Group Project

Phase 2

DM857 Introduction to Programming
DS830 Introduction to Programming

Forest fire on a random graph

In the second phase of this project, you will be developing a program to simulate the evolution of a population of automata over time, influenced by selective pressure from the environment. The focus is on simulating an area of land composed of various patches with different terrains. The adjacency between patches will be represented through a planar graph., i.e. each patch of land will identify a site of a graph and neighbouring patches will be those directly connected by a single edge.

Each patch of land will belong to two main categories (Rock and Tree). The area simulated will hence contain at the initial stage a wood (a collection of tree patches) and some non flamable areas (rock patches). The area is also populated by a fixed number of firemans that can move freely on each piece of land.

The aim of the present project is to described the envolution of wildfire in this environment.

The woods have a (small) probaibility of catching fire (autcombustion) and, once lit, fire can propagate to neighbouring tree patches. A tree patches that is devoured by the fire will turn into a rock patch.

The rock patches will not propagate fire, however, over time, they have a small probability of turning into a tree patch.

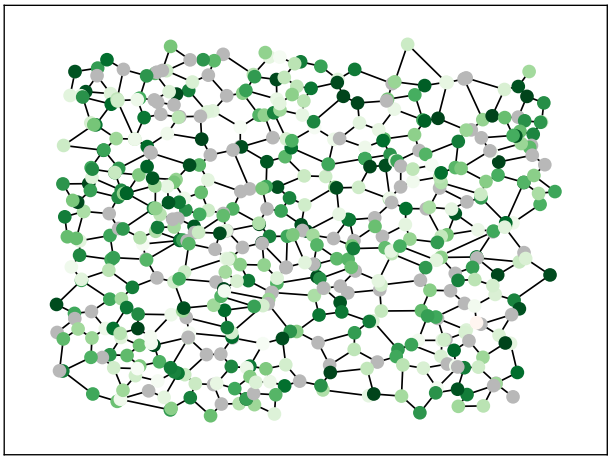

Figure 1: Example of inital configuration, green sites are Treepatches while grey are Rockpatches

The set of parameters to input for running the simulation is has to include at least: the size of the grid or the initial grid read from file, the percentage of the initial tree population, the number of

firefigthers, the set of 3 probabilities (ignition of fire, transmition of fire, forest respawn) and the time limit for the simulation.

However, you may personalise this stage to support more configuration parameters to your liking.

The initial spatial configuration should be either read from file or alghorithmcly generated .

Optional is the possibility of storing and reading also the terrain configurations (i.e. the distribution of Trees and Rocks over the graph).

Code architecture

- Land Representation:
  - Utilize a connected planar graph [1] to describe the land.
  - Use the `graph_helper.py` module for generating and validating planar graphs.
- Object-Oriented Design:
  - Create an object-oriented representation of the simulation.
  - Implement a class structure with a base class (Landpatch) and two subclasses (Treepatch and Rockpatch).
  - Reflect on the specialization of patches into types of terrains.

Class Contracts (minimal requirements):

- Landpatch (Base Class):
  - Method: Return the ID of the next neighbors to the present patch.
- Rockpatch (Subclass):
  - Method: mutate - Allows swapping a Rockpatch with a Treepatch without losing connections to neighbors and associations with firefighters.
- Treepatch (Subclass):
  - Instance Variable: treestats - Identifies the health of the Treepatch [0-256].
  - Method: updateland - Updates the value of treestats due to fire or firefighter action, representing one step of time evolution.
  - Method: mutate - Allows swapping a Treepatch with a Rockpatch without losing connections to neighbors and associations with firefighters.
- Firefighter Class:
  - Carries a variable identifying its skill in extinguishing fires.

[1] A planar graph is a graph that can be drawn on the plane in such a way that its edges intersect only at their endpoints. In other words, it can be drawn in such a way that no edges cross each other.

Project Overview:

Project Components:

1. Configuration

   Objective:
   Set up parameters for random graph generation and simulation settings.

   Tasks:
   Define the parameters for random graph generation:
   Grid size or file source
   Initial tree population percentage.
   Number of firefighters.
   Probabilities for ignition, transmission, and forest respawn.
   Time limit for simulation.

   Deliverables:
   Configurable settings for random graph generation and simulation. Option to read initial spatial configuration from file.
   Optional: Capability to store and read terrain configurations (Trees and Rocks).

2. Simulation

   Objective:
   Run the program to evaluate land geography and evolve the land Tree population over time.

   Tasks:
   Implement a connected planar graph to represent the land.
   Develop the Landpatch base class and subclasses (Treepatch and Rockpatch) adhering to the specified contracts.
   Simulate ignition, transmission, and extinction of fires in the forest.
   Handle the evolution of the forest over time, including transitions from Treepatch to Rockpatch.

   Deliverables:
   Object-oriented representation of the simulation.
   Efficient interaction of land with the Firefighter class.
   Code logic for simulating fire dynamics and time evolution.

3. Reporting

   Objective:

   Present summative results of the simulation.

   Tasks:
   Collect and analyze data during the simulation.
   Generate a comprehensive report summarizing key findings.
   Visualize results, including the spread of fire, changes in tree populations, and firefighter effectiveness.

Deliverables:
Summative report detailing the outcomes of the simulation.
Visual representations, such as graphs or charts, to enhance understanding.

