IE456 Graph Theory

Final Project Report

20.05.2023

Tevfik Buğra Türker 2019402120

Ömercan Mısırlıoğlu 2020402261

Table Of Contents

Introduction

In this report, random graphs and random graph generation processes are discussed. Concept of random graph generation, its history and the contributors of this topics throughout the history are introduced. The two algorithms, Havel-Hakimi and Sequential Algorithm (proposed by Blitzstein and Diaconis [1]) are shown and discussed. The algorithms are tested and compared according to their running times. Furthermore, a connectivity and m/n ratio analysis are conducted.

Blitzstein and Diaconis introduced an algorithm for generating random graphs from a graphical degree sequence, which is tested using the Erdos-Gallai Graphicality Test, in their paper [1]. They also proposed that the algorithm they introduced never gets stuck, and they provided a proof for this proposition. The proof is not examined in this study; however, the experimental results do not contradict with the proof.

Use of Generating Random Graphs

Networks are crucial for the contemporary world in social, technological, or biological aspect. These networks consist of parts that we call vertices and edges, which represent the real-world entities and connections between them respectively. It is essential to inspect and analyze the features of these networks, however, it is generally impossible to gather complete information about a network in real-life applications since there are millions or billions of vertices and edges in real-life networks. There are some solutions to overcome this difficulty, one is generating random graphs that have similar properties as the real-life network. Random graph generation is a process which generates graphs according to a probability distribution. The generated graphs can be used for testing new algorithms, understanding the behavior of some real-life network and study their properties under different conditions.

For instance, to understand to behavior of a food web in Chesapeake Bay, the web is modelled as a graph where vertices represent the 33 types of organisms living in the area. The corresponding degree sequence is used for generating random graphs [1]. This way, a large number of random graphs are created and they are tested against the actual food web.

Random graph generation is a very important tool for people that study networks and this topic is broadly studied in the field of network science [2][4]. The primary objective of this study is to assess the effectiveness of various graph generation algorithms, including Havel-Hakimi Algorithm and Sequential Algorithm. The algorithms are evaluated in terms of their running time and the implementations of these algorithms are shown in detail in the paper. Additionally, the study provides a summary for Erdos-Renyi Graphs and scale-free graphs. At the end the experimental results are presented for different algorithms highlighting the main differences and outcomes. By this study, it is aimed to give researchers and practitioners a better way of understanding these algorithms and provide a clear overview for selecting appropriate, efficient, and effective graph generation technique.

Scale Free Graphs Overview

A scale-free graph is some form of a complex network that displays a power-law degree distribution, where the probability p(k) that a node has degree k is given by [3] $P(k) = k^{-\gamma}$ for large k where γ is some exponent which determines the shape of the distribution and has a value generally in the range $2 < \gamma < 3$. In a scale free graph, some immensely connected nodes (hubs) exist side-by-side with other rarely connected nodes, which results with a heterogeneous distribution of degrees.

The Barabási-Albert (BA) model is a popular method for generating scale-free graphs [5]. In this model, a graph is grown iteratively by adding nodes one at a time and connecting each

new node to n existing nodes with a probability proportional to their degree. Each new node is connected to $d \leq d_0$ existing nodes with a probability that is proportional to the number of links that the existing nodes already have. Specifically, the probability p_i that the new node is connected to node i is $p_i = \frac{k_i}{\sum_j k_j}$ where k_i is the degree of node i and the sum is made over all pre-existing nodes j (i.e., the denominator results in twice the current number of edges in the network). This preferential attachment mechanism leads to the emergence of hubs and a power-law degree distribution with exponent $\gamma = 3$ [5].

The Barabási-Albert model is used for modelling a wide range of networks in real-world such as World Wide Web, social networks, and biological networks [6]. It is a strong tool to understand the general structure and functioning of complex systems while maintaining the simplicity.

An advantage of the Barabási-Albert model is that it relies on a simple and intuitive system for generating networks. The preferential attachment rule is supported by the idea that nodes which are highly connected on the network has a greater probability to attract new connections, which results with the emergence of hubs [5]. This idea has been validated by experimental studies on the real-world networks which shows that natural and artificial networks have a power-law degree distribution.

However, the Barabási-Albert model has limitations, for instance, it assumes that the network indefinitely grows, which is not a realistic assumption for many real-world networks that have a finite size or have some external constraints upon them [5]. Additionally, the model does not consider other significant features of real-world networks such as community structure, clustering and degree correlations [6].

In spite of the limitations, the Barabási-Albert model is an important tool for studying the dynamics and the structure of complex networks, and it impassioned lots of researches in the field of network science.

