THE COMPARISON OF ALGORITHMS FOR GENERATING RANDOM GRAPHS WITH PRESCRIBED DEGREES

BY MURAT TUTAR AND FATMANUR YAMAN

Boğaziçi University

Abstract. This report analyzes the paper titled "Sequential Algorithm for Generating Random Graphs" by Blitzstein and Diaconis[1]. The paper presents an algorithm for generating random graphs with a given degree sequence. The aim of this report is to examine the algorithm, understand its properties, and evaluate its advantages compared to similar studies in the field of random graph generation. Additionally, we will explore the implications and applications of this algorithm in various domains. The content of the report includes an overview of the sequential algorithm, a discussion of its advantages over other approaches, an examination of the properties of the generated graphs, and a consideration of potential extensions and future research directions. By delving into this paper, we aim to deepen our understanding of random graph generation and its significance in modeling complex networks.

Keywords: Random Graph Generation, Havel-Hakimi, Erdos-Renyi.

1 Introduction

1.1 Presentation, Motivation and Content

For many decades, the studies on generating random graphs has been an active field of research in combinatorics and graph theory. In recent decades, it has also become an important tool in understanding the relationships and dynamics of a wide range of other phenomena, including -but not limited to- social networks, epidemiology or economics. By generating random graphs, researchers managed to model very complex network models and to gain insights by analyzing them.

Various algorithms for generating random graphs have been studied and used so far, including the widely used Erdos-Renyi model. However, in real life applications, these algorithms have several limitations in terms of efficiency and performance.

The motivation of the paper is to provide a comprehensive overview of the problem of generating graphs with given degree sequences, to analyze the state-of-the-art algorithms used in the field, and last but not least, to introduce a sequential algorithm for random graph generation by providing an in-depth analysis describing its features and comparing those to the previous algorithms mentioned in terms of complexity, efficiency, and performance.

The paper presents this sequential algorithm and applies it to generate synthetic food webs, motivated by the desire to understand the structural properties and underlying principles of real food webs. By closely replicating the degree distribution observed in empirical food webs, the algorithm allows for the creation of synthetic webs that capture important features of real ecosystems. These synthetic webs serve as valuable tools for studying and quantifying the statistical properties of food webs, enabling researchers to distinguish meaningful patterns

from random fluctuations. The analysis of synthetic food webs provides insights into the organization and dynamics of ecological networks, contributing to the development of ecological theory and informing ecosystem management strategies. By comparing the properties of synthetic webs to real food webs, the study aims to uncover general principles governing the complex interactions between species in natural ecosystems and shed light on the factors driving the structure and stability of food webs.

In the paper, the authors first introduce the concept of degree sequences and provide an overview of the existing models for generating random graphs with a given degree sequence. After that, the authors propose a sequential algorithm with an in-depth explanation and step by step implementation. Then, they prove that the algorithm is correct and they analyze its run-time. Finally, the empirical performance of the algorithm and its applications are discussed.

1.2 The Use of Generating Random Graphs

Generating random graphs is a widely used approach with many applications in different fields. Some of its main applications are in the following fields:

Social Networks: The usage of random graphs is very common to interpret and model the relationships and social dynamics in a social network. In these graphs, by creating and analyzing the random graphs where nodes represent individuals and edges represent the relationships between them, researchers can make valuable observations regarding the information diffusion and influence patterns.

Epidemiology: The studies on infectious diseases and modeling their spread patterns are another important application of generating random graphs. Creating and analyzing random graphs where vertices correspond to individuals and edges correspond to contact between them helps scientists to understand the dynamics of the spread of infectious diseases.

Biology: In molecular biology, random graphs are used to model protein-protein interactions and gene regulatory networks, allowing for the identification of key components and better understanding of biological processes.

Computer Networks: Generating random graphs are also useful in modeling the structure of the internet, wireless communication networks, and other complex networked systems. This helps in calculating and comparing their resilience, vulnerability, and robustness under various conditions.

Neuroscience: Another application of random graphs is neuroscience, which studies the connectivity of neurons in the brain. By using random graph models where nodes are brain regions and edges are their connections, neuroscientists gain useful information on cognitive processes.

