Complexity & Networks Course (PHYS60010)

Network Project Brief

Tim Evans
Centre for Complexity Science, Imperial College London
17th February 2023

Brief for the Network Project Report

Part of the challenge in writing the report is to satisfy these requirements in this "brief". Marks will be deducted if the Networks project report fails to satisfy the following properties.

Structure:
Keep the headings and their numbering exactly as given in `NetworkProjectReportTemplate.pdf`, keeping them in the order given. The title page should also contain the information indicated in that file (Title, CID, Date, Abstract, but **no name**).

Word count:
There must be no more than **2500 words** in the report excluding front page with any abstract, figure captions, table captions, acknowledgement and bibliography.
An estimate of the number of words used **must** be given at the front of each report.

Page limit:
Reports may not exceed **16 A4 size pages** in total, excluding front page and ultimate acknowledgement and bibliography page.

Appendices:
Appendices are **not** allowed.

Fonts:
Reports must use a **12pt font**.

Margins:
Minimum `2.5cm left and right margins` as used in the template.

Marks:
Marks shown on this paper are indicative of those the Examiners anticipate assigning.

Template:
The file `NetworkProjectReportTemplate.pdf` gives the required structure for the Networks project reports. Other files provided for convenience only.

General Comments

The rules above constitute the "brief" for this project. When submitting a commercial bid or a grant proposal, the client will normally require all the rules or obeyed, otherwise the bid is sometimes not even accepted. Here the project will be accepted but marks will be deducted under the Professional Skills section if the rules of the brief are not followed.

See `NetworkprojectReportTemplates.pdf` on Blackboard which gives the structure required for your report. You can also copy and edit the other files given there as a starting point for your own report file. The templates also include further prompts and a breakdown of the marks. The report should be a readable document which contains answers to the specific questions and issues raised in these notes and in the template. For instance very brief descriptions of the aims and a very brief conclusion may be appropriate.

I am looking for the report to demonstrate that the author has understood the issues and problems. That may require some initiative to go beyond the specific prompts in these notes. Effective programming to get good results along with clear explanations in well presented reports is what is needed. The bulk of marks are for working code, good theoretical derivations, sensible data handling, interpretations and explanations of results. This means that sophisticated programming is only rewarded indirectly provided you wrote and tested the code in reasonable time (i.e. if it enables you to produce results on very large scales quickly).

Some additional suggestions:

- Answer the questions raised in these notes. Provide text, plots etc. to address points with marks.
- Choose your plots carefully to illustrate typical results, to convey important information, and to make relevant comparisons. If you can not see a feature of interest in your plat, perhaps you should use a different plot.
- Make sure you label figures appropriately and give them proper captions that explain what is plotted, what you see, emphasising the point(s) you want to convey to the reader without any reference to the main body of the text.
- Explain all symbols used in texts and plots such as any points, lines, error bars, and variable names.
- Your source code should be commented appropriately and easy to read. You do not need to explain the programming in great detail, but basic information on the origin and nature of the code used, brief comments on any 'tricks' used and a quick summary of strategies used to test the code may all be appropriate.

Plagiarism

Plagiarism may be defined[1] as "Plagiarism is when you copy someone else's work, words or ideas and use these in your coursework, thesis, report, etc., and do not acknowledge that you have done this. This can be either intentional or unintentional". Plagiarism applies to computer code as much as it does to text. So if you copy code, such as the log-binning code provided on Blackboard, cite it e.g. by adding a comment just above the code you use, indicating the URL of the source. It is great if students discuss the project and the code between themselves, just make sure you write your own version of your project, produce your own plots from your own code. For instance you must be able to explain your code if needed. You *must* submit your code as part of the project and both code and text report are checked with appropriate plagiarism software. We unfortunately find significant plagiarism in both text and/or code (not simple mistakes) a few cases every year and penalties are imposed. If you find yourself in any difficulties, use the student support available as early as possible, *in confidence if needed*, such as your personal tutor.

