Fractal Structure of Rainforests

Lucas Trigal* and Luis Irisarri†

Institute for Cross-Disciplinary Physics and Complex Systems, IFISC (CSIC-UIB)
Campus Universitat de les Illes Balears, E-07122 Palma de Mallorca, Spain

1 INTRODUCTION

Recent research in forest ecology has increasingly focused on the concept of self-organization and criticality in forest ecosystems, particularly in the context of gap dynamics in rainforests [1–4]. This research has revealed that non-linear systems, operating far from equilibrium and possessing extended spatial degrees of freedom, frequently transition spontaneously into a state characterized as "self-organized critical". The pioneering work by Solé and Manrubia has been instrumental in this area. In this work, a paradigm to study fractal properties of low-canopy gaps in rainforests is established. The study focuses on two parts. Firstly, the examination of the Barro Colorado Island (BCI) rainforest in Panama. Then they formulate a *cellular automaton* that allows for a more general study of a rainforest model. They propose that the appearing patterns are indicative of a self-organized critical state. Moreover, they argue that this self-organized critical state should exhibit fractal behaviour because, otherwise, if it were a non-fractal distribution, we would be referring to a state that follows a non-arbitrary order. This distinction arises because they posit that the non-linear dynamical process of gap formation in forest ecosystems can give rise to the formation of fractal structures, treefall disturbances are important for the persistence of the ecosystem. Therefore, fractality would result from the fact that the distribution of gaps and rainforest does not adhere to a traditional geometry. Alternatively, it could stem from the jungle exhibiting self-similarity across different areas at a fixed scale.

In order to characterize this patterns, one can measure the fractal dimension. For a *self-similar* fractal composed of m copies of itself scaled by a factor s, the fractal dimension D is defined as [5, p. 413]:

$$D = \frac{\log m}{\log s}. \tag{1}$$

However, most fractals are not self-similar and generalizations of this definition are needed. In fact, many definitions have been proposed (see [6] for a further details). In this work, we will consider the *Box Dimension* definition [5, p. 416]:

$$D = \lim_{\epsilon \to 0} \frac{\log N(\epsilon)}{\log(1/\epsilon)}, \tag{2}$$

where $N(\epsilon)$ is the number of boxes of size ϵ needed to cover the fractal, and if the value of D were a non-integer number, we would be dealing with a geometry that cannot be analyzed using "traditional" tools. More generally, we consider the **Correlation Dimension of Order** q defined by[1] [1, p. 32]:

$$D_q := \lim_{\epsilon \to 0} \frac{1}{(q-1)} \frac{\log[X(q)]}{\log(\epsilon)}, \qquad q \in \mathbb{R}, \tag{3}$$

where:

$$X(q) := \sum_{i=1}^{N(\epsilon)} p_i^q. \tag{4}$$

Here, p_i is the probability that a box of size ϵ is populated. This dimension provides us with a wealth of information about the geometric structure of the fractal set.

On the other hand, since real fractals are in fact multifractals [1, p. 34], we further consider the **Fractal Spectrum Dimension** $f(\alpha)$ [7]:

$$f(\alpha(q)) = q\alpha(q) + D_q(1-q), \tag{5}$$

where $\alpha(q)$ is the diverging exponent defined by $p_k \approx \epsilon_k^{\alpha}$, which can be determined through [7]:

$$\alpha(q) = \frac{\mathrm{d}}{\mathrm{d}q}\left[(q-1)D_q\right]. \tag{6}$$

*mail: ltg072@id.estudiant.uib.es
†mail: luisirisarri1@id.estudiant.uib.es

[1]Notice how for $q = 0$ we re-obtain eq. (2).

This report will delve into their findings, specifically analysing the forest game model they introduced and its implications for understanding rainforest ecology.

2 FOREST GAME

The *Forest Game*, a cellular automaton model developed by Solé and Manrubia, serves as a crucial tool for simulating the dynamics of forest gaps and their macroscopic spatial regularities. This model, based on simple rules of tree growth, competition, and death, illustrates how complex patterns emerge from straightforward processes.