Economics: In the field of economics, random graphs are useful in modeling the economic networks like financial systems. This way, economists gain more insights regarding the dynamics of financial systems.

Ecology: In the field of ecology, a usage area of random graphs is the study of interactions between species in an environment. In these graphs, nodes represent the species and edges represent their interactions. By analyzing these graphs, the policymakers can provide better ecosystem management.

Along with widely used areas above, random graphs also have many more in very different fields, including physics (string theory), graph theory, mathematics (random matrix theory) and so on. They are very important in modeling complex systems and analyzing their structure and dynamics.

2 Related Work

2.1 Explanation of Erdos-Renyi Graphs

Erdos-Renyi graphs are used to generate a random graph model based on a given number of vertices and edges. There are two variations of the model, namely the G(n, M) model, where a graph is generated with n vertices and M edges selected randomly from all possible edges, and the G(n, p) model, where a graph is constructed with n vertices and each edge is included with some probability p.

In the G(n, p) model of Erdos-Renyi graphs, the parameter p represents the probability of including each edge in the randomly generated graph. Specifically, for each pair of vertices, an edge is included in the graph with an independent probability of p. Therefore, p determines the likelihood of any two vertices being connected by an edge in the generated graph. As p increases from 0 to 1, the model becomes more inclined to include edges, resulting in a denser graph with more connections between vertices. Conversely, as p approaches 0, the graph becomes sparser with fewer edges and less connectivity.

Compared to G(n,M) model, G(n,p) is more widely used and it generates graphs with an expected number of edges which follows a binomial distribution, indicating that the generated graph is more likely to have a relatively uniform degree distribution; although including some outlier vertices with significantly higher or lower degrees than the remaining vertices.

Erdos-Renyi graphs are widely used in graph theory and its application areas for modeling random graphs. They can be used to study properties of large networks and to compare real-world networks to randomly generated graphs. However, they may not accurately capture all the properties of real-world networks, and more complex models may be needed to fully capture the behavior of real-world networks.

2.2 Scale-Free Graphs: What they are, Where they arise, How they are generated

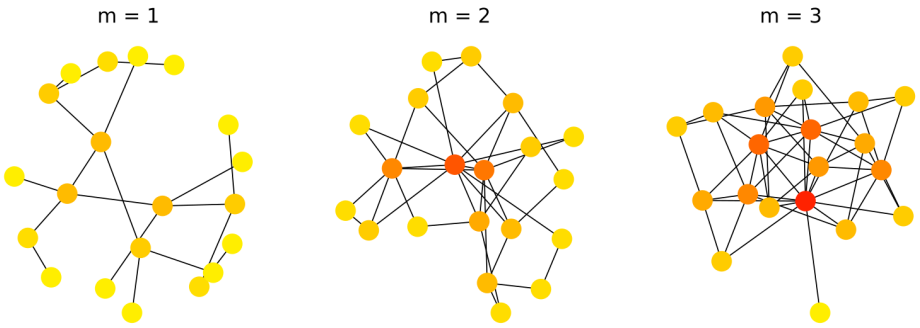
Scale-free graphs are graphs which demonstrate a power-law degree distribution. In other words, they have a few vertices with very high degrees whereas the majority of the vertices have a low degree. In these networks, the probability of a vertex having a degree k

has a power-law distribution, $P(k) = k^{-\gamma}$ where γ is a positive constant. To put it verbally, scale-free graphs are distinguished by the existence of highly connected vertices, known as hubs, that are playing a central role in the overall structure of the network.

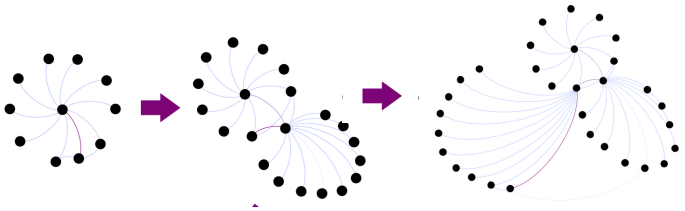
They arise in many different environments in complex real world systems, including -but not limited to- the internet, social networks, citation networks, and biological networks. They are characterized by being resilient to unexpected random failures since most of the vertices have a low degree, and the hubs with high number of connections can ensure the continuation of the network's connectivity.