[1] Quote from "Plagiarism awareness for undergraduates" provided by the Imperial College Library who can provide further support and advice.

Submission

For each project, two copies of the **report** and the **complete source code** must be submitted through two links given on the `PHYS60010 - Complexity and Networks 2022-2023` Blackboard web site. If you do not have access to Complexity & Networks course on Blackboard, please contact the student office well in advance of the project deadlines.

- One copy of the report must be submitted in `pdf` **format only** using the specific Networks project link given on Blackboard This component of your submission will be plagiarism checked via *TurnItIn*. Use your CID number to identify your report. Do **not** put your name on the report. Likewise, please name the `pdf` file containing your report using your CID number, something like
  ***CID*-NetworksReport.pdf**, e.g. `012345678-NetworksReport.pdf`.

  Note that every year we have a small number of students who fail to submit a report to Turnitin successfully but who are not aware of this failure. Students that have submitted to Turnitin successfully will be notified on the screen immediately that their submission has been successful and they will receive a receipt emailed to their registered email address. If you do not receive a receipt, you should assume that your submission has not been successful. Students can check the submission inbox to see if their submission is there or not. You should try the system well in advance of submitting your final version. If a problem persists, contact the ICT Service Desk.

- For each of two projects, your complete collection of source code must also be uploaded along with a SECOND copy of the same `pdf` file of the report, all in a single `zip` file. Source files must be in standard human readable format e.g. you must convert Jupyter notebook `.ipynb` files into Python script `.py` files. Source files includes those used to analyse the data and to produce any plots in the report. Other files (e.g. a README file is good practice) may be included but are not required. In particular, *we do not require data files* which can produce a large zip file rejected by the submission system. Ask if in doubt. The code is also checked for plagiarism using specialised software.

  - The name of the zip file should include your CID and project,
    CID-NetworksCode.zip e.g. `012345678-NetworksCode.zip`
  - All files required to run the code, other than standard libraries, must be present. We need to see how you produced your plots and results that you reported in your text.
  - Put a copy of your report `pdf` file in these ZIP files alongside the code. This copy acts as a backup in case of any problems with TurnitIn.

Deadline: The deadline for submitting your Networks Project Report and your code is

MONDAY 27th MARCH 2023 at 10:00 London time

Late Submission of Projects

Standard departmental policy at the time of submission for work submitted late will be applied. For instance, at the time of writing, I understand that the late submission rules means that a project report submitted more than 24 hours late will incur zero marks unless you have mitigating circumstances.

Typical problems associated with computers are not accepted as valid excuses so do not leave submission to the last minute. Since we all expect problems with computers that means you must leave enough time to allow for that. You may make more than one submission, only the most recent one is retained for marking. I **strongly** advise you to post an early version as a backup and to test that the system works for you. Evidence of a test submission on Blackboard will support your case should there be later technical issues. Note the system imposes a sharp cutoff in time, late submission by even one second is not possible on the primary submission portals. Both report and code are checked for plagiarism (including any allowed late submissions) and this includes the use of automated systems.

If you have good reasons for a late submission you will need to use the mitigating circumstances procedure. At the time of writing you *must* contact the Senior Tutor providing the details of the relevant mitigating circumstance preventing you from submitting the course work on time. You must submit the relevant form together with corroborating evidence to the Senior Tutor, in advance of the deadline whenever possible. The mitigation advisory panel (not the lecturer) will make a decision on possible extension of deadline or other method of mitigation.

If in any doubt about the criteria for mitigating circumstances, ask and ask as early as possible. Remember health issues are not the only grounds for making a request for mitigation. If in *any* doubt (we mean this) please talk to your Personal Tutor or email the Senior Tutor and arrange a meeting.

So if there are mitigating circumstances, do initiate those procedures as soon as possible. I am not formally part of the mitigating circumstances procedure but do contact me as well as early as possible if there are any problems, formal or informal, and I can also give advice.