The Forest Game is defined by a two-dimensional square lattice of $L \times L$ points together with periodic boundary conditions [3, p. 533]. Each cell (i, j) represents a tree of height $S_n(i, j)$ at the time step n. The tree height is bounded by $S_0 \leq S_n(i, j) \leq S_c$, where S_0 is the minimum and S_c is the maximum tree size. The game's rules are as follows:

1. **Birth rule**: A tree of size S_0 is born with probability p_b in each empty cell.
2. **Growth rule**: A tree of size $S_n(i, j)$ will grow to:
$$S_{n+1}(i, j) = S_n(i, j) + \Delta n(i, j), \tag{7}$$
where $\Delta n(i, j)$ is the Heaviside function defined as:
$$\Delta_n(i, j) := \Theta\left[\mu - \frac{\gamma}{8} \sum_{\langle r,s \rangle} S_n(r, s)\right]. \tag{8}$$
Here, μ is a parameter and γ is the interaction strength. The sum is over the 8 nearest neighbours of the cell (i, j), that is, we consider the Moore neighbourhood.
3. **Death rule**: A tree of size $S(i, j)$ dies with probability p_d or if $S(i, j) \geq S_c$.
4. **Gap formation rule**: A dying tree will form a gap of radius R if the size of neighbours (not necessarily the nearest ones) is less than or equal to the size of the tree. That is [2, p. 51]:
$$\sum_{B(R)} S_n(r, s) \leqslant S_n(i, j) \tag{9}$$
where $B(R)$ is the neighbourhood of radius R around the cell (i, j).

The significance of the Forest Game lies in its ability to replicate observed natural phenomena using a minimal set of rules. By adjusting parameters such as tree growth rate, death probability, and competition, the model can exhibit a variety of patterns that resemble real-world forest dynamics. The game's results support the idea that rainforest ecosystems might be operating near a critical point, where a small change could lead to significant transformations in the forest structure. This insight has profound implications for understanding the stability and biodiversity of rainforests, providing a theoretical framework for future ecological studies and conservation strategies.

2.1 Numerical Implementation

Regarding the computational implementation, we consider a random forest as the initial state of the system. We have taken the following parameters by default $S_0 = 0.1$, $S_c = 30$, $\mu = \gamma = 1$ and $p_b = 0.3$.

In order to compute the fractal measures, we have followed the paper's methodology calculating $X(q)$ by [2, p. 51]:
$$X_q(\epsilon) = \sum_{j=1}^{j_{\max}} N(j) \left(\frac{j}{N_P}\right)^q. \tag{10}$$
where $N(j)$ is the number of boxes containing j gap points, N_P is the total number of gap points and $j_{\max}$ is the maximum number of gap points in a box. In this manner, we have taken $j_{\max} = 16$ and $\epsilon = 1/20$. The code is available in appendix A.

