SCIF30005: Forest Fire Model

Dr. Laura Ratcliff
laura.ratcliff@bristol.ac.uk

Outline

Recap:

- we've been looking at different cellular automata models
- this includes image blurring and falling sand
- we've also looked at adding OpenMP and MPI parallelisation, and assessing performance

This video:

- we'll be introducing a forest fire model, another type of cellular automaton
- we'll be covering the model rules, but also analysing its behaviour
- we'll finish with a brief discussion on parallelisation

What's next:

- the forest fire model will be the focus of your mini project

Outline

Outline

Forest Fire Model

We start with a square ($N \times N$) grid which represents a forest, with each site in the grid either being:

- empty – 0
- containing a tree (> 0), which can be in different states:
  - **alive** – 1
  - **burning** – 2
  - **burnt** (ash) – 3

We then mimic a forest fire, by:

- randomly generating a forest of (living) trees
- setting some trees on fire
- allowing the fire to burn through the forest

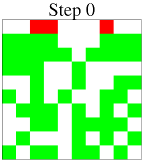
We use a Von Neumann neighbourhood (left, right, above and below the current cell, *not* diagonal cells)

The Rules

- Initialisation:
  - randomly fill the grid with **living** trees with probability p
  - any trees in the top row are set to **burning**

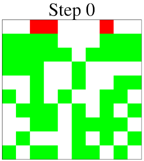
The Rules

- Initialisation:
  - randomly fill the grid with **living** trees with probability p
  - any trees in the top row are set to **burning**



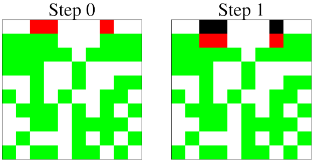
The Rules

- Initialisation:
  - randomly fill the grid with **living** trees with probability p
  - any trees in the top row are set to **burning**
- Iterate the model, so that at each time step t:
  - any **living** tree next to a tree which was **burning** at step $t-1$ will start **burning**
  - any tree which was **burning** at step $t-1$ is now **dead**
  - all other sites stay in the same state



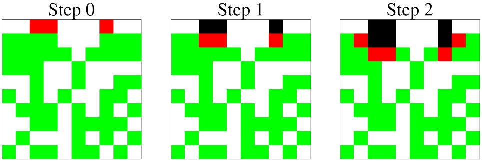
The Rules

- Initialisation:
  - randomly fill the grid with **living** trees with probability p
  - any trees in the top row are set to **burning**
- Iterate the model, so that at each time step t:
  - any **living** tree next to a tree which was **burning** at step $t-1$ will start **burning**
  - any tree which was **burning** at step $t-1$ is now **dead**
  - all other sites stay in the same state



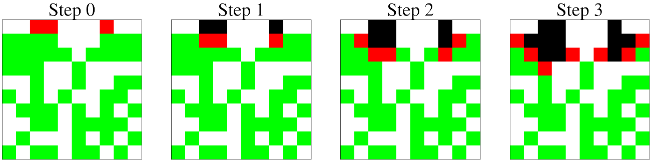
The Rules

- Initialisation:
  - randomly fill the grid with **living** trees with probability p
  - any trees in the top row are set to **burning**
- Iterate the model, so that at each time step t:
  - any **living** tree next to a tree which was **burning** at step $t-1$ will start **burning**
  - any tree which was **burning** at step $t-1$ is now **dead**
  - all other sites stay in the same state



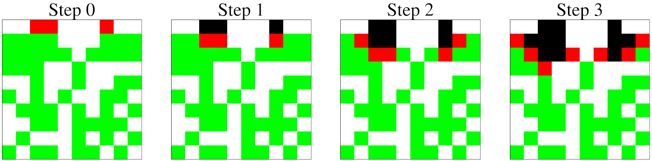
The Rules

- Initialisation:
  - randomly fill the grid with **living** trees with probability p
  - any trees in the top row are set to **burning**
- Iterate the model, so that at each time step t:
  - any **living** tree next to a tree which was **burning** at step $t-1$ will start **burning**
  - any tree which was **burning** at step $t-1$ is now **dead**
  - all other sites stay in the same state



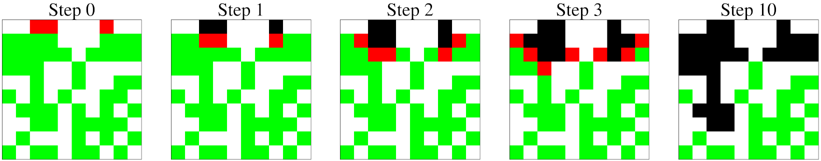
The Rules

- Initialisation:
  - randomly fill the grid with **living** trees with probability p
  - any trees in the top row are set to **burning**
- Iterate the model, so that at each time step t:
  - any **living** tree next to a tree which was **burning** at step $t-1$ will start **burning**
  - any tree which was **burning** at step $t-1$ is now **dead**
  - all other sites stay in the same state
- Finish:
  - the simulation stops when there are no more **burning** trees



