Generating Random Graphs with Tunable Clustering Coefficient

Nidhi Parikh

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
Computer Science and Applications

Lenwood S. Heath, Chair
Madhav Marathe
Anil K. Vullikanti

March 18, 2011
Blacksburg, VA

Keywords: Clustering coefficient, complex networks, random graphs, algorithms

Generating Random Graphs with Tunable Clustering Coefficient

Nidhi Parikh

ABSTRACT

Most real-world networks exhibit a high clustering coefficient—the probability that two neighbors of a node are also neighbors of each other. We propose four algorithms Conf-1, Conf-2, Throw-1, and Throw-2 which are based on the configuration model and that take triangle degree sequence (representing the number of triangles/corners at a node) and single-edge degree sequence (representing the number of single-edges/stubs at a node) as input and generate a random graph with a tunable clustering coefficient. We analyze them theoretically and empirically for the case of a regular graph. Conf-1 and Conf-2 generate a random graph with the degree sequence and the clustering coefficient anticipated from the input triangle and single-edge degree sequences. At each time step, Conf-1 chooses each node for creating triangles or single edges with the same probability, while Conf-2 chooses a node for creating triangles or single edge with a probability proportional to their number of unconnected corners or unconnected stubs, respectively. Experimental results match quite well with the anticipated clustering coefficient except for highly dense graphs, in which case the experimental clustering coefficient is higher than the anticipated value. Throw-2 chooses three distinct nodes for creating triangles and two distinct nodes for creating single edges, while they need not be distinct for Throw-1. For Throw-1 and Throw-2, the degree sequence and the clustering coefficient of the generated graph varies from the input. However, the expected degree distribution, and the clustering coefficient of the generated graph can also be predicted using analytical results. Experiments show that, for Throw-1 and Throw-2, the results match quite well with the analytical results. Typically, only information about degree sequence or degree distribution is available. We also propose an algorithm Deg that takes degree sequence and clustering coefficient as input and generates a graph with the same properties. Experiments show results for Deg that are quite similar to those for Conf-1 and Conf-2.

ACKNOWLEDGMENTS

I express my sincere gratitude to my advisor, Dr. Lenwood S. Heath, for being a constant source of inspiration, support, and guidance. I thank him for patiently listening to all my ideas, meticulously going over cumbersome results, giving valuable direction to my work, and always being there to talk. I am also thankful to my committee members Dr. Anil Vullikati and Dr. Madhav Marathe for useful discussion. I also thank the Department of Computer Science, Virginia Tech.

Thanks to Astha and Nimisha, my roommates, and all my friends for being there and making graduate school fun.

Finally, I thank my parents Meena and Kiran Parikh for their love and understanding.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Motivation

Many systems in the real world take the form of networks, such as the Internet [18], the World Wide Web [4, 19, 20], social networks of connections among individuals [32], collaboration networks [2, 28, 39], neural networks [40], food webs [7, 13], and the citation network of scholarly papers [19].

One of the major goals in various applications of these networks is to measure and understand the characteristic properties and behavior of these networks under various processes taking place on the networks. For example:

1. Robustness: Against random or targeted removal of nodes (i.e., attack on Internet routers).
2. Diffusion: Gradual spread of signals over time (i.e., disease or information spread).
3. Search: How individual nodes look for and access information (i.e., to understand the effect of network topology on search strategies).

However, a typical real-world network is so large that it is nearly impossible to collect a complete data set for it, i.e., a data set for the network of the World Wide Web is obtained by crawlers to which URLs for a few seed web pages are given. As hyper-links appear in these seed web pages, those are added into the data set. Then, other hyper-links from these new web pages are added to the data set and the process continues. Naturally, it is almost impossible to reach all web pages available on the Internet. Also, these real-world networks change over time, and often it is required

to predict the structure or behaviour of these networks in the future, i.e., it may be important for a cell phone company to understand movement of people to plan the infrastructure needs for the future or the government may be interested in knowing travel patterns to address future needs of the transportation system.

The usual practice to address this problem is to investigate the properties of real-world networks by empirical measurements and then to develop models that imitate real-world networks and generate graphs with desired properties. These generated graphs are then analyzed with respect to their structural properties. Such modeling can help in exploring "what if" scenarios by simulation when data on real networks is impossible or difficult to collect or for predicting behavior of a large system in the future.