As a *last resort* only, you may send copies of your report and code to me by email. You should also email me directly to explain the nature of the problem. However, I am not allowed to mark items which don't follow the regulations and I will have to impose the same strict cutoffs in time as the automatic systems unless Mitigating Circumstances procedures have been followed. To be fair to all students on the course, I will try to make the same rules are applied to everyone.

Marks

The project folder on Blackboard contains a pdf `NetworkProjectReportTemplate.pdf` with a breakdown of marks given as a guide, the final mark scheme may differ. This pdf document also contains specific prompts along with every mark. This pdf document is a very useful guide when working on the project so I recommend strongly that you keep a copy of this template available while working on the project

The two projects (Complexity Project and Networks Project) carry equal weight. The projects will each contribute 45% of the final mark for the Complexity & Networks course with a 2.5% contribution from each of the four online tests.

Also note that once the **Complexity** Project Report is submitted, you are committed to the whole course. That is, you will be given a mark for the whole course if we have a Complexity Project report.

Numerical Tools

A network library is the simplest way to approach the project but it is not essential. In this project we are focusing only on the degree distribution. So this project needs only a tiny part of a network library and you could consider alternatives if you are confident[2]. Discussing these options with demonstrators in the computer lab is always a good starting point.

I recommend that you use `python` and, if needed, the `networkx` library. Both are on the Physics PCs (e.g. see 1st and 2nd year lab information on Blackboard) and are easy and free to

[2] See my `C++` code `simplegraph` for an example of how I might create my own basic network functions.

install on any type of machine. The `python` language is easy to learn and to use, not the fastest but quick enough for this project. Many other standard libraries, especially `numpy`, `scipy` and `matplotlib`, deal with all statistical and plotting needs and are installed on the Physics PCs and most python installations. These libraries are well known with a lot of online support (and free) making `python` a complete solution. It might (just) be worth learning `python` for this project. An example code using `python` and `networkx` is provided on Blackboard (`ersimpledegreedistribution.py`) to help you get started. Note that if you use Jupyter notebook `.ipynb` files, these must be converted into the standard readable Python script `.py` files to upload as part of the project submission. You can do this simply under the `File` menu selecting the Download as option and then the `Python (.py)` option within the final list.

In terms of the number of nodes N, I managed to reach $N = 10^5$ with my `python` code for low m around 3, running in a few minutes on a typical desktop. Reaching $N = 10^6$ would be excellent (some students have reached that) while $N = 10^4$ is probably the least number you should aim for.

Also note that log binning for integer values is often used by students for this project. The code used for the Complexity project (`logbin-2020.py` at time of writing) may be used here (citing the source of this code of course). As with any external code used, read the comments, look at the examples, and understand what the code does before you start using it. It is acceptable to use external code, such as the log binning routines provided, as long as any code written by others is properly cited in the comments in your code as this reused code will be detected by our plagiarism software. Our primary interest is to see how students use software to investigate a problem and we are not directly interested in the code you used. We can and will look at the code if there are any issues.

Description of Network Project

Introduction

In this project you will study the degree distribution produced by some simple models of a growing network. The first model considered here was described by Barabási and Albert (1999) [1] but it is identical to an undirected version of Price's 1976 citation network model [2]. In terms of the degree distribution, these models are in turn completely equivalent to the models of Yule (1925) [3] and Simon (1955) [4]. The central idea is that fat tails seen in many distributions in various contexts could be explained in terms of a "rich get richer" principle. This concept goes back to the 19th Century (at least) when Pareto noted that 80% of the land in Italy was owned by just 20% of the population. The idea is so universal that it occurs in many different guises — the "Pareto principle", the "80–20 rule" or even then Matthew effect (Matthew's gospel "For everyone who has will be given more"). For simplicity we will use the terminology of Barabási and Albert [1] who talk about PREFERENTIAL ATTACHMENT and we will refer to this model as the BA MODEL. This is so our language matches that of most recent network literature.

Definition of Model

The model to be used in the project is defined as follows.