The Rules

- Initialisation:
  - randomly fill the grid with **living** trees with probability p
  - any trees in the top row are set to **burning**
- Iterate the model, so that at each time step t:
  - any **living** tree next to a tree which was **burning** at step $t-1$ will start **burning**
  - any tree which was **burning** at step $t-1$ is now **dead**
  - all other sites stay in the same state
- Finish:
  - the simulation stops when there are no more **burning** trees



Variations and Extensions

We can also adapt or extend the model:

- stopping the simulation when the fire reaches the bottom as well as when it burns out
- initialising the fire differently, e.g. trees in the centre of the grid
- filling the grid with trees and using the probability to determine if a tree catches on fire
- introducing directionality, e.g. the fire can go down but not up, or otherwise including wind, e.g. allowed to move 2 cells in a given direction
- allowing the fire to spread to next-nearest neighbours in other ways
- ...

$\rightarrow$ all of these affect both the model behaviour and **the parallel implementation**

But before we think about parallelisation, let's take a look at the model itself...

Outline

Analysing the Model

The forest fire is a type of **percolation** problem – technically, site percolation on a square lattice

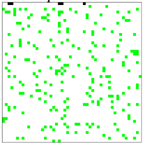
Consider the question: will the fire reach the bottom of the grid?

Analysing the Model

The forest fire is a type of **percolation** problem – technically, site percolation on a square lattice

Consider the question: will the fire reach the bottom of the grid?

$p = 0.1$

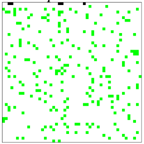


Analysing the Model

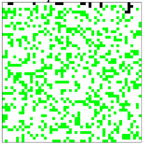
The forest fire is a type of **percolation** problem – technically, site percolation on a square lattice

Consider the question: will the fire reach the bottom of the grid?

$p = 0.1$



$p = 0.3$

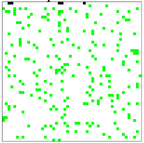


Analysing the Model

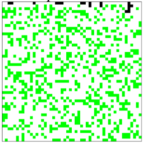
The forest fire is a type of **percolation** problem – technically, site percolation on a square lattice

Consider the question: will the fire reach the bottom of the grid?

$p = 0.1$



$p = 0.3$

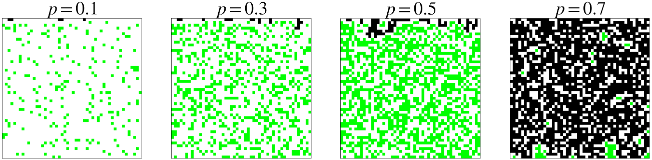


$p = 0.5$

Analysing the Model

The forest fire is a type of **percolation** problem – technically, site percolation on a square lattice

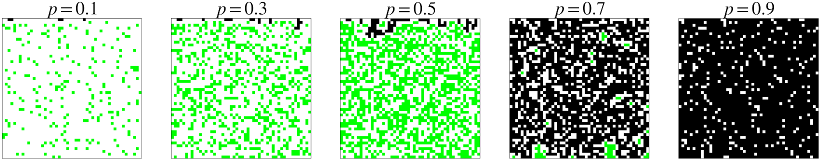
Consider the question: will the fire reach the bottom of the grid?



Analysing the Model

The forest fire is a type of **percolation** problem – technically, site percolation on a square lattice

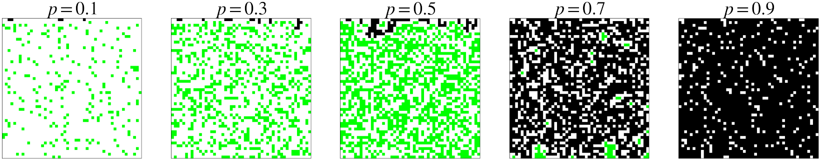
Consider the question: will the fire reach the bottom of the grid?



Analysing the Model

The forest fire is a type of **percolation** problem – technically, site percolation on a square lattice

Consider the question: will the fire reach the bottom of the grid?