It has been observed that most of these real-world networks share a few common properties:

1. Skewed degree distribution: Node degrees are distributed over wide range and nodes are there with degree at every scale but with a few nodes having very high degree and high percentage of the nodes having low degrees. While many real-world networks believed to follow power law degree distribution, Clauset et al. [8] show that some of them actually follow it, for some of them, it seems to be good fit, some of them do not certainly follow it, and for some of them the possibility of power law degree distribution is not entirely ruled out.

2. Low mean geodesic distance: For a network with n nodes, the average of the shortest distance between all pairs of nodes is bounded by $O(\log n)$.

3. High clustering coefficient: If a node u is connected to node v and node v is connected to node w then, there is a high probability that nodes u and w are also connected.

In the classic Erdös-Renyi model, each pair of nodes is connected with the same probability p. It produces a graph with a degree distribution that is Poisson. The configuration model describes a method to generate a graph with a prescribed degree sequence. In the preferential attatchment model, a new node u is added at each time stamp and an edge of this new node u is connected to the previous node v with probability proportional to its degree $d[v]$. It produces graphs with power law degree distribution but low clustering coefficient. The small-world model produces a graph with high clustering coefficient and low mean geodesic distance, but it does not have a power law degree distribution.

1.2 Problem

Traditional algorithms for graph generation do not allow one to tune the clustering coefficient. For the classic Erdös-Renyi or $G_{n,p}$ model [11], the clustering coefficient is the same as the probability p of connecting any pair of nodes. For the preferential attachment model [3, 31], as the number of nodes increases, the clustering coefficient decreases to 0. The small-world model [39] can produce networks with high clustering coefficient, but, as the number of random connections increases, the clustering coefficient decreases.

Recently, Newman [17] proposes an algorithm to generate a random graph with tunable clustering coefficient. It is based on the configuration model [25]. But instead of taking a degree sequence as input, it takes a triangle degree sequence and a single edge degree sequence as input. Here, triangle degree of a node v, is defined as the number of triangles node v participates in and single-edge degree of node v is defined as the number of edges incident on node v, which do not participate in any triangles. However this algorithm does not specify a few important details, such as the probability with which each node should be selected to create triangles and single edges. We have developed four versions of this algorithm, Conf-1, Conf-2, Throw-1, and Throw-2. As Conf-1 and Conf-2 stick to the input degree sequence, the action taken at each step depends upon the residual triangle and single edge degrees and hence the previously added triangles and edges. Consequently it is difficult to analyze. However, we can approximately anticipate the clustering coefficient from the input triangle and single edge degrees. Conf-1 and Conf-2 differ in the probability with which nodes are selected to create triangles and single edges. Throw-1 and Throw-2 do not explicitly create the input degree sequence. For Throw-1, three nodes selected to create a triangle or two nodes selected to create a single edge need not be distinct while, for Throw-2, they have to be distinct. We analyze Throw-1 and Throw-2 theoretically. Experiments show that for graphs of moderate density, the clustering coefficient of a graph generated by Conf-1 and Conf-2 matches the anticipated clustering coefficient, and the clustering coefficient of graph generated by Throw-1 and Throw-2 matches the analytical clustering coefficient. Also, in [17], the algorithm is evaluated for clustering coefficient only for the case of a graph with average degree less than or equal to 2. We evaluate our two algorithms for much higher average degree. Typically, the information about triangle and single edge degree is not available, only information about the degree sequence is available. We propose an algorithm Deg that takes degree sequence and clustering coefficient as input and generates a random graph with the given properties. We show that the experimental

results are similar to those for CONF-1 and CONF-2.

1.3 Reader's Guide

Most of the work in this area is usually inspired by the properties of real-world networks and tries to model them. So we began by describing different types of real-world networks and some of the common properties of these networks in Chapter 2 and Chapter 3. Various graph generation models are described in Chapter 4. Chapter 5 covers the related work done in this area. In Chapter 6, we generate power-law graphs and observe the naturally occurring range of clustering coefficients. Experiments on observing the naturally occurring range of clustering coefficients for power-law networks are also covered in the same Chapter. Chapters 7 and 8 contain the main results of the thesis. In Chapter 7, we discuss Newman's algorithm and present four versions of it. We propose a new algorithm that generates a random graph with given degree sequence and clustering coefficient in Chapter 8, and we conclude in Chapter 10.