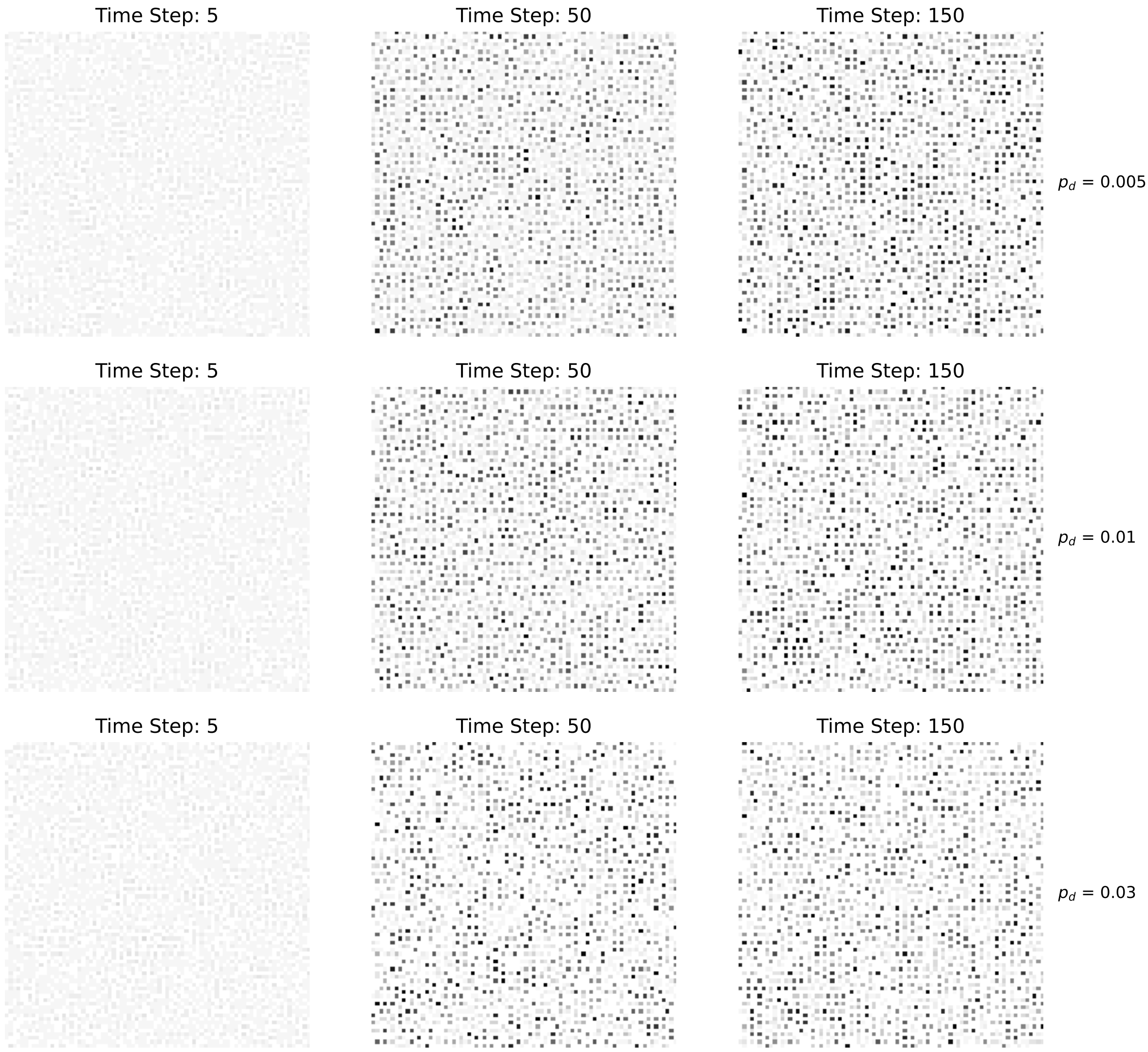

Figure 1: Images of the forest simulation according to the automaton presented earlier for different times and different death probabilities ($p_d = 0.005, 0.03, 0.5$) are provided. As can be observed, they all start from a more or less empty forest with low density, and over time, they form the distribution of trees and gaps that we can see. It can be observed that with the increase in the death probability, for the last time, there are fewer darker black points, indicating lower density, and more white gaps. This is directly related to the fractal behavior of this system and will be discussed later. However, we can see how this is a easily recognizable phenomenon in the images of the forest. A lattice of 80x80 was used, and the birth probability was $p_b = 0.3$

3 RESULTS

In this section, we will present the results obtained from the Forest Game. We will start by measuring the *biomass* evolution in time[2]. Then, we will compute the fractal dimension and spectrum dimension for an equilibrium state.

[2]Time here is discrete, representing either years or seasons, depending on study of interest.

