Network Complexity

Thesis Subtitle

Yipei Zhao

A dissertation presented for the degree of
Master of Science

MSc Data Analytics
Aston University
United Kingdom
05 September 2021

1 Introduction

In my literature review, several complexity measures were introduced, includes the theory and the difference between them.

1.1 Random graphs

We have many real networks in the actual world, but defining or observing every one of them is not feasible. For simulation and comparison reason, network scientists introduced the idea of random network. They are also known as Erdos-Renyi network in honour of two mathematicians: Pal Erdos and Alfred Renyi. They have important contributions to understand the properties of a random network[2]
There are two definitions of a random graph:

- $G(n, p)$ network. A graph with n nodes will be initialised, there will be at most $(n)(n-1)/2$ edges. Every edge will have a probability p to be instantiate. This approach brings a randomness propertyto the graph; number of edges m. A $G(n, p)$ graph returns a fixed n but a different m everytime. The expectation of m is $p(n)(n-1)/2.$, but usually comes with a small difference.
- $G(n, m)$ or $G(n, L)$ network. A graph with n nodes will be initialised, m/L edges will be connected from a random node to another random node. Due to the non-randomness of $G(n, m)$ networks, they are used to simulate the behaviour of a random network in this thesis.

1.1.1 Random graphs property

2 Methods

2.1 Implemeted methods

In the literature review, 9 methods were introduced, 7 methods were succesfully implemented and tested with a new method MAri based on MAg. The implemented methods are:

- Subgraph measures:
  - $C_{1e,st}$
  - $C_{1e,spec}$
  - $C_{2e,spec}$
- OdC (Entropy measure)
- Product measures:

- MAg
- Cr
- Ce
- $MAri$

2.2 MA_{RI}

The MAg measure is a product measure, which distributes higher complexity to graphs with medium number of edges and lower complexity at both tails. Using the product of redundancy R and mutual information I, with a normalisation formular, MAg is defined as[3]:

$$
\begin{aligned}
R &= \frac{1}{m}\sum_{i,j>i} ln(d_i d_j) \\
I &= \frac{1}{m}\sum_{i,j>i} ln(\frac{2m}{d_i d_j}) \\
MA_R &= 4(\frac{R - R_{path}}{R_{clique} - R_{path}})(1 - \frac{R - R_{path}}{R_{clique} - R_{path}}) \\
MA_I &= 4(\frac{I - I_{clique}}{I_{path} - I_{clique}})(1 - \frac{I - I_{clique}}{I_{path} - I_{clique}}) \\
MA_g &= MA_R * MA_I
\end{aligned}
\tag{1}
$$

$R_{path}, R_{clique}, I_{path}$ and I_{clique} represent lowest redundacy, highest redundancy, highest mutual information and lowest mutual information of graphs with fixed m and n respectively. The equations can be found in the literature review.
Even though MA_g is well defined and normalised, a problem with the MA_g is it does not assign highest complexities to graphs with the most middle number of edges for small graphs. The highest complexity is reached at $n^{1.5}$, instead of $\frac{n(n-1)}{4}$.

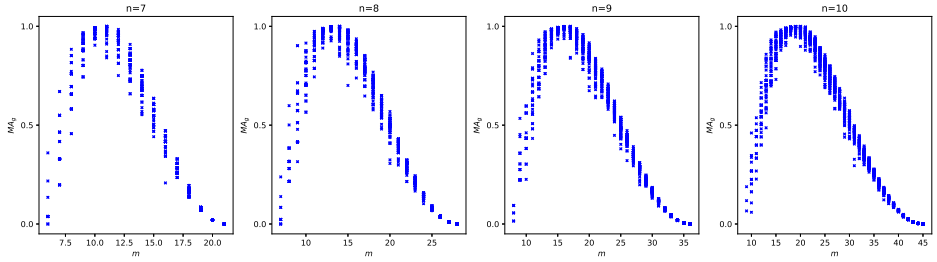

Figure 1: MA_g complexity values of graphs with n=7,8,9,10 nodes

Hence, we may change the normalisation term such that the highest complexity is shifted more towards the middle number of edges. The normalisation term becomes:

$$MA_{RI} = 4(\frac{R - R_{path}}{R_{clique} - R_{path}})(\frac{I - I_{clique}}{I_{path} - I_{clique}}) \tag{2}$$

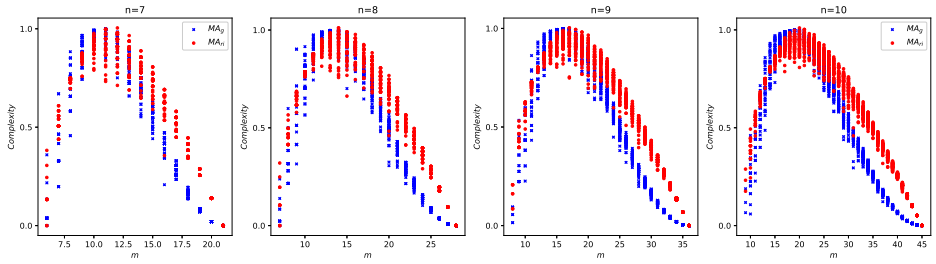

Figure 2: Comparison of MA_{RI} and MA_g.