Chapter 2

Graph: Definitions and Properties

2.1 Preliminaries

A *graph* $G = (V, E)$ is a mathematical structure representing a set of nodes V and a set of edges E, each of which is a two element subset of V. Edges are represented by $e = (u, v)$, where u and v are known as *ends* of e. E is also known as the adjacency relation. If each edge $e = (u, v)$ of a graph $G = (V, E)$ is an unordered pair of nodes u and v, G is known as an *undirected graph*. It represents a symmetric relationship between its ends u and v. Figure 2.1 shows an example of an undirected graph having 5 nodes. It is the graph $G = (V, E)$ where $V = \{A, B, C, D, E\}$ and $E = \{(A, B), (A, E), (B, C), (B, D), (C, D), (D, E)\}$.

If each edge $e = (u, v)$ of a graph $G = (V, E)$ is an ordered pair of nodes u and v, G is known as a *directed graph*. It represents an asymmetric relationship between ends u and v. For an edge

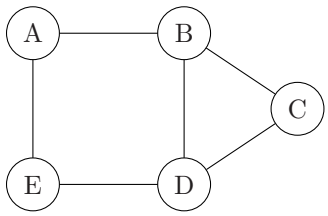

Figure 2.1: Undirected Graph

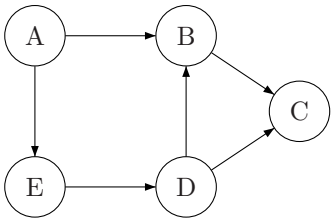

Figure 2.2: Directed Graph

$e = (u, v)$ in a directed graph, u is called the *tail* and v is called the *head* of e. Also, e is said to be *leaving* node u and *entering* node v. Figure 2.2 shows an example of a directed graph. It is the graph $G = (V, E)$ where $V = \{A, B, C, D, E\}$ and $E = \{(A, B), (A, E), (B, C), (D, B), (D, C), (E, D)\}$.

For an undirected graph, the *degree* $d[v]$ of node v is the number of edges incident on it. For example, in Figure 2.1, the degrees of nodes A, B, C, D and E are 2, 3, 2, 3, and 2, respectively. For a directed graph, *in-degree* $d_i(v)$ and *out-degree* $d_o(v)$ of node v are defined as number of edges entering and leaving node v respectively. For example, in Figure 2.2, the in-degree and out-degree of node A are 0 and 2, of node B are 2 and 1, of node C are 2 and 0, of node D are 1 and 2, and of node E are 1 and 1 respectively.

A *path* from a node u to a node v in a graph $G = (V, E)$ is a sequence of edges (x_0, x_1), (x_1, x_2), ... (x_{n-1}, x_n) where $x_0 = u$ and $x_n = v$. The *length* of a path is the number of edges in it. For example, in Figures 2.1 and 2.2, there is a path (A, B) of length 1 from node A to node B and a path (A, B), (B, C) of length 2 from node A to node C. A path that starts and ends at the same node is a *cycle*. For example, in Figure 2.1, there are cycles (B, C), (C, D), (D, B) and (A, B), (B, D), (D, E), (E, A) of lengths 3 and 4, respectively. A cycle of length one is known as a *self-loop*. A graph with at least one cycle is known as a *cyclic graph*, as in Figure 2.1, and a graph with no cycle is known as an *acyclic graph*, as in Figure 2.2. A graph $G = (V, E)$ is *connected* if, for every $u, v \in V$, there exists a path from u to v. The graph in Figure 2.1 is a connected graph, and the graph in Figure 2.2 is not.

A *multigraph* is an ordered pair $G = (V, E)$ where, V is a set of nodes and E is a multiset of unordered pairs of nodes. Edges that have the same pairs of end points are known as *multiple edges*.

A network is made of entities and relationships between them. Graphs can be used to represent such relationships, such as the friendship network of people can be represented as an undirected net-

work, where each person is represented by a node and friendship between two persons is represented by an undirected edge between the corresponding nodes. Similarly, in the case of the network of the World Wide Web, each node represents a web page and each directed edge represents a hyperlink from the source page to the destination page. In the literature, these two terms, graphs and networks, are used quite interchangeably.

2.2 Degree Distribution

The *degree distribution* P_k of an undirected graph G is the probability that a node chosen uniformly at random has degree k. It is also the fraction of nodes with degree k. In the Erdös-Renyi model, the probability of having an edge between each pair of nodes in a network is p and hence the degree distribution is a Poisson distribution. However most real world networks follow a power-law degree distribution ($P_k \propto k^{-\alpha}$). Networks that follow a power-law degree distribution are called scale-free networks. Power-law degree distribution was first identified by Price [31] for a citation network with the value of α between 2.5 and 3. Later, a power-law distribution has been observed for networks such as the Internet, the collaboration network of movie actors [18], the World Wide Web, and the citation network of papers in physics [18, 26].