3.1 Biomass Evolution

We define the biomass at time step n by:

$$B_n = \sum_{i,j}^{L} S_n(i, j). \tag{11}$$

That is, we sum the size of all the trees in the lattice at time step n. In fig. 2 we show the evolution of the biomass for $L = 40$ in $t \in \{1, \dots, 700\}$. We can see that initially, the biomass increases significantly, but after $n \approx 50$ time steps, it reaches an equilibrium state. At this point, the rainforest attains an attractor of the self-organized critical state, that is, it fluctuates around a constant value. These fluctuations arise from the nonlinearity of the gap formation mechanism.

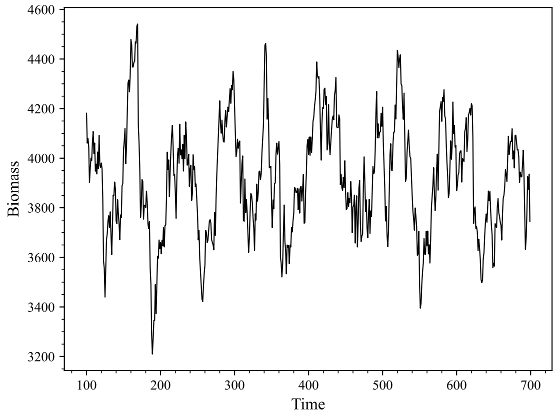

Figure 2: Biomass Dynamics. We compute the biomass for $L = 40$ with $p_d = 0.01$ in $t \in \{1, \ldots, 700\}$ via eq. (11).

Furthermore, we have analysed the fourier spectrum from $n = 100$ to ensure the equilibrium state. We observe in fig. 3 that the behaviour of the spectrum scales like $f^{-\beta}$, with a β value falling within the range [missing value for us].

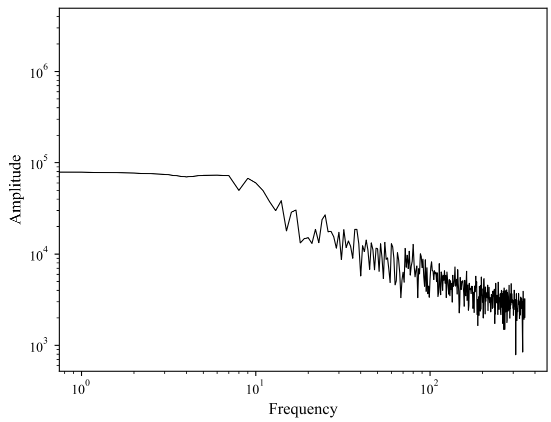

Figure 3: Fourier Transform of the Biomass. For the same values as in fig. 2. The obtained β value is 0.833. Near to the value of the reference [3]

3.2 Fractal Structure

We now study the fractal structure of the rainforest. We have computed the fractal correlation dimension and the fractal spectrum dimension for[3] $p_d \in \{0.005, 0.01, 0.03\}$ and $L = 80$. In fig. 4 we show D_q for $q \in [-8, 8]$ (RHS plot). We observe that as the mortality rate increases, the fractal dimension tends towards 2. This implies that as the likelihood of mortality and gap formation rises, the critical state of the system becomes more random. All these values are close to those obtained from a similar study on a real rainforest. In the case of BCI [1], the results indicate a fractal dimension of around 1.86. In our case, we observe a certain dependence of the fractal dimension on the death probability, which increases as the probability of death increases. The average results obtained for the fractal dimension of the cases simulated by the automaton are as follows: for $p_d = 0.005$, la $D_0 = 1.81 \pm 0.02$, for $p_d = 0.01$, $D_0 = 1.90 \pm 0.01$, and, for $p_d = 0.03$,$D_0 = 1.96 \pm 0.02$. In all cases, we obtain values less than 2, indicating a fractal structure. The dependence on death probability suggests that as the probability of a tree dying and forming a gap increases, the distribution is affected and approaches an integer value of 2. We could hypothesize that this is the limit when the death probability is maximum, effectively killing the forest and completely losing any fractality. Furthermore, with the figure 4, we can obtain values for other important magnitudes in the study of fractals, such as the information dimension (D_1) or correlation dimension (D_2).Therefore, once this relationship is detected, it is interesting to view this model as a tool for the calculation and control of forest health.