Erdos – Renyi Graphs Overview

The Erdos-Renyi model is a known and a classic method for generating random graphs that has been used widely in graph theory and network science [7]. In this model a graph is created with connecting n nodes to edges that have been selected uniformly from all possible pairs of nodes. It is also called G(a,p) (a = total number of vertices, p = probability of connecting an edge). As expected, distribution of the degree of any vertex is binomial and P(deg(v) = k) → $\left(\frac{a-1}{k}\right) p^k (1-p)^{a-1-k}$. All graphs with "a" nodes and "b" edges have equal probability of $p^b (1-p)^{\frac{1}{2}a(a-1)-b}$. The Erdos-Renyi model has several interesting properties, including a Poisson degree distribution and a phase transition from a regime of small connected components to a regime of a single large connected component as the average degree of the graph is increased. To show, for large a due to the property of a.p (a times p) (constant),

P(deg(v) = k) → $\frac{(ap)^k e^{-ap}}{k!}$ is called Poisson random graph. Poisson random graph is a special type of Erdos-Renyi random graph model [5].

Erdos – Renyi Graphicality Criterion

Erdos-Gallai Theorem: A sequence of non-negative $d_1 \geq d_2 \geq \ldots \geq d_n$ can be a degree sequence of a finite simple graph on n vertices if and only if $\sum_{i=1}^{n} d_i$ is even and for every $1 \leq k \leq n$, we have

$$\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(k, d_i)$$

The degree sequence $(d_1, d_2, \ldots, d_n)$ is a graphical sequence if there is a simple graph with vertex set {1, 2, …, n} where vertex i has degree d_i. Erdos-Gallai conditions are necessary for a sequence of numbers to be graphic and it can be shown [1]. The sufficient condition of the theorem has to be shown as the proof. In Choudum's proof, a mathematical induction on the sum of degrees is shown (degree: *s*). Theorem holds when *s* = 0 or *s* = 2 [8]. Assume that for sequences whose sum of degrees is *s-2* and let π: $d_1 \geq d_2 \geq \cdots \geq d_p$ be a sequence with even *s*. Let is satisfy Erdos-Gallai conditions. Then the proof is completed by induction if the sequence is graphical. Consider the sequence π^*.

$$\pi^*: d_1 \geq \cdots \geq d_{t-1} \geq d_t - 1 \geq d_{t+1} \geq \cdots \geq d_{p-1} \geq d_p - 1$$

In the paper by Choudum (1986), it is shown that the degree sequence π^* satisfies the Erdos–Gallai theorem by dividing the proof into five cases and proving each case separately. The proof relies on the inequality min(a,b) – 1 ≤ min(a-1,b), which is used to show that π^* is graphic based on the induction hypothesis [8]. This approach allows for a straightforward and concise proof of the Erdos–Gallai theorem. The details of the proof can be found in the original paper by Choudum (1986) and further references [8]. Let G be a realization of π^* on the vertices $v_1, v_2, \ldots, v_p$. If $(v_t, v_p) \notin E(G)$, then, G + (v_t, v_p) is a realization of π. Then, let $(v_t, v_p) \in E(G)$. Since $deg_G(v_t) = d_t - 1 \leq p - 2$, there is a v_m such that

$(v_m, v_p) \notin E(G)$. Since $deg_G(v_m) \geq deg_G(v_p)$, there is a v_n such that $(v_m, v_n) \in E(G)$ and $(v_n, v_p) \notin E(G)$. Deleting the edges (v_t, v_p) and (v_m, v_n) and adding the edges (v_t, v_m) and (v_n, v_p), we get a new realization of $G^* \; of \; \pi^* \; where \; v_t \; and \; v_p$ are non-adjacent. Then, adding (v_t, v_p) to the G^* is a realization of π [8].

Advantages of Sequential Algorithm

The sequential algorithm has a less probability to get stuck in a loop or fail to generate a graph than the Havel-Hakimi algorithm and pairing algorithms. This is because the sequential algorithm generates the graph creating one vertex at a time and it can adjust the degree sequence as the algorithm goes along, whereas the Havel-Hakimi algorithm and pairing algorithms generate the entire graph at once and may fail if the degree sequence is not suitable for generating a graph.

On one hand, in Sequential algorithm, since there will be no restarts needed the worst-case running time is fixed and has an upper bound which is $O(n^2)$, which becomes $O(n^3d)$ for d-regular graphs. On the other hand, the other algorithms which need restarts has an unbounded worst case running time, thus making their performances harder to evaluate [1].

Moreover, the Sequential Algorithm lends itself well to uniform graph generation. By sequentially adding edges and adjusting the degree sequence, the algorithm ensures that every realization of the given degree sequence has a positive probability of being generated. This uniformity in graph generation is particularly advantageous when analyzing and comparing properties or conducting simulations on random graphs.

In summary, the Sequential Algorithm stands out as a reliable and efficient choice for uniform graph generation. Its ability to avoid getting stuck, fixed worst-case running time, and capability to generate uniformly distributed graphs make it a valuable tool for various applications, ranging from modeling social networks to simulating complex systems.

Implemented Algorithms

Graphical Random Generation Algorithm (Uniform)

The algorithm works as follows:

1) Generate n random integers between 1 and n-1

2) Sort these n integers in non-increasing manner

3) Check the graphicality with Erdös-Gallai

4) If the sequence is graphical Stop and provide the sequence, otherwise go to Step 1.