2.3 Clustering Coefficient

It has been observed in most real-world networks that, if there is an edge between nodes u and v and an edge between nodes u and w, then there is a higher probability that there is an edge between nodes v and w. In the context of a social network of friendship between individuals, it indicates that there is a higher probability that a friend of your friend is also your friend. This property is measured by transitivity or the clustering coefficient. There are multiple definitions of clustering coefficients used in the literature.

The number of *triangles* at node v is given by

$$\Delta(v) \quad = \quad |\{(u,w) \in E \mid (v,u) \in E \text{ and } (v,w) \in E\}|.$$

The number of *triples* at a node v is the number of paths of length two in which v is the central

node. Therefore, for a node v, the number of triples at node v is

$$\tau(v) = \binom{d[v]}{2}.$$

The *local clustering coefficient* $C(v)$ for v is defined as the ratio of the number of triangles connected at node v, $\Delta(v)$, to the number of connected triples at node v, $\tau(v)$ [39]. More precisely,

$$C(v) = \begin{cases} \frac{\Delta(v)}{\tau(v)} & \text{if } d[v] \geq 2; \\ 0 & \text{otherwise.} \end{cases}$$

The *clustering coefficient* $C(G)$ for the graph $G(V, E)$ is defined as the average of the clustering coefficients of its nodes. That is,

$$C(G) = \frac{1}{|V|} \sum_{v \in V} C(v).$$

As each triangle contains three nodes, the number of triangles in the graph G is

$$\Delta(G) = \frac{\sum_{v \in V} \Delta(v)}{3}.$$

The number of triples in graph G is

$$\tau(G) = \sum_{v \in V} \tau(v).$$

The *global clustering coefficient* $C_g(G)$ or *transitivity* $T(G)$ is defined as the ratio of three times the number of triangles in the graph, $\Delta(G)$, to the total number of connected triples in the graph, $\tau(G)$. That is,

$$\begin{aligned} C_g(G) &= T(G) \\ &= \frac{3 \times \Delta(G)}{\tau(G)} \\ &= \frac{\sum_{v \in V} \Delta(v)}{\sum_{v \in V} \tau(v)}. \end{aligned}$$

Pastor-Satorras et al. [30] measured the clustering coefficient for the Internet at the autonomous level for the period of three years from year 1997 to 1999 and found its value to be in the range 0.18-0.24. Watts and Strogatz [39] studied the clustering coefficient for the movie actor network, the power grid and the C. elegans network and found values of 0.79, 0.08, and 0.28, respectively. These are higher than the corresponding values 0.00027, 0.005, and 0.05 for a random graph with the same number of nodes and average degree.

We use use the global clustering coefficient for all our algorithms and hence the term clustering coefficient refers to the global clustering coefficient or the transitivity for chapter 6 onwards.

2.4 Small-World Effect

Milgram's experiment was the first demonstration of the small-world effect in networks, indicating that most pairs of nodes are separated by a short distance. It has an obvious implication on the dynamic processes taking place on networks, such as the spread of a rumor or an epidemic, suggesting that a rumor or epidemic will spread rapidly in a small-world network. Consider an undirected network. The *mean geodesic (shortest) distance* (l) of a network is defined as the average of the shortest distances between all pairs of nodes in the network. For a network with n nodes, where nodes are labelled as integers from 1 to n, the mean geodesic distance l is given by

$$l = \frac{1}{\binom{n}{2}} \sum_{i>j} d_{ij},$$

where d_{ij} is the shortest distance between node i and node j. For networks with more than one component, the shortest distance d_{ij} becomes infinite when nodes i and j are not in the same component. It leads to an infinite mean geodesic distance. Hence an alternative definition for mean geodesic distance is as given below [21]:

$$l^{-1} = \frac{1}{\binom{n}{2}} \sum_{i>j} d_{ij}^{-1}$$

A network of n nodes is said to have the *small-world effect* if the mean shortest distance is bounded by $O(\log n)$. Bollobas and Riordan [5] prove that it is a $O(\log n/(\log(\log n)))$ function of the number of vertices for scale free networks. Kleinberg et al.[22, 23] evaluate network diameter empirically for citation graphs in the physics literature, a citation graph for U.S. patents, a graph of the Internet, and a co-authorship network evolution. They show that in the growing network, the number of edges increases superlinearly over the number of nodes and diameter actually decreases with time.