Therefore, we observe that our automaton can predict the distribution of a real rainforest and its fractal properties.

[3] These are typical rates for BCI rainforest [8]

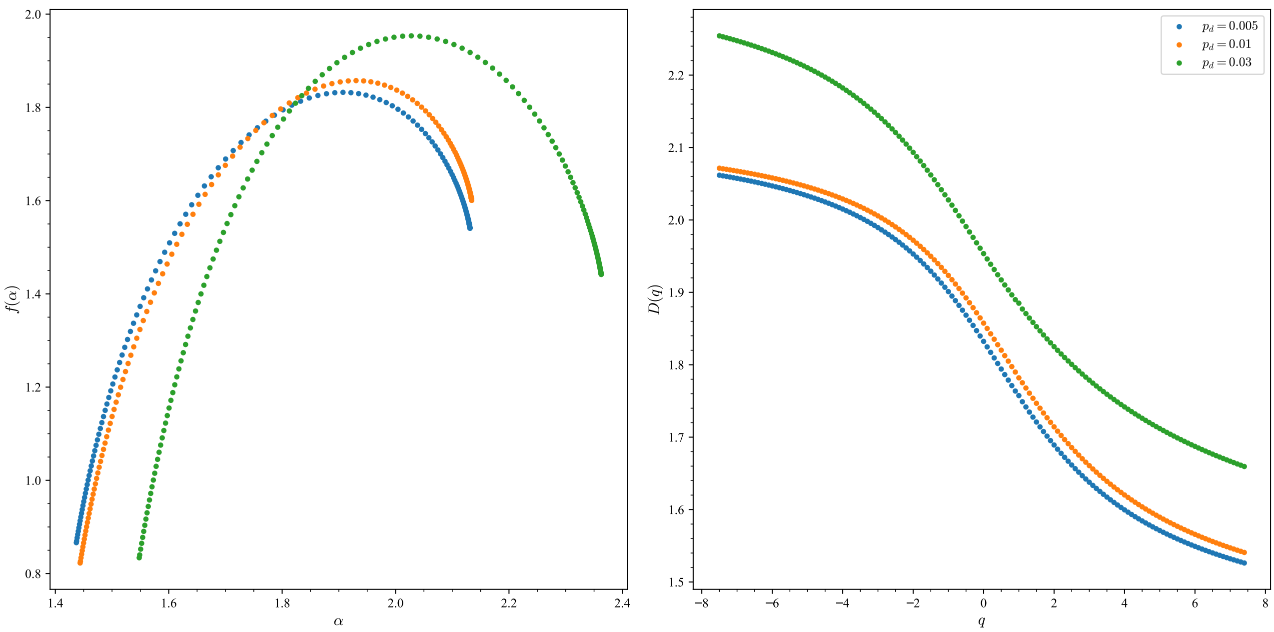

Figure 4: a) Multifractal spectrum and b) Fractal dimension of the forest for different values of the death probability. A lattice of 80x80 was used, and the birth probability was $p_b = 0.3$

4 CONCLUSIONS

The present report has served to rework the results obtained in previous works with the intention of highlighting the relevance of such approaches, ranging from the study of fractals to applications in ecological scenarios. Through the formulation of an automaton with non-linear rules, a critical state has been achieved, manifesting behaviors such as self-organization or multifractality in the distribution of gaps. However, not only does the gap distribution exhibit distinct behavior, but also the biomass, showcasing temporal self-similarity. Nevertheless, this doesn't necessarily represent the limit of the automaton, as many of the conditions that constitute it are limited or simplistic.

In the current context, concerns about climate change and its impact on the biosphere could prompt a revival of such approaches. Exploring the relationship between an increase in mortality and its potential effects on the fractal geometry of clearings in a forest becomes pertinent. However, with the rules established so far, we have achieved a system that self-organizes to reach a critical state, displaying spatial patterns that are self-similar.