Remark: 0 is omitted in Step 1, since 0 creates disconnected vertices and pairwise exchange cannot make it connected.

Graphical Random Generation Algorithm (Exponential)

To generate exponential random graphical degree sequences, replace step 1 in the previous algorithm with:

1) Generate n random exponential numbers with lambda = 2/n. Then round up to obtain integer values. If any of these integers exceed n-1, reject them and generate different random exponential numbers until all integers are smaller than n.

Havel – Hakimi Algorithm

Havel – Hakimi algorithm is used for constructing simple graphs with given degree sequences, where degree sequence is a list of numbers where the n^{th} number on the list is the number of edges incident to vertex n. This algorithm takes a degree sequence and as an output

it gives a simple graph (a graph without multiple edges between vertices and no loops) that consist the given degree sequence.

Highest Degree Vertex First, distributed to the Highest Degree

The algorithm works as follows [1]:

1) The degree sequence is sorted according to non-increasing order

2) If the degree sequence is empty, existing output is given and algorithm is terminated.

3) Remove the first vertex from the degree sequence and connect it to the k highest degree vertices in the degree sequence (k is the degree of the removed vertex).

4) Decrease the degrees by 1 (of the k vertices that the removed vertex was connected).

5) If any vertices with degree 0, remove them

6) Repeat step 1.

Random Vertex Selection, distributed to the Highest Degree

For this algorithm, just replace **step 3** with the following:

3) Remove a random vertex from the degree sequence and connect it to the k highest degree vertices in the degree sequence (k is the degree of the removed vertex). (Step 3)

Smallest Degree Vertex First, distributed to the Highest Degree

For this algorithm, just replace **step 1 and step 3** with the following:

1) The degree sequence is sorted according to non-decreasing order

3) Remove the first (smallest) vertex from the degree sequence and connect it to the k highest degree vertices in the degree sequence (k is the degree of the removed vertex).

Smallest Degree Vertex First, distributed to the Smallest Degree

For this algorithm, just replace **step 1 and step 3** with the following:

1) The degree sequence is sorted according to non-decreasing order

3) Remove the first (smallest) vertex from the degree sequence and connect it to the k lowest degree vertices in the degree sequence (k is the degree of the removed vertex).

Sequential Algorithm

Sequential Algorithm is another graph constructing algorithm. Different from Havel-Hakimi, in Sequential Algorithm while constructing a graph, the degree sequence is considered as "*residual degrees*" which is the remaining number of edges which are waiting to be chosen. The algorithm aims frequently subtracting 1 at certain coordinates of degree sequences until all elements of the degree sequence is 0.

The algorithm works as follows [1]:

1) Let E be an empty list of edges.

2) While degree sequence is not all 0, do:

3) Choose degree i with the smallest positive value.

4) Compute candidates ($j \in J$) for edge formation (Candidates should be: 1) Different than the chosen vertex, 2) Edge (i,j) should not be already in E. 3) After deducing 1 from the positions i and j in the residual degree sequence, the remaining residual degree sequence should be graphical.)

5) Choose j from the probable candidates list J, with probability proportional to its degree in residual degree sequence.

6) Add the edge (i, j) to E and deduce 1 from the positions i and j in the residual degree sequence.

7) Repeat steps from 4 to 6 until the degree of i is 0. Then go to Step 2.

Pairwise Edge Interchange

To make disconnected graphs connected, pairwise edge interchange algorithm can be used. The pairwise edge interchange operation is performed on a pair of non-adjacent edges (u, v) and (x, y) in a graph, where u, v, x, and y are distinct vertices. The operation involves removing these two edges from the graph and adding two new edges, (u, x) and (v, y), in their place.

The algorithm works as follows:

1) Check the connectivity of the graph G. If G is connected Stop. Otherwise go to Step 2.

2) Choose edges (u, v), (x, y) randomly.

3) If (u, x) and (v, y) forms a simple graph, make the changes and go to Step 1, otherwise go to Step 2.

