation.

The favourable ethical opinion provided is conditional to the following requirements:

1. Commencement of the research

1.1 Governance Audit: Your application may be subject to audit, should any issues your application be identified, your application may be returned to you for modification and a full ethics application may be required.

1.2 Risk Assessment: In accordance with H&S policy and guidance, a risk assessment must be completed or existing risk assessment reviewed/updated before any data collection commences. A copy of the risk assessment should be retained with your research data.

1.3 It is assumed that the research will commence within 12 months of the date of the favourable ethical opinion.

1.4 If the research does not commence within 12 months of the favourable opinion being issued, the lead applicant (or academic supervisor for student research) should send a written explanation for the delay. A further written explanation should be sent after 24 months if the research has still not commenced.

1.5 If the research does not commence within 24 months, the REC may review its opinion.

2. Duration of favourable opinion

2.1 The favourable ethical opinion of the Research Ethics Committee (REC) for a specific research study applies for the duration of the study, as detailed in your application (or any subsequent amendments).

3. Amendments

3.1 If it is proposed to make an amendment to the research, the lead applicant (authorised by the academic supervisor for student research) should submit an amendment to the REC by accessing the original application form on LEAS and creating an amendment form.

4. Monitoring

4.1 A REC may review a favourable opinion in the light of progress reports and any developments relevant to the study. The lead applicant and academic supervisor (for student research), is responsible for ensuring the research remains scientifically sound, safe, ethical, legal and feasible throughout its duration. The lead applicant and academic supervisor (for student research) should submit a progress report to the REC 13 months after the date on which the favourable opinion was given. Annual progress reports should be submitted thereafter.

4.2 Progress reports should be completed and submitted using the forms in LEAS.

5. Conclusion or early termination of the research

5.1 The Lead Applicant should complete the End of Study Form in LEAS once the study has completed. It is also their responsibility to inform the REC of early termination of the project or if the work is not completed.

6. Long Term Studies

6.1 The lead applicant and academic supervisor (for student research) is responsible for ensuring that the study procedures and documentation are updated in light of legislative or policy changes and also for reasons of good practice (e.g. standards for supporting documentation). This should be documented in the progress report to the REC (see above) and, where necessary, an amendment (see above) should be submitted to the REC. The REC may review its opinion in light of legislative changes or other relevant developments.

Additional guidance may be found at here

Statement of Compliance

The Committee is constituted in accordance with the University of Lincoln Research Ethics policy and E-QMS SOP E-01 Ethics Committee Operations.

Yours Sincerely

Sam Lewis | Research Governance Manager, on behalf of University Research Ethics Committee