References

[1] Ricard V. Solé and Susanna C. Manrubia. "Are Rainforests Self-Organized in a Critical State?" In: *Journal of Theoretical Biology* 173.1 (Mar. 7, 1995), pp. 31–40. ISSN: 0022-5193. DOI: 10.1006/jtbi.1995.0040. URL: https://www.sciencedirect.com/science/article/pii/S0022519385700409 (visited on 11/22/2023) (cit. on pp. 1, 2, 4).

[2] Ricard V. Solé and Susanna C. Manrubia. "Self-Similarity in Rain Forests: Evidence for a Critical State". In: *Physical Review E* 51.6 (June 1, 1995), pp. 6250–6253. ISSN: 1063-651X, 1095-3787. DOI: 10.1103/PhysRevE.51.6250. URL: https://link.aps.org/doi/10.1103/PhysRevE.51.6250 (visited on 12/25/2023) (cit. on pp. 1, 2).

[3] Susanna C. Manrubia and Ricard V. Solé. "Self-Organized Criticality in Rainforest Dynamics". In: *Chaos, Solitons & Fractals* 7.4 (Apr. 1996), pp. 523–541. ISSN: 09600779. DOI: 10.1016/0960-0779(95)00091-7. URL: https://linkinghub.elsevier.com/retrieve/pii/0960077995000917 (visited on 12/25/2023) (cit. on pp. 1, 2, 4).

[4] Susanna C. Manrubia and Ricard V. Solé. "On Forest Spatial Dynamics with Gap Formation". In: *Journal of Theoretical Biology* 187.2 (July 1997), pp. 159–164. ISSN: 00225193. DOI: 10.1006/jtbi.1997.0409. URL: https://linkinghub.elsevier.com/retrieve/pii/S0022519397904094 (visited on 12/25/2023) (cit. on p. 1).

[5] Steven H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Second edition. Boulder, CO: Westview Press, a member of the Perseus Books Group, 2015. 513 pp. ISBN: 978-0-8133-4910-7 (cit. on p. 1).

[6] K. J. Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. 2nd ed. Chichester, England: Wiley, 2003. 337 pp. ISBN: 978-0-470-84861-6 978-0-470-84862-3 (cit. on p. 1).

[7] Thomas C. Halsey et al. "Fractal Measures and Their Singularities: The Characterization of Strange Sets". In: *Physical Review A* 33.2 (Feb. 1, 1986), pp. 1141–1151. ISSN: 0556-2791. DOI: 10.1103/PhysRevA.33.1141. URL: https://link.aps.org/doi/10.1103/PhysRevA.33.1141 (visited on 12/26/2023) (cit. on p. 1).

[8] Diana Lieberman et al. "Mortality Patterns and Stand Turnover Rates in a Wet Tropical Forest in Costa Rica". In: *Journal of Ecology* 73.3 (1985), pp. 915–924. ISSN: 0022-0477. DOI: 10.2307/2260157. JSTOR: 2260157. URL: https://www.jstor.org/stable/2260157 (visited on 12/27/2023) (cit. on p. 4).