PART 2: EXPERIMENTAL RESULT ANALYSIS

Graph Generation Ability of Algorithms

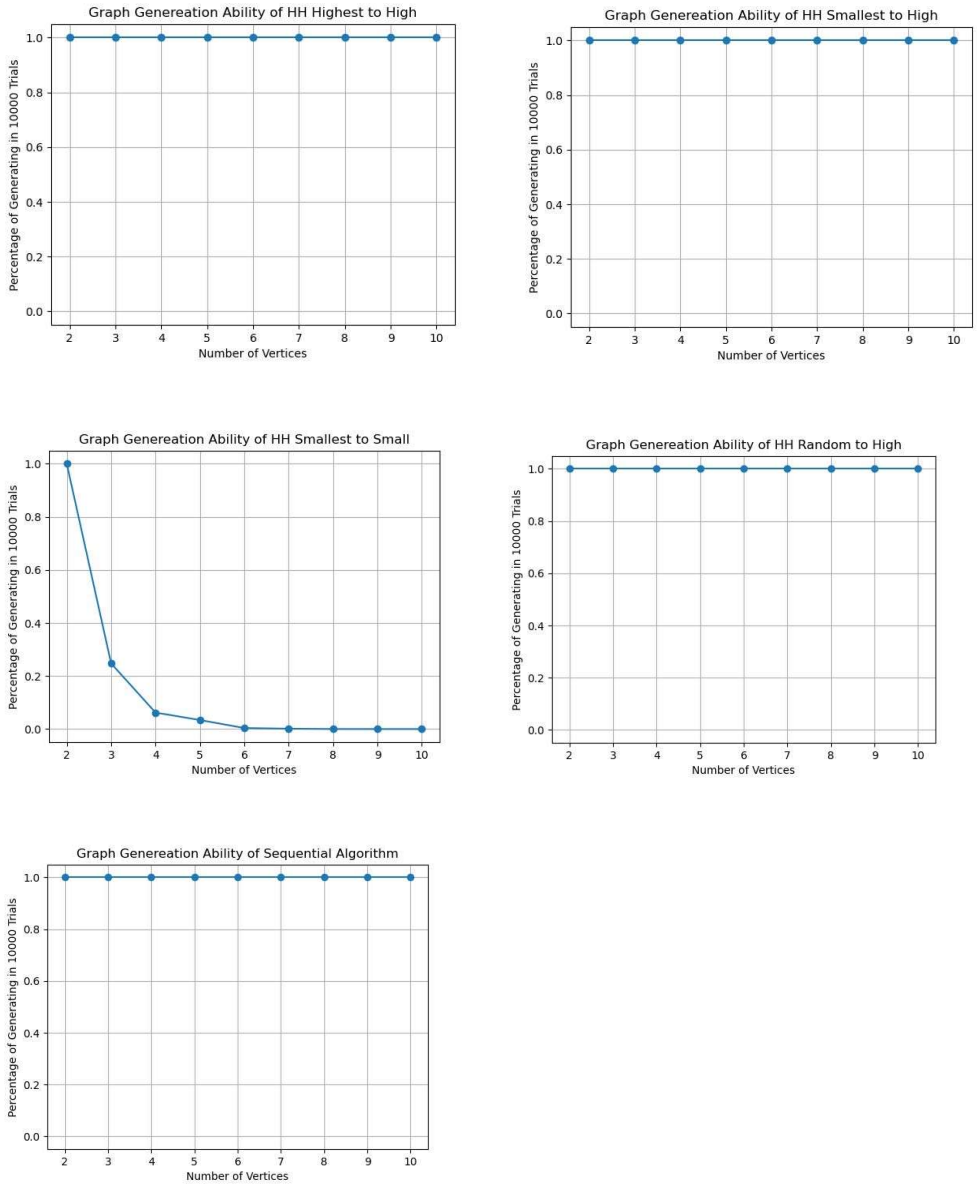

Figure 1. Graph Generation Percentage of Algorithms wrt Degree Length

In **Figure 1.** the percentages for generating a graph from a given **graphical degree sequence** and is shown. In x-axis the length of the graphical degree sequence, in y-axis the percentage of generated graphs (average of **10000** graph generation trials) is shown. According to these results, all algorithms except smallest-to-small algorithm can generate graphs without an issue,

however, smallest-to-small algorithm generates significantly small number of graphs after a degree sequence length of 6. For this reason, in all of the future analysis for these algorithms, smallest-to-small algorithm will not be used.

Computational Time Analysis

For different algorithms and different n values, a computation time analysis has been carried out for n = {10,25,50,100,1000,10000} respectively. Smallest-to-small algorithm is not tested since it could not generate graphs. To obtain more **accurate results**, for every algorithm, a certain **number of trials** is used and their averages is taken. The values in the table are the **average results** of the indicated number of trials, and number of trials are shown at the rightmost column respectively for every algorithm. For the sake of computational time, number of trials have been decreased as **n** increases. Using every algorithm respectively, following results have been obtained.

	Random to High	Highest to High	Smallest to High	Sequential	Number of trials respectively
n = 10	3.9310e-05 sec	3.6976e-05 sec	2.9938e-05 sec	0.0016 sec	(10000, 10000, 10000, 10000)
n = 25	0.0001 sec	0.0002 sec	0.0002 sec	0.0970 sec	(10000, 10000, 10000, 1000)
n = 50	0.0004 sec	0.0005 sec	0.0006 sec	2.629 sec	(10000, 10000, 10000, 100)
n = 100	0.0014 sec	0.0020 sec	0.0022 sec	60.28 sec	(10000, 10000, 10000, 10)
n = 1000	0.1574 sec	0.1852 sec	0.2109 sec	-	(1000, 1000, 1000, 1000)
n = 10000	19.45 sec	22.49 sec	25.38 sec	-	(100, 100, 100, 100)

Table 1. Computational Times

According to the results of computational time analysis test in **Table 1.**, for the lowest n value (10), all of the Havel-Hakimi algorithms works very fast. However, as n value increases, one can observe a separation between computational times of these algorithms such that random-to-high algorithm stands fastest among three, highest-to-high algorithm takes more

time than random-to-high algorithm while staying faster than smallest-to-high algorithm. Smallest-to-high algorithm is the slowest of these three Havel-Hakimi algorithms. For sequential algorithm, the separation between computation times compared to Havel-Hakimi algorithms is obvious. The sequential algorithm is significantly slower than the other algorithms and this can be observed starting from very low n values. The reason for this situation is as follows, sequential algorithm computes all the candidates at every step, it takes a lot of time for creating the graph. With growing n values, the time it takes to complete all the computations increases exponentially. Thus, the sequential algorithms stand as the slowest among used algorithms in this test.