Several methods can be used to generate scale free graphs. Some of these methods are:

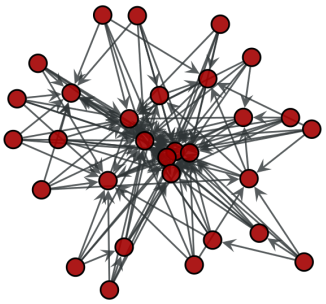
Barabasi-Albert Model[2]**:** This model is based on the concepts of growth and preferential attachment. The model begins with a small number of vertices and then it adds new vertices repeatedly, connecting them to existing vertices according to probabilities that are proportionate to the degree of the existing vertices. This way, the existing vertices with higher degree will be more likely to be connected to the new vertex; and with each connection the probability will keep increasing. This process can be thought as “rich get richer” following a power-law degree distribution. The figure below is an illustration of graphs generated based on the Barabasi-Albert model with the same number of vertices and different parameters for degrees (m=1, m=2, m=3, respectively). The color of a vertex gets darker proportional to its degree.



Bianconi-Barabasi Model[3]**:** This model is an extension of the previous model. It is not only based on preferential attachment and growth, but it also has another concept: Fitness. It assigns a fitness value for each vertex and existing vertices with higher fitness value are more likely to attach the new vertices. The Barabasi-Albert model had a structure such that the late vertices can’t become a large hub, whereas in some real life systems, that is not the case. For instance, although Google was launched later than Yahoo, it has much more connections than Yahoo. This can’t be explained with the Barabasi-Albert model whereas it can be explained by including the Fitness concept, as Google has a higher fitness than Yahoo. An exemplary illustration of a network generated by the Bianconi-Barabasi model is below.



Generalized Linear Preferential Attachment Model[4]**:** This is a different model using the concept of preferential attachment. It has a tunable parameter which can modify the preferential attachment probability function, making it possible to generate graphs with varying levels of heterogeneity in their degree distributions. By changing this parameter, networks with different power-law exponents can be constructed, leading to scale-free networks with different properties. The figure below is an illustration of preferential attachment where a few vertices have a high number of incoming edges, whereas most of the vertices have a low number of incoming edges.



The models above and many other models can generate scale-free networks with unique features and characteristics. Each of these unique models can be applied to different problems and they can capture different aspects of the real-world systems, making the scale-free networks a useful tool with applications in a wide variety of real-life scenarios.

2.3 Explanation of the Proofs for the Erdos-Gallai Graphicality Criterion

Choudum's proof by Induction on the length of degree sequence(n):

Base case is n=1 and if there is only a single vertex, its degree is zero and the sequence is graphical as it is an isolated vertex.

For the inductive step, it can be assumed that the theorem holds for sequences of length n-1. Let d_1, d_2, ..., d_n be a non-increasing degree sequence of length n. If the theorem's conditions hold for this sequence, a graph G can be generated with this degree sequence.

Then, we remove the vertex v with degree d_1 from G, and let d'_1, d'2, ..., d'(n-1) be the degrees of remaining vertices. By the inductive hypothesis, the sequence d'_1, d'2, ..., d'(n-1) is graphical. This means there is a graph G' with this degree sequence. Now, we reintroduce vertex v and connect it to the d_1 vertices with the highest degrees in G'. This

results in a graph with the degree sequence d_1, d_2, ..., d_n. Thus, the theorem holds for sequences of length n.

Unfortunately, we could not find Berge's original paper. We could only find its original version in French, which was scanned and could not properly be translated to English by Google translate or other translating tools.

2.4 Advantage of the Sequential Algorithm to Other Studies

The sequential algorithm proposed by the authors is superior to others in many aspects. It is a simple and efficient algorithm which can be implemented easily along with a good computational performance compared to the existing algorithms.