A CODE

```
    def rainforest(L, S0, p):
    """Defines a random initial state of the forest. That is a square lattice of size LxL with a probability p of having a tree of size S0 in each cell.

    Args:
        L (int): Lattice size
        S0 (float): Minimum tree size
        p (float): Probability of having a tree of size S0 in each cell

    Returns:
        np.array((L,L)): Lattice with the initial state of the forest
    """
    lattice = np.zeros((L, L))

    # Create a mask where each cell has a probability (p) of being True
    growth_mask = np.random.random((L, L)) <= p

    # Set cells in the lattice to S0 where the growth mask is True
    lattice[growth_mask] = S0

    return lattice

def birth(lattice, p_birth, S0):
    """Defines the birth rule for the forest. That is, a tree of size S0 is born with probability p_birth in each empty cell.

    Args:
        lattice (np.matrix): forest lattice
        p_birth (float): probability of birth
        S0 (float): minimum tree size

    Returns:
        np.matrix: forest lattice after applying the birth rule
    """
    # Create a mask for empty cells
    empty_mask = (lattice == 0)

    # Generate a random array of the same shape as the lattice
    random_values = np.random.random(lattice.shape)

    # Create a birth mask where cells are empty and the random value is below the birth threshold
    birth_mask = empty_mask & (random_values <= p_birth)

    # Set cells in the lattice to S0 where the birth mask is True
    lattice[birth_mask] = S0
```

```

    return lattice

def heaviside_vectorized(mu, gamma, lattice):
    """heaviside_vectorized calculates the Heaviside function values for
       ↪ each cell in the lattice. This accounts for \Detla n_{ij} in the
       ↪ growth rule.

    Args:
        mu (float): parameter
        gamma (float): interaction strength
        lattice (np.matrix): forest lattice

    Returns:
        np.matrix: Heaviside function values for each cell in the lattice
    """
    # Define the kernel for neighbor sum
    kernel = np.array([
        [1, 1, 1],
        [1, 0, 1],
        [1, 1, 1]
    ])
    neighbors_sum = convolve2d(lattice, kernel, mode='same', boundary='wrap'
       ↪ ) # https://docs.scipy.org/doc/scipy/reference/generated/scipy.
       ↪ signal.convolve2d.html

    # Calculate the Heaviside function values
    heaviside_values = mu - (gamma / 8) * neighbors_sum
    heaviside_values[heaviside_values < 0] = 0 # Set negative values to zero

    return heaviside_values

def growth(lattice, S0, Sc, mu, gamma):
    """Defines the growth rule for the forest.

    Args:
        lattice (np.matrix): forest lattice
        S0 (float): minimum tree size
        Sc (float): maximum tree size
        mu (float): parameter
        gamma (float): interaction strength

    Returns:
        np.matrix: forest lattice after applying the growth rule
    """

    # Create a mask for cells that are eligible for growth
    growth_mask = (lattice <= Sc) & (lattice >= S0)