Chapter 3

Real-World Networks

3.1 Social Networks

Social networks have been studied by researchers for more than a century. A social network is a set of people with interactions among them. It can be a network of friendship between individuals or a collaboration network of scientists. Due to the availability of large data sets, the recent work in this area has been focused on collaboration networks, where people are grouped together and an edge is placed between two persons if they collaborate or are in the same group. One such network is a co-authorship network of scientists [28], where there is an edge between two authors if they have co-authored a paper. Another example is a collaboration network of movie actors (from the Internet Movie Database) [39], where two actors are connected if they have appeared in the same film. A network of email communications in a university environment has also been studied [10]. An important result in this area is from Milgram's small-world experiment [24]. In this experiment, individuals were asked to pass a letter to a specific person. It was observed that most of the letters that reached their targets were passed through six people on average, leading to the concept of "six degrees of separation" between nodes.

3.2 Technological Networks

Technological networks are designed for distribution or physical movement of some resource; these networks include the Internet, the power grid, and railways. The most important technological network studied extensively is the Internet, a network of physical connections over which binary information is transferred between computers. It is studied at two levels, a fine level where connections among all routers are considered, and a coarse level (also known as the autonomous level) where connection between only autonomous systems or domains are considered [18]. Both networks show a skewed degree distribution, where most of the nodes have few edges while a few nodes have many more edges. Power grid networks are also studied by Watts and Strogatz [39].

3.3 Information Networks

One of the most studied information networks is the citation network of scientific papers. In this network, each node represents a paper and there is an edge from node A to node B if paper A cites paper B. It is a directed acyclic network as a paper can cite only previously published papers. An important discovery by Price was that the in and out-degree distribution in this network follows a power-law [19]. In addition to the physical Internet network, a virtual network of the World Wide Web has been studied extensively [19, 20]. Here, nodes represent web pages, and there exists an edge from one web page to an other web page if there is a hyperlink in one web page (from page) to the other (to page). Unlike a citation network, it is a cyclic network. Its degree distribution also follows a power-law [4]. Another important discovery in this area is that the mean geodesic distance between all pairs of nodes in the network of the World Wide Web is approximately 16 [6]. Hence, it shows a small-world property defined as the shortest average distances between pairs of nodes in the network.

3.4 Biological Networks

Recently, interest in the study of biological networks has increased a lot. Most studied biological networks are gene regulatory networks, networks of metabolic reactions, protein interaction networks, food webs, and neural networks. A neural network may be considered similar to a technological network, a network of signals over neurons analogous to the Internet. In food webs, each node

represents a species in an ecological system and an edge between two species represents a predator-prey relationship. These networks also follow a power-law degree distribution [7, 13]. However, there is an important difference between biological networks and other types of real world networks. Most of the real world networks do not have an upper bound on the number of nodes, and analytical results are obtained for a very large network size, as the number of nodes n grows to infinity, but, in most biological networks, the number of nodes is bounded by a finite number, such as a gene regulatory network, which has a finite number of genes as nodes.

Chapter 4

Graph Generation Models

4.1 Poisson Random Graphs or the Erdös-Renyi Model

Solomonoff and Rapoport [35] and Erdös and Renyi [11] independently presented a simple model for graph generation. A network consists of n nodes and each pair of nodes is connected independently with probability p. This model is also known as the $G_{n,p}$ model. The probability of having a graph with n nodes and m edges is $\binom{M}{m}p^m(1-p)^{M-m}$, where M = $\binom{n}{2}$, which is the maximum possible number of edges. The degree distribution is approximately Poisson, as the probability of a vertex having degree k is

$$P_k = \binom{n}{k}p^k(1-p)^{n-k}.$$

The only real-world network property matched by this model is the small-world property. The mean number of neighbors of a vertex is $z = p(n-1)$ and hence the mean number of vertices at distance l is approximately z^l (assuming that none of these nodes share a common neighbors). So the mean distance through the network $l = \log n / \log z$, which increases logarithmically with the number of nodes n and hence represents a small-world property. Both clustering coefficient $C(G)$ and transitivity $T(G)$ are p, as the probability of a connection between two vertices is p whether they have a common neighbor or not.

4.2 The Configuration Model

The configuration model was proposed by Molloy and Reed [25]. Suppose there are n nodes labelled as $1, 2, \ldots, n$ and a degree distribution is given to be $\{d[1], d[2], \ldots, d[n]\}$, where a node v has degree $d[v]$ such that $\sum_{v=1}^{n} d[v] =$ even. Now a network with n nodes and given degree sequence is formed as follows. Form a set L containing $d[v]$ distinct copies of each node v labelled as $v_1,\ v_2,\ \ldots\ v_{d[v]}$. Then choose two elements u_k and $v_{k'}$ at random from this set and add an edge between them. Each element is connected to exactly one other element. Repeat the same process until each element is part of some edge.