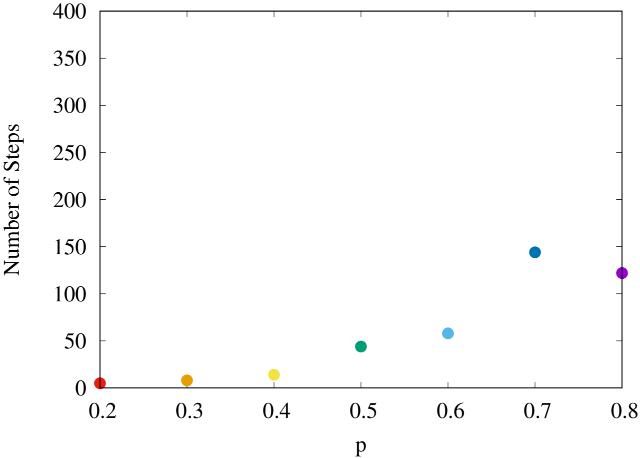
Strong dependence on the probability p, but to understand this, we need to take into account:

- the effect of randomness, i.e. how does behaviour differ with different starting grids, e.g. by averaging over many runs
- the size of the grid, e.g. by testing many grid sizes

$\rightarrow$ need to test *convergence* with respect to number of runs and grid size

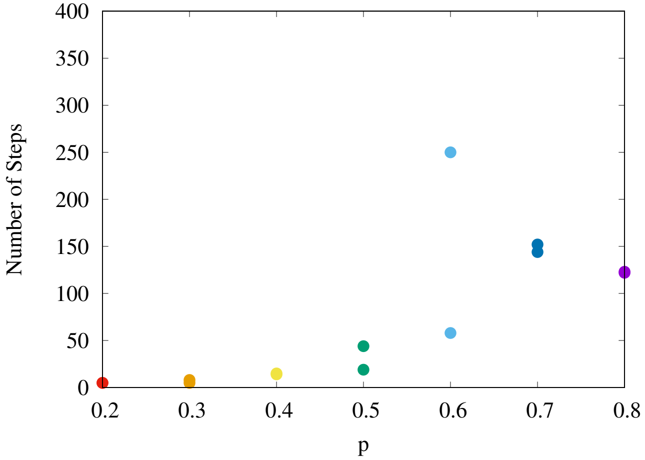
Convergence with Number of Runs

- number of steps before the fire burns out, for $N = 100$
- run the code once for different values of p:



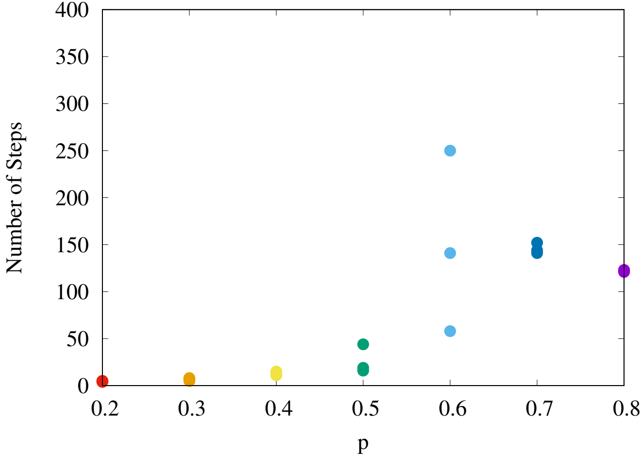
Convergence with Number of Runs

- number of steps before the fire burns out, for $N = 100$
- …and then run a second time…



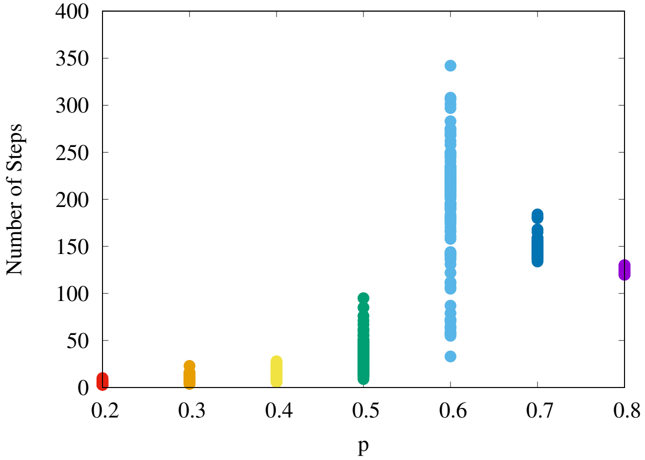
Convergence with Number of Runs

- number of steps before the fire burns out, for $N = 100$
- …or a third time…



Convergence with Number of Runs

- number of steps before the fire burns out, for $N = 100$
- …or a 100 times – results differ due to randomness