```

```python
    # Calculate the Heaviside values for the entire lattice
    heaviside_values = heaviside_vectorized(mu, gamma, lattice)

    # Apply growth only to eligible cells
    lattice[growth_mask] += heaviside_values[growth_mask]

    return lattice

def find_nth_moore_layer(lattice, i, j, n):
    """Find the nth layer of Moore neighbors in a square lattice, i.e, a
        ↪ square (2n+1)x(2n+1) excluding the inner (2n-1)x(2n-1) square. (
        ↪ see https://mathworld.wolfram.com/MooreNeighborhood.html for more
        ↪ details).

    Args:
        i (int): x coordinate of the cell.
        j (int): y coordinate of the cell.
        n (int): n-th Moore neighborhood.
        grid_size (int): The size of the grid.

    Returns:
        list: list of tuples with the coordinates of the nth Moore neighbors
            ↪ .
    """
    L = lattice.shape[0]
    neighbors = []
    for dx in range(-n, n+1):
        for dy in range(-n, n+1):
            if max(abs(dx), abs(dy)) == n:  # This condition excludes inner
                ↪ squares
                # Wrap around the grid using modulo operator to maintain
                    ↪ periodic boundaries
                neighbors.append(((i + dx) % L, (j + dy) % L))

    return neighbors

def find_neighbors_at_R(lattice, i, j, R):
    """Finds the neighbors of a cell exactly at a radius R. Here we consider
        ↪  euclidian distance.

    Args:
        lattice (np.matrix): forest lattice
        i (int): x coordinate of the cell
        j (int): y coordinate of the cell
        R (float): radius

    Returns:
        list: list of tuples with the coordinates of the neighbors
    """
```

```python
    n_rows, n_cols = lattice.shape
    max_offset = int(np.ceil(R))  # Maximum offset to consider

    # Create arrays for row and column offsets
    row_offsets = np.arange(-max_offset, max_offset + 1)
    col_offsets = np.arange(-max_offset, max_offset + 1)

    # Calculate the grid of distances considering periodic boundaries
    row_distances = np.minimum(np.abs(row_offsets), n_rows - np.abs(row_offsets))**2
    col_distances = np.minimum(np.abs(col_offsets), n_cols - np.abs(col_offsets))**2
    grid_distances = np.sqrt(row_distances[:, np.newaxis] + col_distances)

    # Find neighbors exactly at the radius R
    neighbor_mask = grid_distances == R
    neighbor_offsets = np.argwhere(neighbor_mask) - max_offset

    # Calculate neighbor coordinates with periodic boundary conditions
    neighbors = np.mod(np.array([i, j]) + neighbor_offsets, [n_rows, n_cols])

    return list(map(tuple, neighbors))

def generate_radius_list(max_radius):
    """Generates a list of radii to consider from 1 to max_radius.

    Args:
        max_radius (float): maximum radius

    Returns:
        np.array: array of radii
    """
    radius_set = set()
    for x in range(max_radius + 1):
        for y in range(x + 1):
            r2 = x**2 + y**2
            if r2 <= max_radius**2:
                radius_set.add(r2)
    return np.sqrt(sorted(radius_set))[1:] # Remove the first element (R = 0)

def gap_form(lattice, i, j, radius_list, Moore = False):
    """Applies the gap formation rule to a cell. That is, if the sum of the trees in the neighborhood is less than or equal to the tree in the cell, then the trees in the neighborhood die. The radius of the neighborhood is defined by the inequality.

    Args:
```

```
        lattice (np.matrix): forest lattice
        i (int): x coordinate of the cell
        j (int): y coordinate of the cell
        radius_list (np.array): array of radii to consider
        Moore (bool, optional): Flag to choose which distance to use in gap
            ↪ formation. Defaults to False.

    Returns:
        np.matrix: forest lattice after applying the gap formation rule
    """
    S = lattice[i, j]
    S_nn = 0
    for R in radius_list:
        neighbors = find_nth_moore_layer(lattice, i, j, R) if Moore else
            ↪ find_neighbors_at_R(lattice, i, j, R)
        xs, ys = list(zip(*neighbors)) # From [(x1,y1), ...] to [x1, ...], [
            ↪ y1, ...]
        S_nn += np.sum(lattice[xs, ys])
        if S_nn <= S:
            lattice[xs, ys] = 0
        else:
            break

    return lattice

def death(lattice, pd, Sc, Moore = False):
    """Defines the death rule for the forest. A given tree dies randomly
        ↪ with probability pd. A tree that reaches the maximum size Sc also
        ↪ dies.

    Args:
        lattice (np.matrix): forest lattice
        pd (float): probability of death
        Sc (float): maximum tree size
        Moore (bool, optional): Flag to choose which distance to use in gap
            ↪ formation. Defaults to False.

    Returns:
        np.matrix: forest lattice after applying the death rule
    """

    L = lattice.shape[0]

    # Create masks for cells above Sc and cells that die randomly
    above_sc_mask = (lattice >= Sc)
    death_mask = np.random.random(lattice.shape) <= pd
    gap_mask = above_sc_mask | (lattice > 0) & death_mask # combined mask
    gap_indices = np.argwhere(gap_mask) # array of indices of cells that
        ↪ will form gaps
```

```

    # Generate a list of radii to consider
    max_radius = L // 2  # Maximum radius is half the size of the grid
    radius_list = [i for i in range(1, max_radius + 1)] if Moore else
       ↪ generate_radius_list(max_radius)

    # Apply gap_form to the necessary cells
    for i, j in gap_indices:
        lattice = gap_form(lattice, i, j, radius_list, Moore)

    # Set to zero the cells that died
    lattice[above_sc_mask] = 0
    lattice[(lattice > 0) & death_mask] = 0

    return lattice
```