For two nodes u and v, the number of edges joining them is the number of edges in the matching of L joining copies of u to copies of v. Hence, it produces a multigraph. This model produces each multigraph with the given degree sequence with equal probability. A simple graph can be obtained from a multigraph by merging multiedges and removing self-loops.

4.3 Small-world Networks

Strogatz and Watts [39] proposed a model that incorporates high transitivity based on Milgram's small-world experiment. It starts with a network built on a low dimensional regular lattice. Each edge is rewired with some probability p to create short cuts, thus decreasing the mean geodesic distance. The best studied example is the one dimensional lattice with L nodes, where each node is connected to its neighbors at distance k or less. Hence, the degree of each node is $2k$, and the network has Lk edges. Then, for each edge, with probability p, one end of the edge is moved to a node chosen uniformly at random from the lattice. This process is known as rewiring.

If $p = 0$, then it is a complete regular lattice with clustering coefficient $C = (3k - 3)/(4k - 2)$ and diameter $L/4k$ for large L. If $p = 1$ then it is a completely random graph with diameter $\log L/\log k$ but very low clustering coefficient $C = 2k/L$. Between these two extremes, there exists a region that gives low diameter with high clustering coefficient.

Newman and Watts [29] propose a variant of this model where for each edge in the network, with probability p, a new edge is added between randomly chosen pairs of nodes. So the total number of short cuts (randomly added edges) is Lkp, and the mean degree is $2k(1 + p)$. The clustering

co-efficient for this network without rewiring is given by

$$C = \frac{3(k-1)}{2(2k-1) + 4kp(p+2)}$$

Degree distribution of this model does not reflect real-world networks as each node has degree $2k$ for edges of regular lattice, plus binomially distributed edges for short cuts. The probability p_j that a node chosen at random has degree j is

$$p_j = \begin{cases} \binom{L}{j-2k} \left[\frac{2kp}{L}\right]^{j-2k} \left[1 - \frac{2kp}{L}\right]^{L-j+2k} & \text{for } j \geq 2k; \\ 0 & \text{for } j < 2k; \end{cases}$$

4.4 Preferential Attachment Model

The preferential attachment model was first described by Price [31] for the citation network, which builds on the "rich-get-richer" phenomenon. The intuition behind this model is that the rate at which a paper is cited is proportional to its in-degree, as the probability that one comes across a particular paper while reading the literature depends upon the number of papers citing it. Price described it as "cumulative advantage".

It was rediscovered by Barabasi and Albert [3] and named "preferential attachment". This model is the same as Price's model except that it consider the network as an undirected graph. At each time stamp a new node is added with m edges to the existing nodes. The probability P that a new node will be connected to a node v is proportional to the degree of v, $d[v]$. So $P(d[v]) = d[v]/\sum_{i=1}^{n} d[i]$. Barabasi [3] showed that it produces a network with a degree distribution that follows a power law ($p_k \propto k^{-\alpha}$) with exponent $\alpha = 3$.

Chapter 5

Related Work

Several researchers have proposed models based on preferential attachment to generate a graph with power-law degree distribution and high clustering. Holme and Kim [16] offer a model in which a node v and m edges are added at each time step. First a PA (preferential attachment) step (a new node v is attached to an existing node w with probability proportional to the degree of w) is performed. Then, a TF (triad formation) step (connecting new node v to a randomly chosen neighbor u of the previously selected node w in the PA step) is performed with probability p, and another PA step is performed with probability $(1-p)$. The average number of TF steps per node is $p(m-1)$. The clustering coefficient of the generated graph can be tuned by varying the value of p. Guo et al. [14] propose a similar model where the network starts with m_0 connected nodes. At each time step, a new node v is added with m edges. In the first step, node v is connected to one of the existing nodes w with probability proportional to degree of w, and, in the second step, with probability p_t a neighbor s of w is chosen with probability p_s given by

$$p_s = \frac{d[s]^\beta}{\sum_{i \in r_w} d[i]^\beta}$$

where, $d[i]$ is degree on node i, and r_w is the set of neighbors of node w. These steps are repeated m times to create m edges. So the number of triangles at a newly added node v is $p_t m$, and the clustering coefficient of the generated graph can be varied by varying p_t.