Connectivity Analysis

For different algorithms and different n values, a connectivity analysis has been carried out for n = 10, n = 25, n = 50 and n = 100. Sequential algorithm is not computed after n = 25 due to very high computation time. For all algorithms, higher n values could not be computed due to very big computing time. According to this analysis, following results have been obtained:

	Random to High	Highest to High	Smallest to High	Sequential	Number of trials respectively
n = 10	66	469	0	26	(10000, 10000, 10000, 10000)
n = 25	11	476	0	5	(10000, 10000, 10000, 10000)
n = 50	1	485	0	-	(10000, 10000, 10000)
n = 100	0	465	0	-	(10000, 10000, 10000)

Table 2. Connectivity Table

According to the experimental results in **Table 2.** as **n** increases, number of disconnected graphs generated;

- Decreases for **random-to-high** algorithm
- Is not affected for **highest-to-high** algorithm
- Is not affected for **smallest-to-high** algorithm since it never created disconnected graphs
- Decreases for **sequential algorithm**

Density Analysis

In this part, two of the random graph generating algorithms are compared to observe the effect of density over the connectivity of the graph. The density has an effect on the connectivity of the graph as can be seen in below charts, increasing density lowers the probability of graph being disconnected. The following results are obtained **before** making pairwise interchanges (edge swaps). For every algorithm, graphs are generated with two versions of graphical random degree sequence generation algorithm. First one used uniform random numbers between 1 and n-1, whereas the second one used exponential random numbers with lambda = 2/n (also between 1 and n-1). The main difference between two methods according to the following results is that the graphs generated with exponential random numbers are less dense than the graphs generated with uniform random numbers. The main cause of this is when generating exponential numbers, the degree values higher than n-1 are **rejected** and a smaller degrees are generated. Thus, the real expected value of a degree drops below n/2 in an exponential random degree sequence, resulting with less dense graphs.

Figure 2. and Figure 3. belongs to Havel-Hakimi **highest to highest** random graph generating graph algorithm. For different values of n(5,10,20,40), 100000 graphs are generated for each n value. After, these graphs are categorized according to their density and disconnectedness percentage for every density have been shown in below graphs.

As density increases, the probability of a graph being disconnected decreases rapidly to 0.