For each pair of vertices, randomly flip a coin to determine whether they share an edge or not in the classical random graph model G(n,p) of Erdos-Renyi. This approach creates a graph with binomial degree distributions since the degree of a vertex has Binomial(n-1,p) distribution. However, several large networks have power-law degree distributions rather than binomial distribution. The sequential algorithm can also be used to generate such graphs. This is one of the advantages of the sequential algorithm over Erdos-Reyni graphs.

In addition to this, in the sequential algorithm, only the available set of vertices which still construct a graph even though they remove from the current graph is considered. Also, the probabilities are proportional to the degrees in the sequential algorithm. This is also one of the factors that makes the sequential algorithm preferable because of the computational efficiency.

Additionally, the paper introduces the concept of importance sampling as a technique to obtain a uniform distribution of generated graphs within the sequential algorithm. Importance sampling is a variance reduction method commonly used in Monte Carlo simulations to estimate rare events or compute expectations under a target distribution. In the context of random graph generation, importance sampling allows for the generation of graphs with a specific degree sequence that are uniformly distributed.

In the sequential algorithm, after generating a graph using the candidate selection process, the algorithm assigns a weight to each generated graph based on its probability of occurrence under the desired uniform distribution. These weights are calculated using combinatorial factors and probabilities associated with each step of the sequential algorithm. By incorporating these weights, the algorithm biases the selection of graphs towards those with lower probabilities, effectively compensating for the non-uniformity of the original sequential algorithm.

The use of importance sampling enhances the algorithm's efficiency and accuracy in generating graphs that are representative of the desired uniform distribution. It allows researchers to obtain unbiased estimates of various graph properties and perform statistical analyses on the generated graphs with greater precision. By employing importance sampling within the sequential algorithm, the paper provides a valuable tool for generating random

graphs with prescribed degree sequences that follow a uniform distribution, enabling more reliable and robust analysis of graph properties and behavior.

3 Methodology

3.1 Explanation of the Algorithms

Our code generates a random graphical degree sequence and then creates a graph with this degree sequence. This is done by the usage of five different algorithms (Four Havel-Hakimi and One Sequential proposed by the authors). After constructing the graph, its connectivity is checked and if the graph is not connected pairwise edge interchanges are performed to make it connected. The code consists of different functions and algorithms:

random_degree_sequence(n): Generates a random graphical degree sequence of length n. It checks if the generated sequence is graphical by using *is_graphical(seq)* function and repeats the process until a valid graphical sequence is obtained.

is_graphical(n): Checks whether the given degree sequence seq is graphical or not.

is_connected(adj_matrix): Checks whether a graph represented by an adjacency matrix is connected or not by using breadth first search.

swap_edges(adj_matrix, niter=100): Carries out pairwise edge interchanges in the adjacency matrix in order to make the graph connected while keeping its degree sequence unchanged. It chooses the first edge randomly and then checks for a second edge that does not share a vertex with the first edge. After choosing the edge pair, it checks whether the newly generated edges already exist in the graph or not (to make sure the graph remains simple). If those edges do not exist, then the pairwise swap operation is done.

By using the functions above, five algorithms are implemented to generate graphs:

random_vertex_highest_degree(n): This is the implementation of Havel-Hakimi Algorithm with random vertex selection, edges distributed to the highest degree. It randomly chooses a vertex with non-zero degree from the degree sequence and connects its edges to the vertices with the highest degrees. It repeats this until all the degrees in the degree sequence become zero, showing that there are no more edges to be connected. The generated graph is represented using an adjacency matrix.

highest_vertex_highest_degree(n): This is the implementation of Havel-Hakimi Algorithm with highest degree vertex selection, with edges distributed to the highest degree. It chooses the vertex with highest degree and distributes its edges to the vertices with highest degrees. It repeats this until all the degrees in the degree sequence become zero. The obtained graph is represented using an adjacency matrix.

smallest_vertex_highest_degree(n): This is the implementation of Havel-Hakimi Algorithm with smallest degree vertex selection, with edges distributed to the highest degree. It chooses the vertex with the smallest degree and distributes its edges to the vertices with highest

degrees. It repeats this until all the degrees in the degree sequence become zero. The resulting graph is represented using an adjacency matrix.