As showin in figure 2, MA_{RI} and MA_g behave alsmost the same, whereas MA_{RI} assigns higher complexity values to graphs with middle number of edges and averagely higher complexity than MA_g for higher number of edges. The complexity of computation stays the same, which is $O(m)$. MA_{RI} complexity intends to have the nomalization property, which is $0 \leq MA_{RI} \leq 1$, with exceptions.
During the simulation, a few unnormalization cases were found, they are:
These exceptions are only found for small graphs, for larger graphs, complexities are

n	m	Highest complexity
6	9	1.0421694413111797
7	11	1.0203660524038531
7	12	1.0042984248515054
8	15	1.0068924792733018

normalised. Therefore, a solution is proposed.
Observing the equation 2, the extremes are fixed for n, R and I are the variables. Also, R and I are negatively correlated with correlation coefficent equal to -1.

$$\begin{aligned} R &= \frac{1}{m}\sum_{i,j>i} ln(d_i d_j) \\ I &= \frac{1}{m}\sum_{i,j>i} ln(\frac{2m}{d_i d_j}) \\ I &= \frac{1}{m}\sum_{i,j>i} ln(2m) - \frac{1}{m}\sum_{i,j>i} ln(d_i d_j) \\ I &= ln(2m) - R \end{aligned} \tag{3}$$

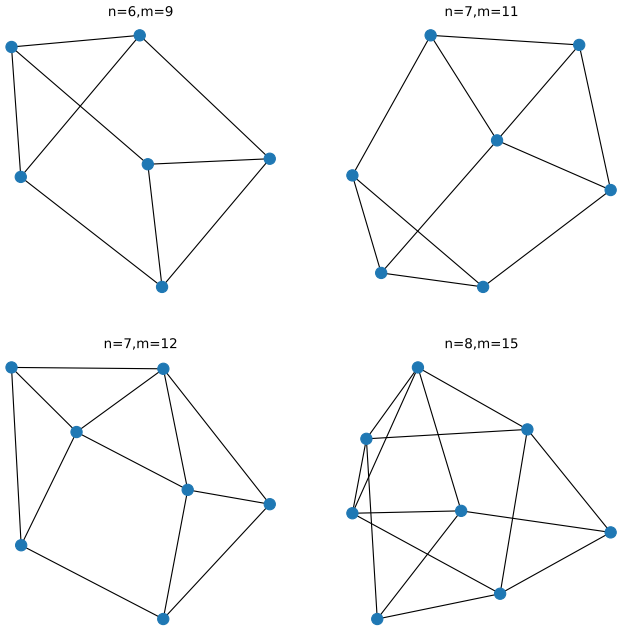

Figure 3: Graphs with highest MA_{RI} complexity.

Thus, if the maximum of $\sum_{i,j>i} ln(d_i d_j)$ can be found, the maxmimum of $MAri$ can be found as well. The maximum can be found in two ways:

- Creates all graphs with the respective n, and find the maximum complexity, and divide all valeus by the maximum complexity.
- As R and I are negatively correlated with coefficent -1, we can find the maximum of $MAri$. Given the fact $R_{min} = R_{path} = ln(2m) - I_{max} = ln(2m) - I_{path}$ and $R_{max} = R_{clique} = ln(2m) - I_{min} = ln(2m) - I_{clique}$. By varying the value of R between R_{min} and R_{max}, MA_{RImax} can be found.

We only suggest to use the solutions for small graphs, not only the problem only occurs for small graphs, but also the complexity will be largely increased for large graphs.

2.3 Potential problems and solutions of different subgraph measures

During the implementation of measures, several problems were found, possible solutions are also given for future projects.
Different subgraph measures are principally simple, but they are complex to compute, within at least $O(n^2)$ time[3]. This is not the only problem. An upper-bound of the complexity $m_{cu} = n^{1.68} - 10$ was introduced by Kim and Wilhelm[3] to normalise the complexity. However, from the simulation, we found that this may not be the actual upper-bound of the different subgraph measures.