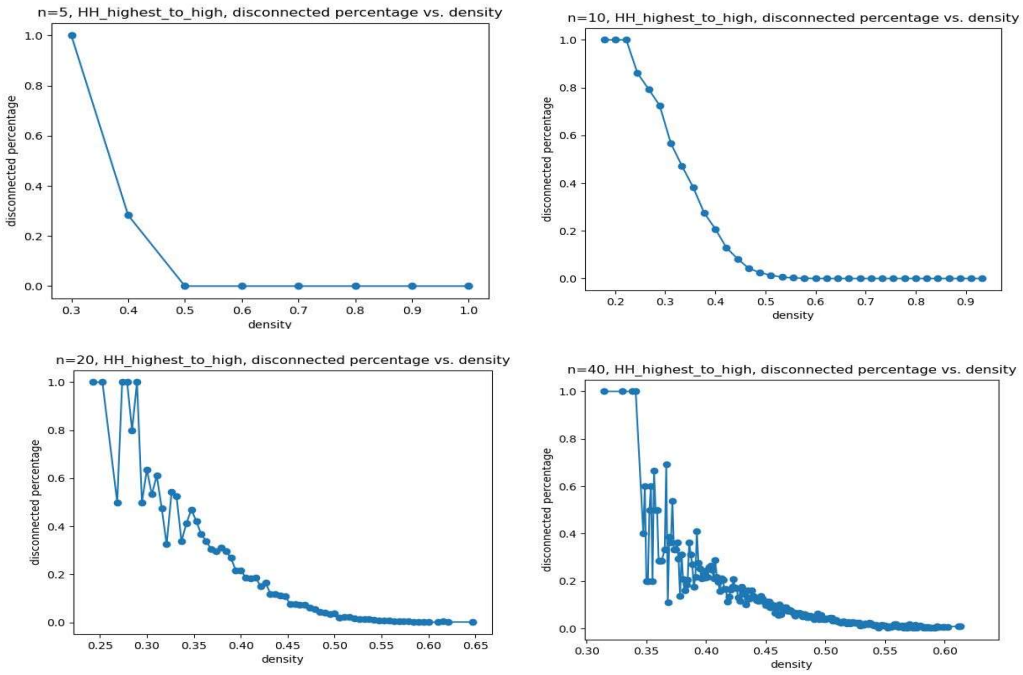

Figure 2. Highest to Highest w/uniform random number generation

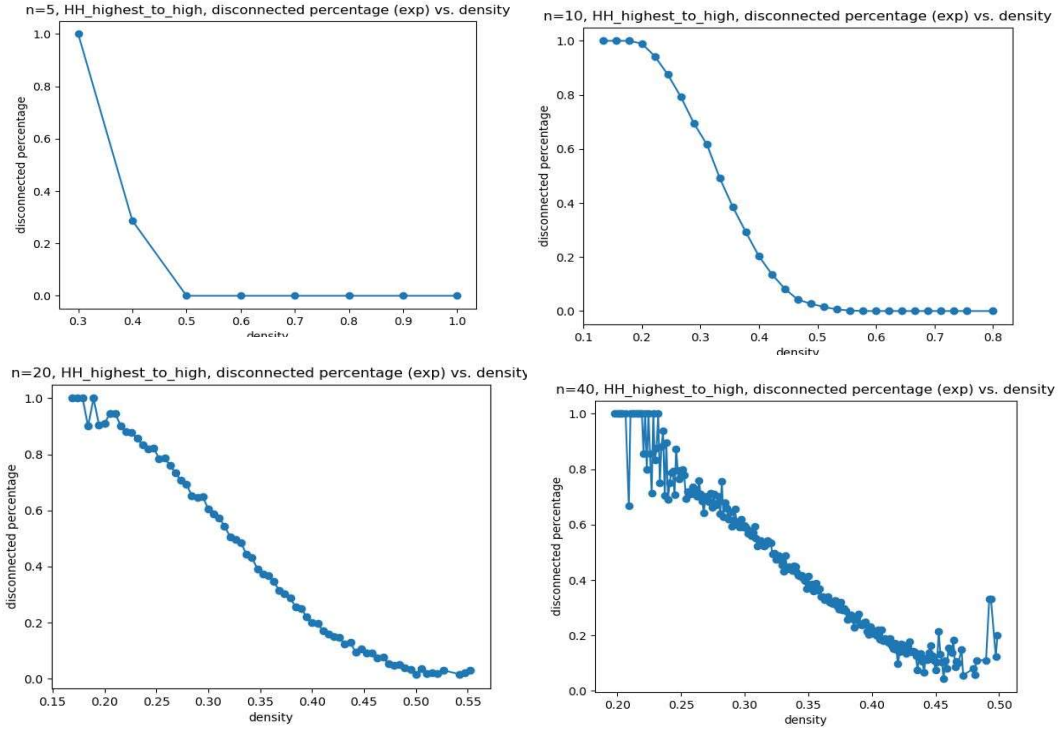

Figure 3. Highest to Highest w/exponential random number generation

Figure 4. and **Figure 5.** belong to Havel-Hakimi **smallest to highest** random graph generating graph algorithm. For different values of n(5,10,20,40), 100000 graphs are generated. After, these graphs are categorized according to their densities and disconnectedness percentage for every density have been shown in below graphs.