1 Configuration

When the program starts, the user will be presented with a setup interface that allows o define the key parameters for the forest fire simulation. The interface must include at least the following main inputs:

1. Graph Generation
   Options:

   - Read from File: Input the location of the text file containing the graph structure.
     Standard: Each non-empty line represents an edge, identified by two integers separated by a comma.
     Handling Errors: The program will address errors related to file reading (e.g., file not found, incomplete lines).

   - Random Generation: Specify the minimal number of sites for the graph.
     Utilize the provided helper routine for random graph generation.

   Verify Graph: Ensure that the generated graph is both connected and planar.

2. Initial Landscape Pattern
   Options:

   - All Woods: Set up the initial configuration with all patches as tree patches.
   - All Rocks: Set up the initial configuration with all patches as rock patches.
   - Random with Fixed Ratio: Choose a fixed ratio (e.g., 80% woods) for a random initial configuration.

3. Firefighters Population and Abilities Options:

   - Number of Firefighters: Specify the desired number of firefighters in the simulation.
   - Average Skill Level: Define the approximate average skill level for firefighters.
     Note: Individual firefighters may have different skill values.

4. Number of Iteration Steps:
   Provide the option to define the number of iteration steps for the simulation.

The user can navigate the main interface and any secondary interfaces provided. Clear instructions and prompts will guide the user through each step of the configuration process. Error handling messages will be displayed if the user inputs are invalid or if any issues occur during file reading.

2 Simulation

In the simulation stage, you will create a dynamic land based on the inputted graph and evolve it according to the specified rules. Here are the key details:

Evolving the Land

- Firefighter Movement:
  - Firefighters can move only once per evolution step.
  - They will move randomly to a neighboring patch if not involved in extinguishing a fire or no fires are burning an adjacent patch..
  - If a fire is in an adjacent patch, they will move to the neighboring burning patch.
- Tree Patch Dynamics:
  - Tree patches will increase their treestats by a fixed amount each step if no fire is lit on the patch ($+10$).
  - If a Treepatch is on fire, its treestats will decrease each step by a fixed amount (-20).
  - If a Treepatch has treestats < 0, it mutates into a Rockpatch.
  - If fire is burning in a Treepatch it can spread to any adjacent Treepatch with fixed probability (default 30%)
- Rock Patch Dynamics:
  - Each Rockpatch has a possibility of becoming a Treepatch at each step (default 1%).
- Simulation Steps:
  - The simulation will consist of a fixed number of steps defined by the user.
  - Each step produces a visualization of the graph evolution (see the provvided files `visualiser_random_forest_gr` and `usage_material_example.py`)[2].
  - Each frame of the visualization represents a step in the evolution of each patch of the land.
- Visualization:
  The generated visualization will showcase the changing states of patches, including the spread of fire, firefighter movement, and changes in treestats. Refer to the provided example clip `random_forest_fire.mov` on itslearning for a visual representation.

[2] Firefigthers are identified by blue circles drawn around the sites.

3 Reporting

After the simulation concludes, the program will present the user with a comprehensive plot displaying the history of key metrics over the simulation steps.
The plot will include:
Metrics:

- Tree Population Size: The count of Treepatches over simulation steps.
- Rock Population Size: The count of Rockpatches over simulation steps.
- Number of Wildfires: The count of wildfires that occurred during each step.

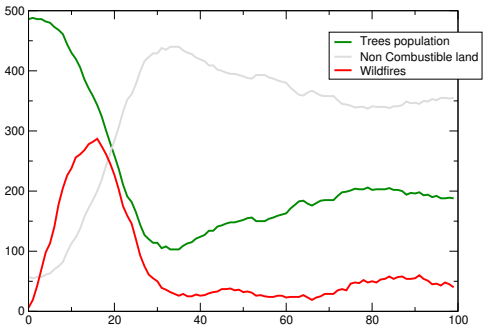

Figure 2: Example report plot

Hand-in

Files

For this phase you must hand in a zip file containing:

- A folder named code containing your implementation, including a module `graph_sim`.
- A PDF document "named_report.pdf" containing your report. The name of the zip file must be the name of your group e.g., group B20.zip.

Setup

- Your solution must contain a module called `graph_forest` which acts as the main module of your program but other than this you may organise your solution as you deem necessary.
- You must not modify any of the modules provided with this assignment (`visualiser_random_forest_graph.py` `graph_helper.py`).
- Your code must be documented and adhere to the common coding conventions and rules of Python and this course.
- It is encouraged the use of the unittest module for testing.
- You can use any module in the standard library of Python, matplotlib numpy. Although you are requested to install you are not allowed to use any routine of networkx , scipy (other third-party modules and packages are not allowed).
- In your report you must describe your implementation explaining and motivating its structure and functioning.
- You must include the code[3] of your solution as an appendix. To save space, you may summarise or omit docstrings, doctests and code for printing configurations and statistics.
- There is no predefined structure for the report besides the code appendix.
- Your report must be written in English and be delivered as a single PDF file printable in black and white (avoid dark background for code listings) and most 15 pages excluding the appendix.
- The report must state the name of the group and list its members (fill name and SDU email, where applicable).

[3] If you prepare your report using LaTeX, there are various solutions for typesetting source code besides plainverbatim, for Python I suggest "minted" or "listings".