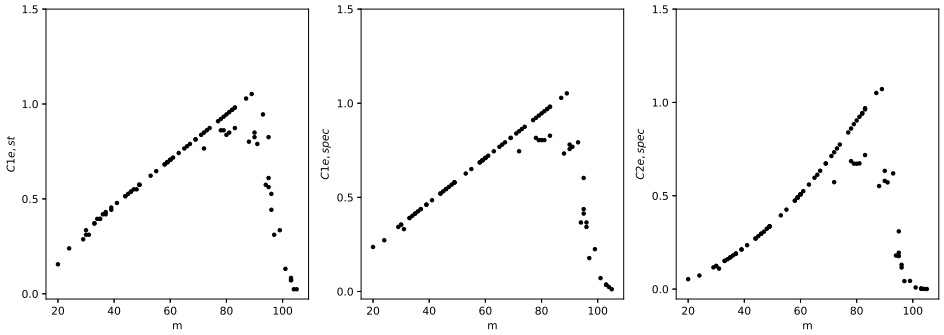

Figure 4: Different subgraph measure of $G(n, m)$ random graphs, with $n = 15$.

The complexity is abnormal for graphs with around 90 edges and 15 nodes as shown in figure 4. This could imply that the upper-bound *mcu* is not correct, but there is another possible reason, which is the problem of floating point arithmetic.
On most machines today, numbers are represented in binary system[1]. For example, 0.2 is 0.00110011001100110011... in a binary system. This series is infinite, represented by $1*2^{-3}+1*2^{-4}+1*2^{-7}+1*2-8$.... For obvious reasons, computer scientists don't want to work with infinite series, therefore, the series is approximated. On a modern compueter, the series is usually approximated to 63 digits with 1 digit represents the sign of the number. After approximation, the error could cause the equal operation to fail. A well known example is that for modern programming language or machine that operates this numbering system, 0.2+0.1 does not equal to 0.15+0.15. As a result, the comparison may cause more number of different subgraph than actual. The core of different subgraph measure is to compare the cofactor($C_{1e,st}$)/spectrum($C_{1e,spec}$ and $C_{2e,spec}$) of a subgraph. Given the fact that the proabbility of a decimal number to appear in the sprectrums is high and the cofactor will also be very large for a large graph. The comparison will be inaccurate. There are three possible solutions:

- As suggested, errors will be made when approxiamted by the machine. An error threshold can be used when comparing spectrums and cofactors. For example, two numbers with relative error less than 1% can also be considered as equal

numbers. One disadvantage is the increase of complexity, taking more time and effort to compare the spectrums/number of spanning trees.

- Similarly, numbers can be rounded before comparison to avoid error. This is used in the implementation of different subgraph measures, all cofactors and spectrums are rounded to first 10 significant figures. This solution requires less computation time than first solution. The drawback is that similar graphs can be considered as isomorphic graphs, this also applies to the first provided solution, but with higher accuracy for large graphs.

- Instead of using m_{cu} as a normalisation parameter, m can be used for one-edge-deleted subgraph complexity and $\binom{m}{2}$ for two-edges-deletied subgraph complexity. This gurantees the normalisation and avoid the mistake that caused by the first two solutions, but on the other hand, causing the complexity to be different.

A unique problem with $C_{2e,spec}$ is the value is not properly normalised for small graphs.

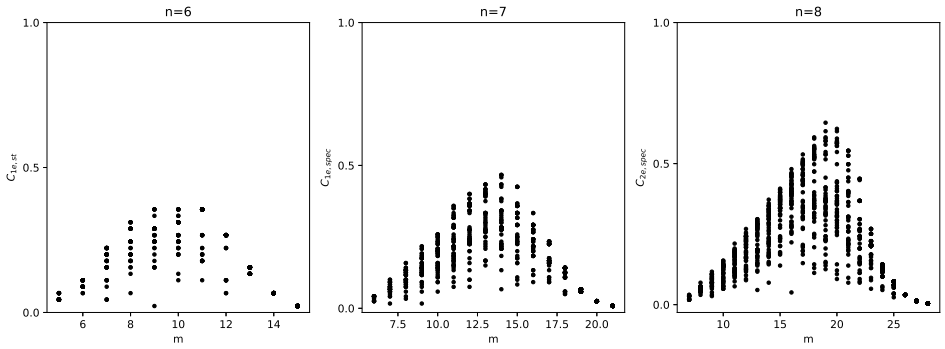

Figure 5: $C_{2e,spec}$ complexities for n = 6,7,8 respectively.

The upper-bound of $C_{2e,spec}$ is 0.5 while $n \leq 7$. To have an upper-bound at 1, the complexity value have to be scaled by 2. However, scale by 2 will cause the complexity to exceed 1 for larger graphs. Thus, we sticked to the original normalisation and $C_{2e,spec}$ will have an upperbound at 0.5 for $n \leq 7$.