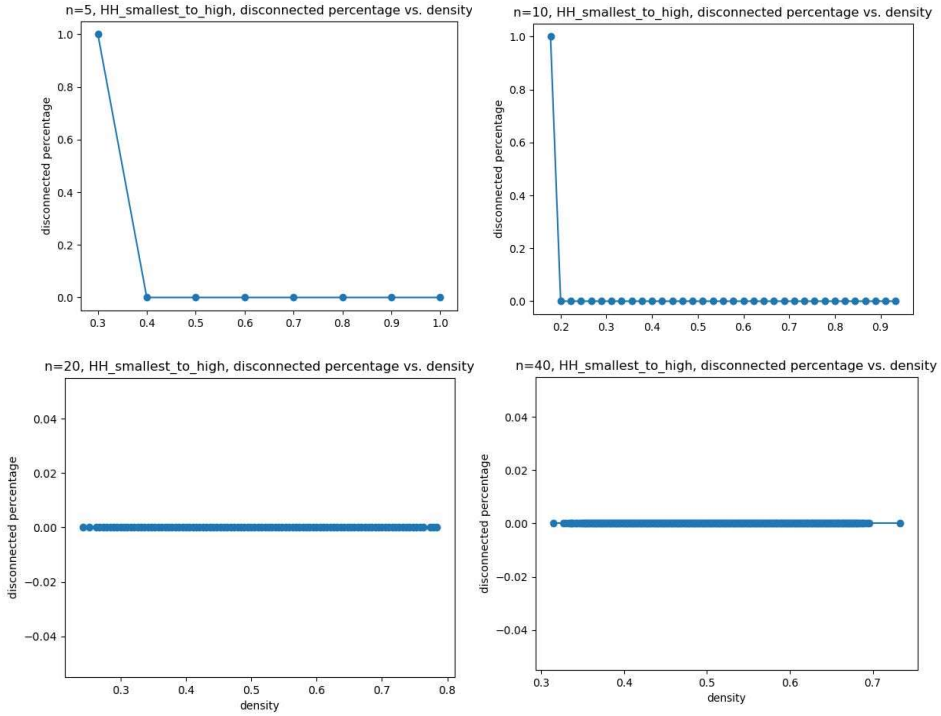

Figure 4. Smallest to Highest w/uniform random number generation

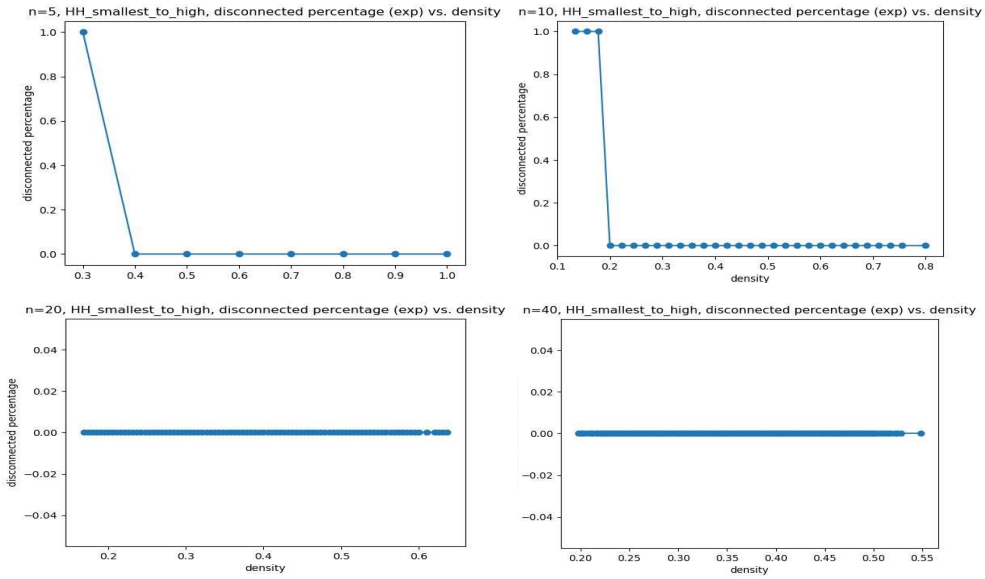

Figure 5. Smallest to Highest w/exponential random number generation

Figure 4. and **Figure 5.** shows that in **smallest to high** algorithm, the probability of having a disconnected gaph is ~0, as a result there are nearly no observations of a graph being disconnected. This is also coherent with the experimental result of small-to-highest algorithm.

In **Figure 6.** and **Figure 7.** It is shown that **Random-to-high** algorithm gave a number of disconnected graphs between small-to-high and highest-to-high algorithms, thus it has similar charts with highest-to-highest algorithm with **lower** disconnectivity ratios for density values.