1. Set up initial network at time t_0, a graph $\mathcal{G}_0$.
2. Increment time $t \rightarrow t+1$.
3. Add one new vertex.
4. Add m edges as follows:
   - Connect one end of the new edge to the new vertex
   - Connect the other end of each new edge to an existing vertex chosen with probability Π. This will be specified in different ways in the different tasks.
5. Repeat from 2 until reach final number of N vertices in the network.

Overall Objective

What are we trying to create with our simulation? We are trying to compare a numerical simulation against the corresponding mathematical equations, specifically the equations for the degree distribution developed in the lectures and in this project.

However numerical simulations are rarely exact, they almost always represent some 'acceptable' approximation to the exact case we are interested in. Likewise, 'exact' mathematical results are often only valid under some approximation, are for some extreme limit (infinite sized systems for instance) and are for some simplified version of a model which allows for mathematical analysis. The definition of the model given here is not precise enough so this required you to make some sensible choices guided by the limitations in both the numerical and mathematical analysis. So the second question is, if we made some compromises in the numerics and some simplifications in producing the mathematical description, what were these approximations, were they acceptable and how do they effect our comparison between theoretical and numerical results?[3]

There is sometimes no one perfect or correct answer to specific issues that occur in this project. I am really looking for students to think about what they are doing, using whatever resources they can to find answers. When puzzled, try the 5 B's in the following order — brain (you), board (e.g. these notes, lecture notes), book (`google`, `stackoverflow`), buddy (discuss issues between yourselves), boss (demonstrators in the lab). Students should then flag the issues in their report, explain their approach to these points and justify their solutions. Use your Imperial Physics mind set to tackle this project.

So highlight any ambiguities in moving from imprecise outline of the model given above to mathematical equations and numerical implementation, and suggest an acceptable solution.

A good way to see what the issues are might be to work out a few iterations of the model by hand, drawing out the graph, keeping the lists by hand that your computer code will keep, even picking some random numbers yourself.

1 Phase 1: Preferential Attachment Π_{pa} (PA)

In Phase 1 the probability Π for choosing the existing vertex at one end of a new edge is $\Pi = \Pi_{\mathrm{pa}}(k) \propto k$, that is we attach to an existing vertex with probability proportional to its degree.

1. First write a programme to create one instance of the BA model. Do *NOT* just call a library routine to generate the whole BA network (most libraries have one) as this will gain no marks. In any case it may not implement the model in the way you need and in later parts of the project you may need to alter the model. You may use other library tools (to generate an initial graph, to bin data) and that is fine, just be careful that the library function does what you actually want. You may also learn by looking at how others code this problem, cite your source in your code's comments, and generally apply the same rules about plagiarism as you would when writing text.
   One problem you will face in writing your own code is choosing a vertex in proportion to its degree. Some approaches to this issue are:-

   - your network package/graph library has a function that chooses an edge at random. Then choose an end of this edge at random.
   - keep a list[4] of existing edges and choose one at random (or again a library might do this for you), then choose one end of that edge at random.
   - every time you connect an edge to a vertex, add that vertex to a special "attachment" list. Choosing an entry at random from this attachment list gives you a vertex in proportion to its degree (why?).

   Whatever your approach, you will need to explain why it achieves the desired result. You will need to describe how you did this in your report.

[3] Due to lack of time I am not asking you the most important question — how does theory or numerical modelling compare against real data?

[4] This, as with many other aspects of this project, may be programmed using variable length arrays or similar structures. It is worth learning about them in your chosen language. For instance in `C++` look at the 'containers' of the 'STL', in particular try a 'vector'. In `python` they are called 'lists' and are central to the language (there are no fixed size arrays in the basic language). `java` has 'ArrayList' in its 'collections' class. One you master them you will find they come with useful methods (a sorting method is usually built in) and there are other useful structures which work in a similar way e.g. they all have a set structure so no need for you to check for duplication in these.

Now check your programme is working correctly. This is very important. You need to explain in your report why you think your numerical results are correct.