The model proposed by Wang et al.[38] is similar to Holme and Kim's model but with the local world concept. The local world $v(s)$ of node v is the set of all nodes at distance s or less from node

v. In this model, at each time step, a new node v is added with m edges. A node u is chosen at random from the existing nodes, and the local world of newly added node v is set to be $u(s)$. Node v is connected to a node $w \in v(s)$ according to preferential attachment. Then with some probability p, node v is connected to the highest degree neighbor of node w, and, with probability $(1-p)$, the PA step is repeated. Here also, the clustering coefficient can be varied by varying p. Wang et al. [37] also propose a model based on a local concept. In this model, network initially start with n_0 nodes and e_0 edges. At each step, M nodes are selected from the present network and are referred to as the local world of the new node. With probability p, a new node v is added and m edges are added from v to the previously existing nodes according to the preferential attachment. With probability $1-p$, m edges are added by selecting an existing node w according to preferential attachment and an edge is randomly added among its neighbors. Here also the clustering coefficient of the generated graph can be varied by varying the value of p. These models produce a connected graph with power-law degree distribution.

Schank and Wagner [34] propose a model that starts with two connected nodes. At each time step, a new node is added with m edges to previously existing nodes according to preferential attachment. Then, for each node $v \in V$, two neighbors u and w are chosen at random and are connected. This step is repeated o times for each node. Varying the clustering coefficient can be achieved by varying the value of o. But a real world network may not be connected, or the degree distribution may vary from a power-law. It remains an open problem to generate a graph with an arbitrary degree distribution and tunable clustering coefficient.

Gleeson [12] proposes a model that takes joint probability distribution $\gamma(k, c)$ (a randomly chosen node in the graph has degree k and is part of a c-clique) as an input. It assumes that each node is part of at most one clique and hence each node has $(c-1)$ clique edges and $(k-c+1)$ external edges. A graph is modeled as disjoint cliques connected by external edges. Each clique can be thought of as a super node and a graph can be generated by connecting stubs corresponding to external edges similar to the configuration model of Molloy et al. [25]. Though it gives a method to generate a graph with not only triangle structure but any subgraphs, it is not a realistic representation of real world networks, as a node may be part of more than one clique or subgraph.

Guo and Krains [15] propose a model to generate a directed graph where in-degrees and out-degrees follow a power-law. The clustering coefficient used here is the average of the clustering coefficient of the nodes $C(G)$ in the graph. In this model, based on the in-degree and out-degree

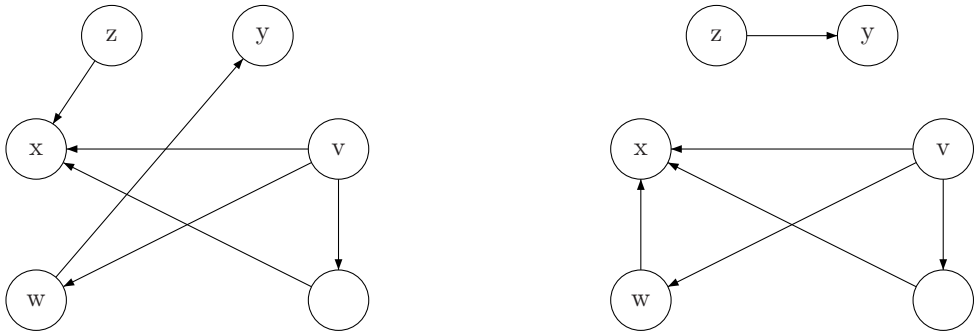

Figure 5.1: Increasing clustering coefficient using Guo and Krains' Algorithm

power exponent, the number of nodes n and the number of edges m, in-degree and out-degree sequence is constructed. Then, a simple random directed graph is generated by connecting in-stubs and out-stubs randomly similar to the configuration model [25] without multi-edges (two edges connecting in the same direction and same pair of nodes) and self-loops. If the average clustering coefficient of the generated graph is lower than the input clustering coefficient, then edge-switching is done as follows to increase the value of clustering coefficient. Choose a random node v and four distinct nodes w, x, y, and z such that they meet the following conditions:

1. w and x are neighbors of v.
2. y and z are not neighbors of v.
3. $(w, x) \notin E$ and $(z, y) \notin E$.
4. $(w, y) \in E$ and $(z, x) \in E$.
5. $C(v)$ of neighbors shared by w and $y <$ target clustering coefficient.
6. $C(v)$ of neighbors shared by x and $z <$ target clustering coefficient.

Delete edges (w, y) and (z, x) from E, and add edges (w, x) and (z, y) to E which increases the local clustering coefficient of the randomly chosen node v as shown in Figure 5.1. This process is repeated until either the clustering coefficient of the generated graph is greater than or equal to the target clustering coefficient, or it reaches a certain predefined number of trials.