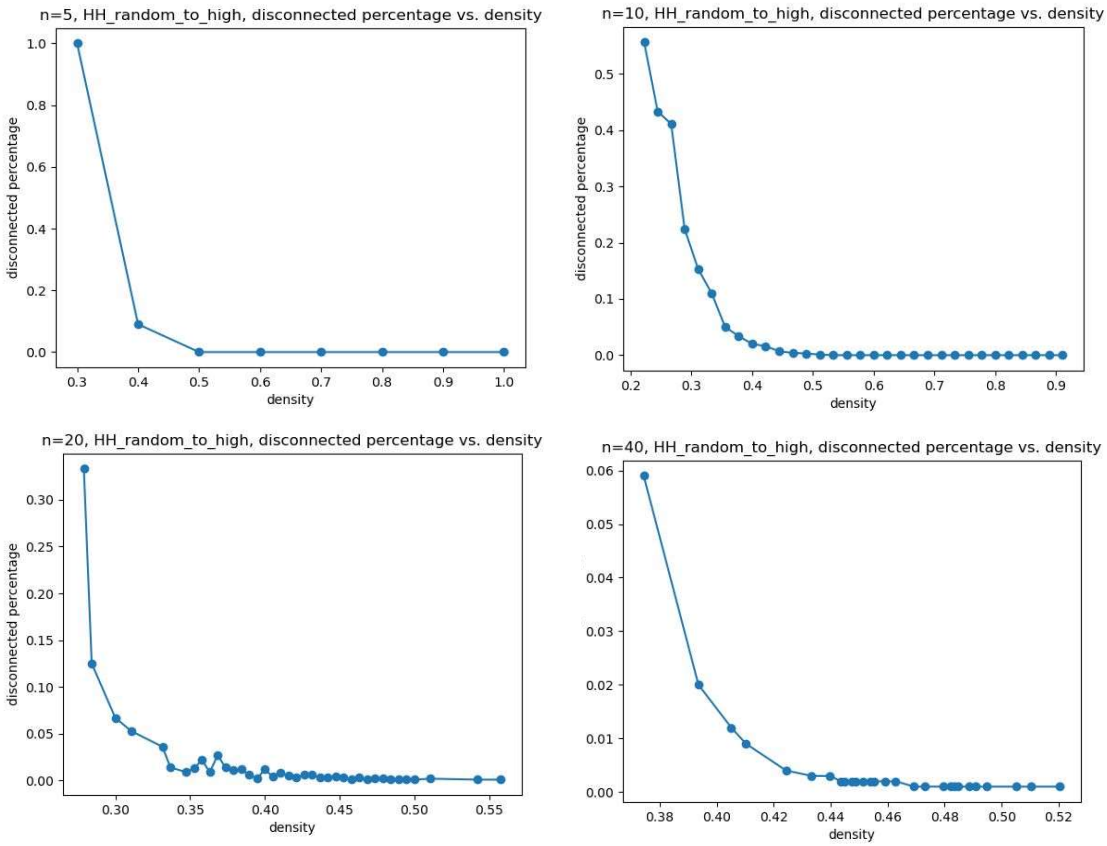

Figure 6. Smallest to Highest w/Random random number generation

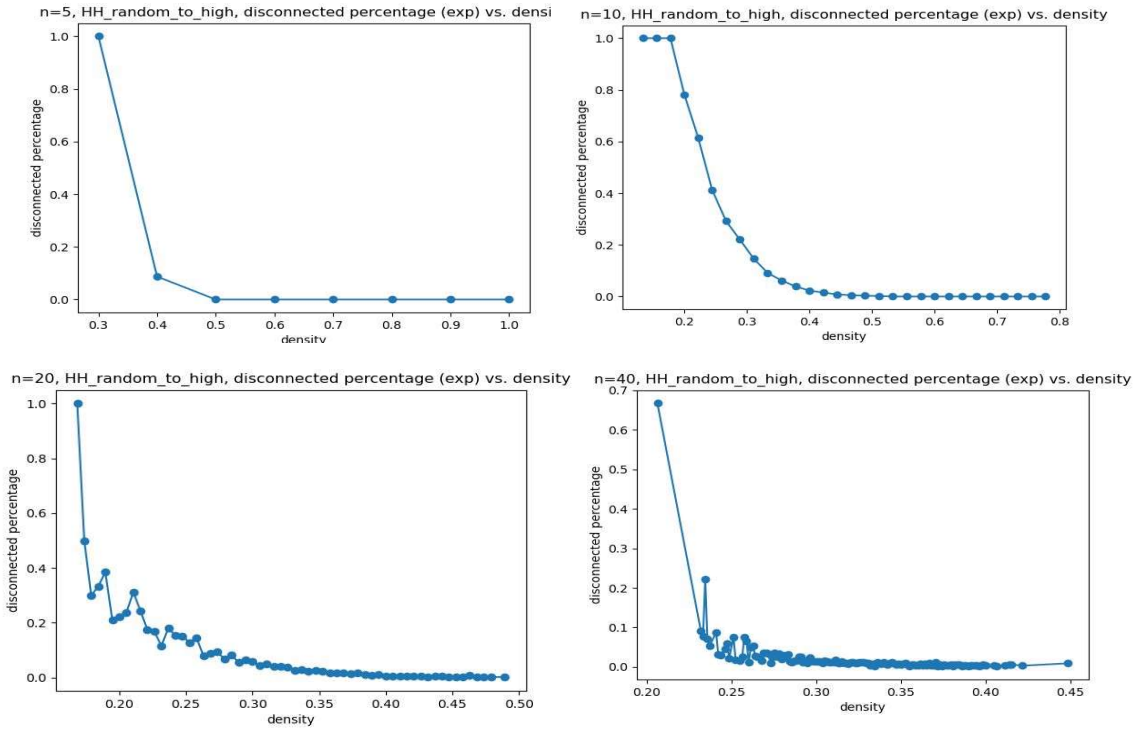

Figure 7. Random to Highest w/exponential random number generation

References

[1] J. Blitzstein and P. Diaconis, “A sequential importance sampling algorithm for generating random graphs with prescribed degrees,” pp. 1–30, 2006.

[2] Newman, M. E. J., Barabási, A. L., & Watts, D. J. (2006). The structure and dynamics of networks. Princeton University Press.

[3] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," Science, vol. 286, no. 5439, pp. 509-512, Oct. 1999.

[4] Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., & Hwang, D. U. (2006). Complex networks: structure and dynamics. Physics Reports, 424(4-5), 175-308.

[5] M. E. J. Newman, "Networks: An Introduction," Oxford University Press, 2010.

[6] B. Bollobás, "Random Graphs," Cambridge University Press, 2001.

[7] P. Erdős and A. Rényi, "On Random Graphs," Publicationes Mathematicae Debrecen, vol. 6, pp. 290-297, 1959.

[8] S.A. Choudum, "A simple proof of the Erdos-Gallai theorem on graph sequences," Bull. Austral. Math. Soc., vol. 33, no. 1, pp. 67-70, 1986.