2. Find the best theoretical form for the degree distribution you can find. You may start from the master equation given in the lectures and on the problem sheet. The continuous version derived in the lectures is the minimal required ($p_\infty(k) \propto k^{-3}$) but problem sheets indicate how to get a more precise answer needed for full marks. You should check that your solution obeys all the theoretical constraints on the degree distribution.

3. Now write some code which will find the degree distribution from the numerical simulation. Do this for fixed N (the number of vertices in the network) but varying m (number of edges added per step) and explain why you chose the values you studied.

   Use these results to make a comparison of the numerical degree distribution with your theoretical predictions, both visually and statistically. Think of what to plot which makes this comparison clear. You may need to improve your knowledge of statistics by finding a standard test appropriate for comparing data to a model. All such tests will be implemented in standard library functions so you do not have to write your own code for this. You might consider if any of the following are useful in this case: the coefficient of determination R^2 (as produced by Excel in linear plots), Pearson's χ^2 test, using the general χ^2 function (or it's cousin, the reduced χ^2 function), the KS (Kolmogorov-Smirnov) test. You should yourself ask if your test is appropriate and what the answer means[5]. Do you expect a perfect fit for all value of degree k? What problems does the fat-tail of this distribution cause? Try to find a way to improve the problems caused by the fat tail.

4. Now investigate the finite size effect by looking at a single value of m (justify your choice for m) but varying N. If you can, do this with several numerical runs for each set of values to improve statistics.

   First give your best theoretical estimate for the largest expected degree k_1. Then use this to compare against numerical values, in particular variation with N. Can you estimate errors in any measurements of k_1?

   Look at possible data collapse in the degree distribution. You should focus on the tail (why?) and on particular on deviations from the theoretical prediction for an infinite sized system (try $p_{\text{data}}(k)/p_{\text{theory}}(k)$). You will need good statistics. Can you see simple behaviour in any deviations from the infinite time large degree limit of the degree distribution?

In your report you should:-

- Describe how you implemented the BA model numerically and explain why your approach works. Note any issues or special cases which arise doing this numerically, how you dealt with these and your justification for your numerical solution. What initial graph did you use? What type of network are you creating? Ask what choices have you made numerically and ask how they compare with those made in your theoretical derivations. Justify the choices you make.

  Explain how you know that your programme is working correctly.

  Describe the parameters your programme needs, the values you chose, and why.

- Give your best theoretical derivation for the form of the degree distribution in the long time limit.

[5] Telling me some number is 0.9 means little on its own, how do I know if that is actually a poor result and we needed 0.99 for a good fit?

- Show how you compared your theoretical result to your numerical data for fixed N but different m. Is it a good fit? How did you arrive at your conclusion, visually and statistically? How did you deal with any problems that a fat tailed distribution causes?

- Give your best theoretical estimate of how the largest expected degree, k_1 (subscript 1 indicating the degree of the vertex ranked first by degree size) depends on the number of vertices N in a finite size system. You should give a numerical study of the behaviour of k_1 and N. Estimate uncertainties/errors where possible.

  You should describe how you tried to use an understanding of the k_1 scale to look for any finite size effects in the tail of the distribution, describing any results found but further mathematical investigation of finite size effects is not required (it is extremely hard to do).

2 Phase 2: Random Attachment Π_{ra} (RA)

Repeat the numerical and theoretical analysis of Phase 1 but now assume that all existing vertices are equal when attaching new edges, i.e. $\Pi = \Pi_{\mathrm{ra}} \propto 1$. For the theoretical side you will need to extend what we have seen in the lectures and problem sheets. If you're uncertain about the theoretical derivations, you could also try looking at the numerical results first and see if that suggests the functional form you are looking for. I would also suggest you might try the continuous k differential equation approach first and then try find an exact solution for the discrete k case.

The information required in the report exactly as in phase 1. However as only small changes will be needed numerically, most of the credit in this phase is for deriving the theoretical results, analysing the numerical results, and comparing the two. In your report you should

- Give your best theoretical derivation for the form of the degree distribution in the long time limit. Check the properties of your theoretical solution. What is the largest degree k_1 in this case?