3 Result

4 Conclusion

References

[1] *15. floating point ARITHMETIC: Issues and Limitations*. URL: https://docs.python.org/3.8/tutorial/floatingpoint.html.

[2] A.L. Barabási and M.Ã. PÃ3sfai. *Network Science*. Cambridge University Press, 2016. ISBN: 9781107076266. URL: https://books.google.co.uk/books?id=iLtGDQAAQBAJ.

[3] Jongkwang Kim and Thomas Wilhelm. "What is a complex graph?" In: *Physica A: Statistical Mechanics and its Applications* 387.11 (2008), pp. 2637–2652.

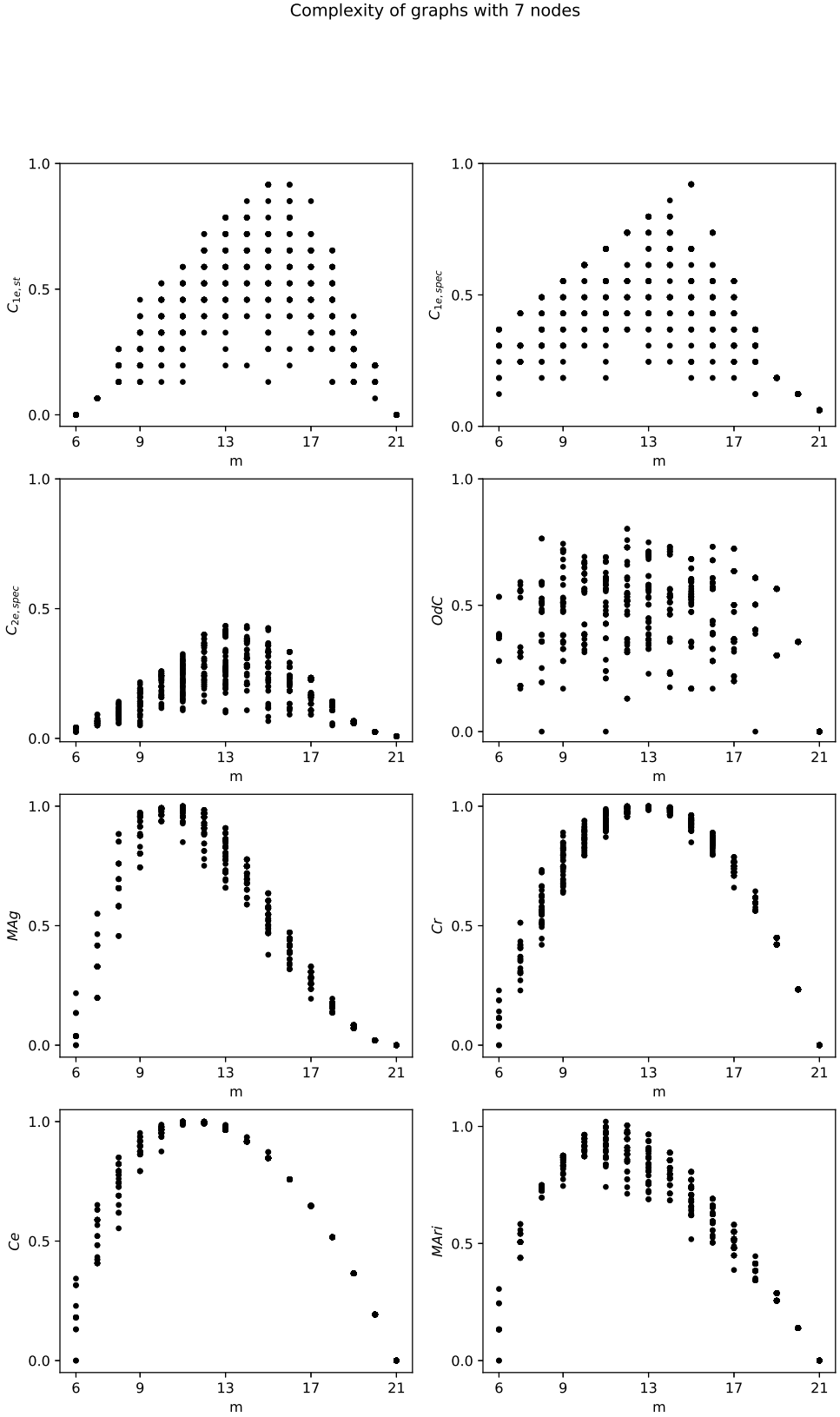

Figure 6: results of implemented methods of graphs with n =7. Methods from top-left