Networks Project
A Growing Network Model

CID: 01881584

19th February 2023

Abstract: You may want to write a concise abstract that briefly puts the work into context, explains what was done, how it was done, the main results, the conclusions you could draw and the implications hereof.

Word count: 282 words in report (excluding front page, figure captions, table captions, acknowledgement and bibliography).

1 Implementation of the Oslo model.

1.1 TASK 1

First, I created a class *oslo_lattice*, which implements the boundary-driven Oslo model. The algorithm itself is fairly simple, mirroring the steps outlined in the project notes. However, one non-trivial thing was the algorithm that deals specifically with relaxation, as it must ensure there are no supercritical sites before driving the system with another grain.

The obvious, *naive* algorithm, would be to keep iterating through all L sites checking for supercriticality, and if a full iteration occurs without encountering a simple supercritical site, relaxation terminates. However, I have devised an optimized algorithm, which I named the *ceilidh algorithm*, which works like this:

$currentIndex \leftarrow 0$
$maxIndexAffected \leftarrow 0$
while $currentIndex < L$ AND $currentIndex \leq maxIndexAffected$ **do**
  if site at $currentIndex$ is supercritical **then**
    relax current site
    $currentIndex \leftarrow currentIndex - 1$
    $maxIndexAffected \leftarrow \max(maxIndexAffected, currentIndex + 1)$
  else
    $currentIndex \leftarrow currentIndex + 1$
  end if
end while

Instead of repeated full iteration, the algorithm keeps making single "steps forward" and "steps back", since the relaxation of a site at index i can only cause supercriticalities on sites with indices ranging from $i - 1$ to $i + 1$, that is, on itself and its direct neighbours. The $maxIndexAffected$ variable applies the same principle to prevent the algorithm from needlessly iterating through the entire lattice when the last supercriticality gets resolved.

Tests:

- **Matching seed should yield deterministic behaviour.** I initialized several instances of the lattice with the same exact initial properties and hardcoded the seed for the random.py package to reset to a constant value before simulating every instance, and ran the simulation with 100 steps on all of them. I expected the states of all instances to be the same. This turned out to be true: all states across all instances were the same after the same amount of steps when resetting the seed each time.
  This assures that the stochastic behaviour of the programme is effected entiredly by the random.py package, usage of which can be easily monitored.

- **For any state, $z_i \leq z_i^{\text{th}}$ always.** I created 10 instances with $L = 100$ and simulated 10-100 steps on each. Then I checked whether $z_i \leq z_i^{\text{th}}$ for all i for their states. If this weren't true, it would mean there are supercritical sites not being relaxed before starting the next step.
  The test concluded well: there were no violations of this principle detected. This assures the relaxation algorithm used is valid.

- **The average pile heights match the values in the project notes.** After running the simulation on 10 instances and averaging the values after 500 and 5000 steps respectively, the average value of h_1 was found to be 26.7 for $L = 16$ and 53.5 for $L = 32$. These values match the ones given.

- **Performance of the relaxation algorithm used shouldn't be worse than the naive algorithm.** In fact, since each found supercriticality increases the number of sites to be checked by L for the naive algorithm and by 2 for the ceilidh algorithm, the time complexity of ceilidh should be less by one power of L than that of the naive algorithm. Indeed, running a benchmark test confirms this: the ceilidh algorithm always outperforms the naive algorithm, and the ratio of runtimes increases as L increases. TODO INSERT PLOT.

1.2 References

References probably not really needed for a project but you might have to cite something. Follow the standard advice on plagiarism. For LaTeX issues and bibliographies see this file `mh2920_01881584_complexity_project_report.tex` for comments on how (not) to do it. For instance, these two publications are great [2, 3].

NB: This page (acknowledgement and bibliography) does not count towards the 2500 word limit nor the 16 page limit.

Acknowledgements

Not required but you might want to thank A.Demonstrator for help.

References

[1] T.S. Evans, *Slides for Networks course*, Physics Dept., Imperial College London, 2014, downloaded from Blackboard 2nd February 2015.

[2] K.Christensen and N.Maloney, *Complexity and Criticality*, Imperial College Press, London, 2005.

[3] T.S. Evans and R. Lambiotte, "Line Graphs, Link Partitions and Overlapping Communities", Phys.Rev.E. **80** (2009) 016202.