- Compare your theoretical result for the degree distribution to your data. Is it a good fit and how did you arrive at your conclusion? What parameter values did you choose and why? What is the largest degree k_1 numerically for this model?

3 Phase 3: Existing Vertices Model

We will now change the algorithm so that when we add new edges sometimes we attach both ends of the new edge to existing vertices. The idea is that if this is a model for links between web sites (the context of the original Barabási-Albert paper [1]) then web sites continue to add links after the web site is first created.

The model is now as follows. As before each time step, we add one new vertex. We then add m new edges but in one of two different ways:

- For r of the edges, connect one end of the new edge to the new vertex and the other end to an existing vertex chosen with probability Π_1.

- For the remaining $(m - r)$ of the edges connect *both* ends of the new edge to two existing vertices, each vertex chosen with probability Π_2.

Note that we now have a new parameter $r \in \{0, 1, \ldots, m\}$. You can solve this model analytically[6] for arbitrary r. For simplicity, students should look at the following case numerically:

- Choose $r = m/3$, so at each time step one third of the new edges link the new vertex to an existing vertex, and the remaining two thirds of the time, the new edges link two existing vertices.

- Choose $\Pi_1 = \Pi_{\text{ra}}$ and $\Pi_2 = \Pi_{\text{pa}}$, i.e. whenever we add an edge from a new to an existing vertex we will choose the existing vertex using random attachment but we will use preferential attachment to find two existing vertices when a new edge is added between two existing vertices.

If you're uncertain about the theoretical derivations, you could also try looking at the numerical results first and see if that suggests the functional form you are looking for. I would also suggest you might try the continuous k differential equation approach first and then try to find an exact solution for the discrete k case.

In your report you should

- Give your best (most general) theoretical derivation for the form of the degree distribution in the long time limit. Check your solution.

- Explain how your numerical results compared to your theoretical result. Do this at fixed N and at least two values of m. Is it a good fit and how did you arrive at your conclusion about the fit? Analyse the finite size effects by looking for possible data collapse.

Professional Skills:

Your report should be:

- written in clear comprehensible English,

- well presented,

- well organised and following the brief.

This is highly subjective but generally most reports that follow the brief score well on these aspects. Many of the problems come from plots which might have: small unreadable fonts used for labels, labels used for variables are different from those in the text (they reflect the naming scheme used in the programme), unexplained equations of some fit with unrealistic accuracy in unexplained coefficients. Remember, you also have to satisfy the basic criteria of the brief (margins, length, font size etc) or marks will be deducted here.

Hints and Tips

Trust your training and think!

These hints and tips are based on previous reports produced for this project. Read and follow the production brief. Use the mark scheme in `NetworkprojectReportTemplates.pdf` to help you decide how much time to spend on any one part. Too often we saw plots that did not live up to the basic standards which have been emphasised throughout your degree. Ask

[6] In principle both Π_1 and Π_2 can be mixtures of preferential and random attachment and this is still solvable analytically but that is too many options to investigate numerically in this project.

if the plot you produce conveys what you want to say effectively? If a feature is too small to see then perhaps your visualisation should be improved. Often insufficient information is given in a figure or its caption, the parameters used are not all specified or even the model used for the plot is not mentioned. Error analysis was often poor — are the error bars in plots defined and are they appropriate, how do you know if a given measure of uncertainty is large or small for your chosen measure of error? There are often what appears to be contradictions between plots and statistics given in reports.

References

[1] A.-L. Barabási and R. Albert, *Emergence of scaling in random networks Science*, **286** 173 (1999).

[2] D. J. de S. Price, *A general theory of bibliometric and other cumulative advantage processes* J.Amer.Soc.Inform.Sci. **27** 292–306 (1976).

[3] G. U. Yule, *A mathematical theory of evolution based on the conclusions of Dr. J.C. Willis*, F.R.S. Phil. Trans. B, **21-87** 21–87 (1925).

[4] H.A. Simon, *On a class of skew distribution functions*, Biometrica, **42** 425 (1955).