smallest_vertex_smallest_degree(n): This is the implementation of Havel-Hakimi Algorithm with smallest degree vertex selection, with edges distributed to the smallest degree. It chooses the vertex with the smallest degree and distributes its edges to vertices with the smallest degrees. It repeats this until all the degrees in the degree sequence become zero. The generated graph is represented by an adjacency matrix.

the_sequential_algorithm(n): This is the implementation of Sequential Algorithm. It takes the number of vertices and gives the edges connections like Havel-Hakimi. In every iteration it takes the vertex with the smallest degree and connects it with one of the remaining vertices by controlling that a graph can be constructed with the remaining vertices and edges. It decreases the degrees of the vertices when they connect. The connection vertex is found by selecting according to the probabilities proportional to their degrees among the available vertices. This operation will take till all degrees reach 0. At the end, this algorithm gives which vertices have a connection between them.

The main program runs the five algorithms using a random degree sequence. For each algorithm, it checks if the graph is connected and performs edge interchanges to ensure connectivity. It then prints the adjacency matrix of the final connected graph.

4 Experimental Results

To compare the algorithms and reach conclusions, experiments are performed. To apply the algorithms, first a graphical degree sequence had to be obtained. This is done by several steps. To begin with, a random sequence of n positive integers is initialized. Each integer is obtained from a uniform distribution, meaning all values in the range have an equal chance of being picked. Then, the graphicality of the sequence is checked for each sequence, by using a function based on Havel Hakimi Theorem for graphicality checking. If the sequence is not graphical, it is replaced with a new random sequence until it finds a sequence that is graphical. In essence, this function uses trial and error to generate random degree sequences until it finds one that can be realized as a simple graph.

4.1 Run Time Comparison

According to the results of the experiment with different values for order and seed, the fastest algorithm among the Havel-Hakimi algorithms is to start with the highest degree vertex and distribute its edges to the other highest degree vertex. Starting from a random vertex and distributing the edges to vertex with highest degree and starting from the smallest distributed to highest degree vertex have similar running times. The slowest among Havel Hakimi methods is to start from the vertex with the smallest degree and distribute its edges to the smallest degree vertices, and in addition to that it usually can not construct a graph. It is because the algorithm could not find appropriate graphs as the number of vertices increases,

since the vertices with high degrees could not be connected to other edges with this algorithm and need to do self-loops which makes the graph non-simple. Also, among all the algorithms the Sequential Algorithm is the slowest one. This is because it checks the graphicality of the remaining graph for every pair of vertices while setting the candidate list.

The experimental results with different values for order (n=5, n=25, n=50, n=100, n=500) and seed (average of 100 different results) supporting our conclusions are displayed in the Appendix as Table 1, Figure 1, and Figure 2. The results are first tabularized and then visualized by means of charts. In the visualization, the sequential algorithm is not shown as its runtime is very high and it causes the results of Havel-Hakimi algorithms to disappear. Also, the method in which the smallest vertex is distributed to the highest vertex is not included because most of the time it is unable to generate a graph and its results are inconclusive.

The comparisons are visualized based on two aspects. Firstly, the runtimes for different algorithms are compared with different n values. Secondly, the algorithms are compared for each of the n values separately. The results are supporting the claims mentioned before (see Table 1, Figure 1, Figure 2 in the appendix).