Convergence with Number of Runs

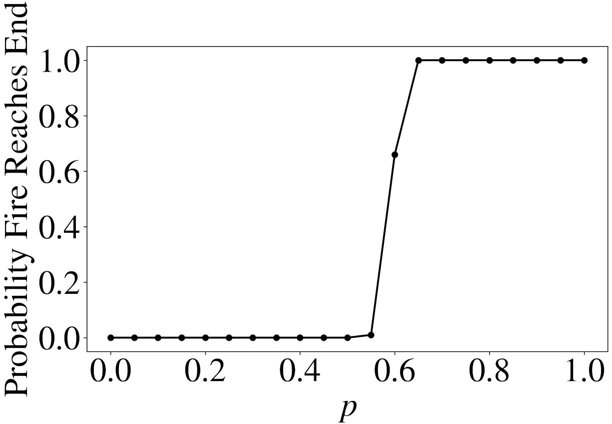
- better to look at *average* values over M runs
- in the ideal world, we would run over every possible starting grid, but this is impractical
- instead we can systematically increase M and see how the average changes
- as we increase M, the average will change less and less, eventually reaching *convergence*
- but the converged value of M might vary depending on the value of p
- $\rightarrow$ not enough to see what happens at e.g. $p = 0.2$ if we want to understand the model behaviour as a whole

Convergence with Grid Size

- the behaviour also changes with grid size
- this is more complicated as the number of steps taken will also increase with the size of the grid
- so would instead look at how the *shape* of the p *vs.* number of steps plot changes with grid size
- can also look at the probability that the fire reaches the end of the grid…

Critical Probability

Interesting change in behaviour around $p = 0.6 \rightarrow$ can see this if we look at probability the fire reaches the bottom before burning out:



$\rightarrow$ there is a **phase transition** at the critical probability $p_c \simeq 0.6$

To explore this behaviour more precisely, need both a large grid and lots of runs $\rightarrow$ computational cost is important!

Convergence Summary

- convergence is something we often need to check when running scientific models
- we can think of our model as being converged if we are ‘close enough’ to the ideal behaviour, e.g. if I ran the model for every possible random starting grid, would I get similar average behaviour to when I run it M times?
- the definition of close enough depends on both the problem and desired accuracy
- unconverged results may be inaccurate or even meaningless
- may need to think both:
  - qualitatively, e.g. has the shape stopped changing?
  - and quantitatively, e.g. has the average number of steps changed by $< x\%$ when averaging over additional runs?
- in practice need to balance cost *vs.* accuracy – want to identify minimum required accuracy for lowest possible cost
- also need to balance cost of running convergence calculations – more isn't always better
- there is more than one way to present convergence data

Outline

Parallelising the Forest Fire Model

We have two factors to consider:

(1) the need to do **lots of runs**

(2) the need to consider **large grids**

Problem 1 is an example of an **embarrassingly parallel** problem, e.g.

- run the model with different random seeds on different cores
- only thing to communicate is the final outcome (e.g. number of steps before burning out, whether or not it reaches the end)

$\rightarrow$ leads to almost perfect performance, i.e. time for 2 runs on 2 cores $\simeq$ 1 run on 1 core

We will focus on problem 2, i.e. how to speed up a single run

General MPI Tips

- there are lots of different strategies you can use to implement MPI
- generally, there are 2 broad questions to ask:
  - how am I going to divide the data among tasks, i.e. which tasks will have what information?
  - what and how am I going to communicate between tasks, i.e. what MPI calls should I use to send/receive the data?
- often more than one strategy is possible, but sometimes one approach is better than another (e.g. it means sending less data)
- there is also sometimes a trade-off between ease of implementation and efficiency
- correctness is more important than complexity – start with whichever approach seems easiest to you

Inspiration

- the forest fire model has various features in common with the other models we've looked at so far:
  - we need to iterate over a grid $\rightarrow$ both image blurring and falling sand
  - we only need information from immediate neighbours $\rightarrow$ both image blurring and falling sand
  - we need to keep track of both the current and previous grids $\rightarrow$ image blurring
  - the model ends when a specific criterion is met (the fire stops burning) $\rightarrow$ falling sand
- MPI strategy: we looked at two MPI versions of the image blurring model
- $\rightarrow$ you should be able to use what you've learned from the previous workshops to help with the project

Outline

Summary

In this video, we introduced a forest fire model:

- we looked at the basic rules
- we talked about some extensions to the model
- we looked at how the model behaves
- we talked about convergence
- we discussed parallelisation

What's next:

- mini project on the forest fire model – instructions on Blackboard