4.2 Connectivity Comparison

In addition to the runtime comparisons, we tried to compare the algorithms in terms of connectivity. As mentioned before, the algorithm starting with the smallest vertex with its edges distributed to the vertex with the smallest degree, almost never achieves to generate a connected graph. Among the other three Havel-Hakimi Algorithms, the percentage of connectivity is very close to each other for each n (number of vertices) value. Also, it is observed that for a very small vertex number (n=5), the connectivity percentage is significantly lower than higher n values. It is seen that as n increases the connectivity percentage also approaches to 1 (for n=500), however there are also variances in the results as can be seen from the values n=25 and n=50. Based on our previous observation, we would expect that n=50 had a higher percentage, however n=25 has a better connectivity rate. So, many factors contribute to the result of this percentage and it is not deterministic (we can't say that the connectivity percentage will always definitely increase if we increase n, but it is of course very likely). Also, it can be observed that Havel Hakimi Algorithms are very successful in generating connected graphs, especially as the n value increases, as the connectivity percentages are more than 0.95 after a relatively high value for n. Overall, we can say that the most important step for generating a connected graph by using one of the Havel-Hakimi Algorithms is to distribute the edges of the chosen vertex to the vertex with highest degree among the remaining vertices; it is not very crucial to start from a specific vertex, the crucial thing is the methodology based on which the edges are distributed.

4.3 Number Of Edge Swaps Comparison

In addition to the runtime and connectivity analysis, the number of edge swaps is another important metric that shows the success of the algorithms. If a generated graph does

not form a connected graph, the algorithm makes it connected by swapping the edges. The swap operation adds extra runtime and complexity to the existing algorithms. Thus, it should be also analyzed.

For different values of the vertices and the different types of algorithms, the number of needed interchanges are given in Table 3. As it can be seen that, at first the needed number of swaps increases when the number of vertices increases. It can be interpreted as the algorithms have hard times for finding the connected graphs in the first trial as the number of vertices increases. In addition to this, it is normal to expect having a large number of swaps as the number of vertices increase, because of the large number of swap candidate vertices.

The algorithm based on the comparison was made in Figure 3. All types of algorithms show similar trends. The needed swaps for n=500 is zero for all types of algorithms. When the number of vertices is higher than some specific threshold, the algorithms can find the connected graphs in the first trial without needing to do the swaps.

5 Conclusion

In conclusion, the article provides a comprehensive explanation and analysis of various algorithms for generating random graphs with a given degree sequence. The code implementation includes five algorithms: four variations of the Havel-Hakimi Algorithm and one Sequential Algorithm proposed by the authors. Both the Havel-Hakimi Algorithm and the Sequential Algorithm have their strengths and weaknesses. The Havel-Hakimi Algorithm with the highest degree vertex selection and edge distribution to vertices with the highest degrees provides a fast and reliable approach for generating connected graphs. On the other hand, the Sequential Algorithm offers a different perspective by considering the degree distribution probabilities, although it comes with a longer runtime. The choice between these algorithms depends on specific requirements and considerations in the context of generating random graphs with a prescribed degree sequence.

APPENDIX

Table 1 - Run Time Table

Number of Vertices	highest_vertex_highest_d..	random_vertex_highest_de..	smallest_vertex_highest_..	the_sequential_algorithn
n=5	0.000943622	0.000172854	0.000136757	0.002548218
n=25	0.000360491	0.000937274	0.000867379	1.858118
n=50	0.001689568	0.00307663	0.001487277	18.50253
n=100	0.003528366	0.004518704	0.004108219	Null
n=500	0.04268979	0.05490005	0.04758506	Null
n=1000	0.154603	0.2041315	0.1749804	Null

Figure 1 - Run Time Algorithm Based Comparison

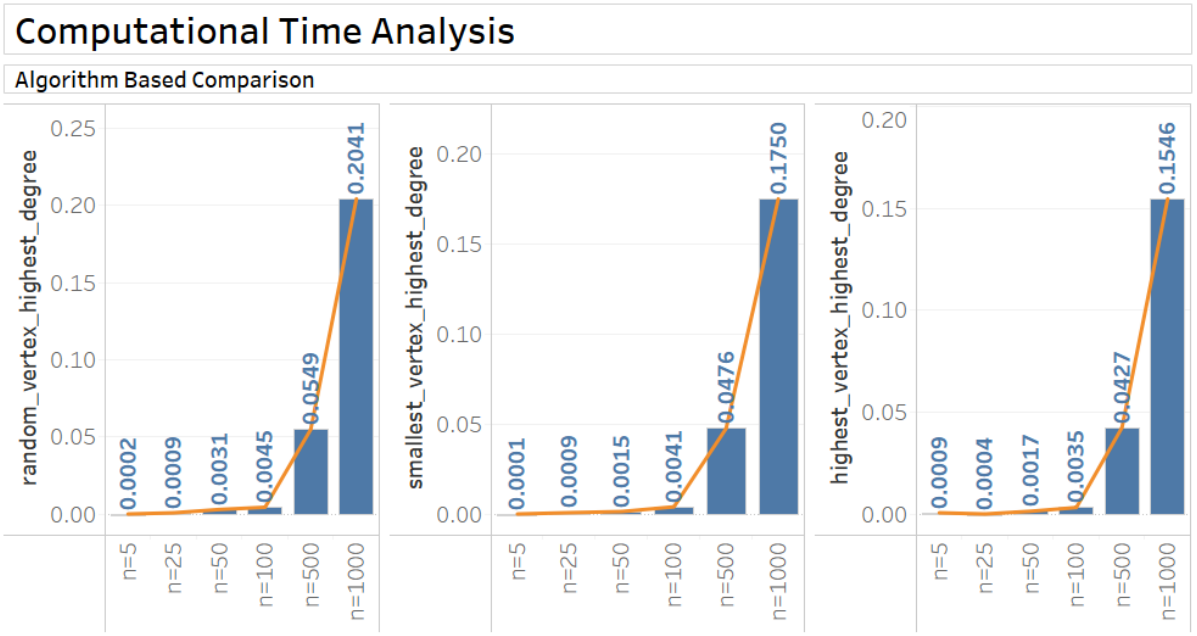


Figure 2 - Run Time Number Of Vertices Based Comparison

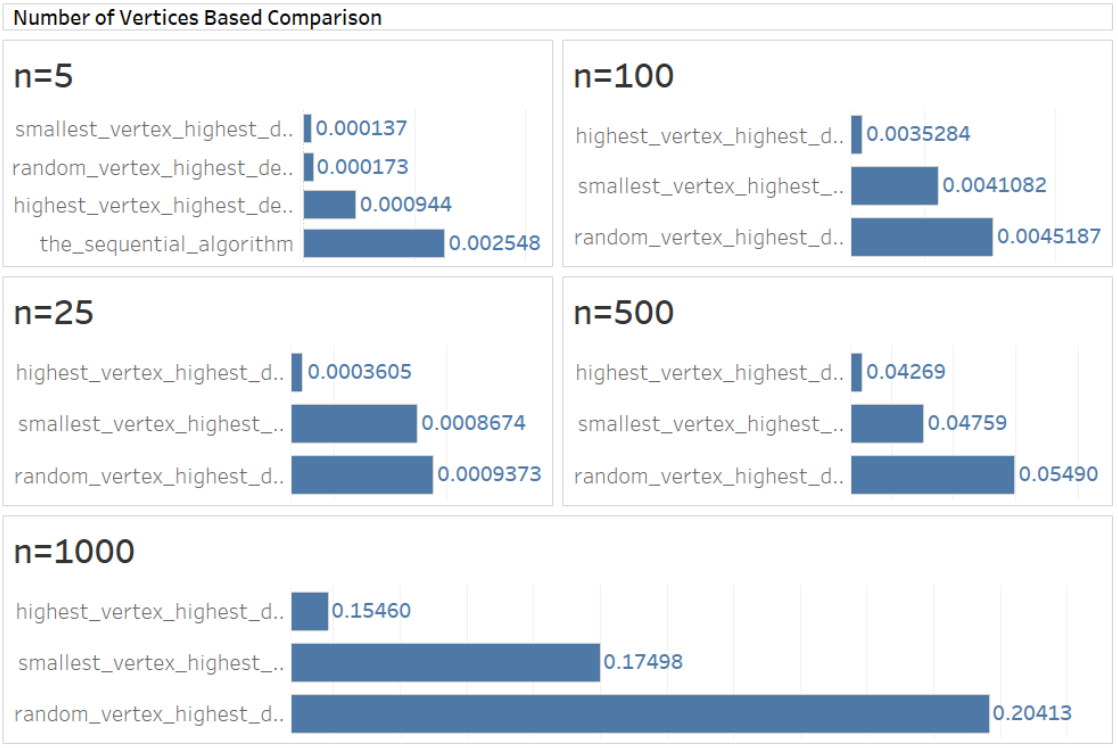


Table 2 - Connectivity Table

Number of Vertices	highest_vertex_highest_d..	random_vertex_highest_de..	smallest_vertex_highest_degr
n=5	0.84	0.85	0.85
n=25	0.96	0.97	0.97
n=50	0.94	0.94	0.94
n=100	0.97	0.97	0.97
n=500	1	1	1

Figure 3 - Connectivity Algorithm Based Comparison

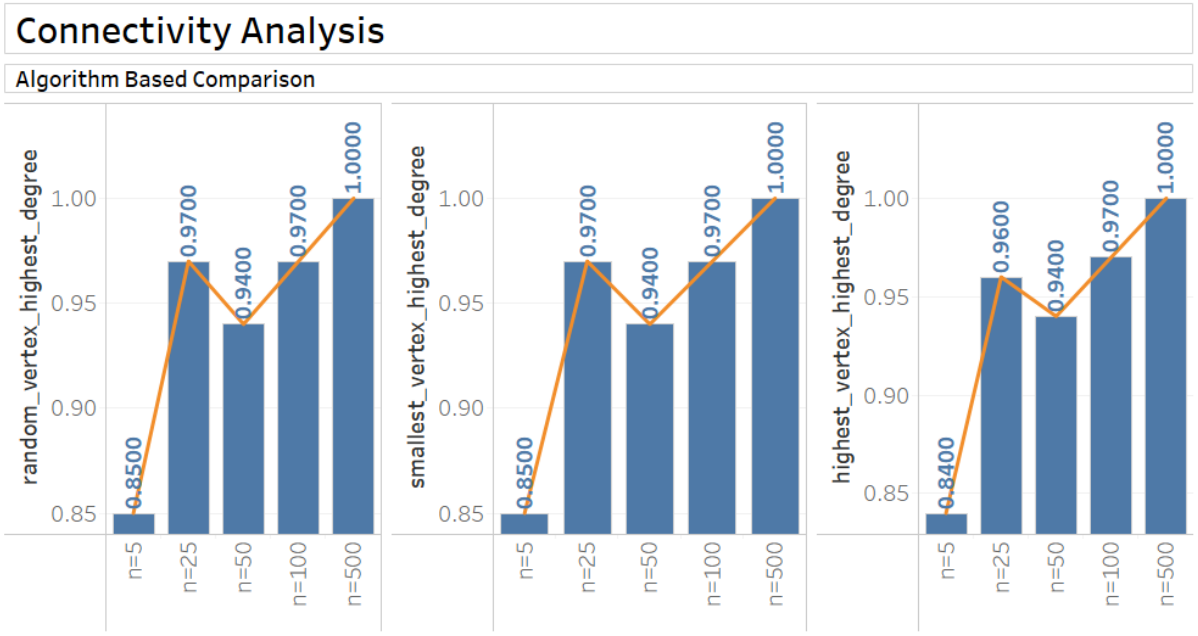


Table 3 - Number Of Swaps Table

Number of Vertices	highest_vertex_highest_d..	random_vertex_highest_de..	smallest_vertex_highest_degr
n=5	68	56	61
n=25	39	38	36
n=50	171	171	170
n=100	147	150	146
n=500	0	0	0

REFERENCES

1- Blitzstein, J., & Diaconis, P. (2010). A Sequential Importance Sampling Algorithm for Generating Random Graphs with Prescribed Degrees. Internet Mathematics, 6(4), 489-522.

2- Barabási, A. L., & Albert, R. (1999). Emergence of scaling in random networks. Science, 286(5439), 509-512.

3- Bianconi, G., & Barabási, A. L. (2001). Competition and multiscaling in evolving networks. Europhysics Letters, 54(4), 436-442.

4- Dorogovtsev, S. N., Mendes, J. F., & Samukhin, A. N. (2000). Structure of growing networks with preferential linking. Physical Review Letters, 85(21), 4633-4636.