Newman [17] proposes an algorithm to generate a random graph with tunable clustering coefficient. It is based on the configuration model [25]. Instead of taking a degree sequence as input, it takes a triangle (which represents corners) degree sequence and a single edge (which represents stubs) degree sequence as input. However this algorithm does not specify a few important details, such as the probability with which a node should be chosen to form triangles and single edges and whether it always chooses three distinct nodes for creating triangles and two distinct nodes for single edges or not. Typically, the information about triangle and single edge degree is not available, only information about the degree sequence is available. See Chapter 7 for more details.

Chapter 6

Power-law Graph Generation

As discussed earlier, many real-world networks follow a power-law degree distribution. Hence, we conducted an experiment to observe the naturally occurring range of clustering coefficient for power-law networks.

6.1 Algorithm

We use the preferential attachment model proposed by Barabasi and Albert [3] to generate power-law graphs. In this model, at each time stamp a new node is added with m edges to the existing nodes. The probability P that a new node will be connected to a node v is proportional to $d[v]$. Reka et al [33] show experimentally that the clustering coefficient of the graph generated by Barabasi-Albert model decreases with the increases in number of nodes as per the power-law ($C = n^{0.75}$).

Pseudocode for this algorithm is as given in Figure 6.1. A graph with n nodes labelled as $1, 2, \ldots, n$ is created as follows. It starts with V nodes and E edges as given below:

$$\begin{aligned} V &= \{1, 2, \ldots n_0\} \\ E &= \{(1,2),\ (2,3),\ \ldots\ (n_0 - 1, n_0),\ (n_0, 1)\}, \end{aligned}$$

which form a cycle with these nodes. Then, at each time stamp, one node i is added along with m edges connecting it to n_0 different previously existing nodes. Hence, it requires $n > n_0$. Here, we set $n_0 = m + 2$. This process is repeated $n - n_0$ times to have a total of n nodes. For creating an

edge to a previously existing node v, it is selected at random with probability $\frac{d[v]}{\sum_{k=1}^{i-1} d[k]}$.

At each step in the algorithm, the array $PS[v]$ holds a value $\sum_{k=1}^{v} d[k]$ for each node v existing at that time stamp. When a new node i is added, there are $i-1$ nodes existing previously. $RandomNumber(1, PS[i-1])$ generates a random number r from the range 1 to $\sum_{k=1}^{i-1} d[k]$. At this point, $PS[i-1]$ can be seen as a set S having $d[k]$ copies of each node $k = 1, 2, \ldots, i-1$ and $|S| = \sum_{k=1}^{i-1} d[k]$. $RandomNumber(1, PS[i-1])$ selects one copy from above set S. As set S has $d[v]$ copies of node v, the probability of selecting it is proportional to $d[v]$. Once a copy is selected from set S, subroutine FIND-CNode is used to find corresponding node.

Suppose we want to generate a graph with input parameters $n = 6$ and $m = 2$ and $n_0 = 4$. The algorithm starts with creating nodes $V = \{1, 2, 3, 4\}$ and edges $E = \{(1, 2),\ (2, 3),\ (3, 4),\ (4, 1)\}$, which forms a cycle. The degree of each node and content of array PS is as given below:

Node i	$degree[i]$	$PSm[i]$
1	2	2
2	2	4
3	2	6
4	2	8

Table 6.1: Degree and PrefixSum for each node i = 1 to 4 initially.

The next step is to create a node $i = 5$ and two edges incident upon it. A random number r is selected in the range 1 and $\sum_{k=1}^{i-1} d[k]$, which is between 1 and 8. Suppose, the random number selected is $r = 3$. The next step is to identify which existing node does this number r correspond to. From algorithm FIND-CNODE $(r, high)$ (Figure 6.2), it can be seen that $r = 3$ corresponds to node $v = 2$. As node $i = 5$ is not connected to any node at this point, an edge $(5, 2)$ is added to the edge set E. The contents of sets V and E are as given below:

$$\begin{aligned} V &= \{1, 2, 3, 4, 5\} \\ E &= \{(1, 2),\ (2, 3),\ (3, 4),\ (4, 1),\ (5, 2)\} \end{aligned}$$

The degree of each node and the contents of array PS are modified to reflect this edge addition as shown below:

```
PREF-ATTACH(n, m)
 1  ▷ input: number of nodes n, number of edges that need to be added with each node m
 2  ▷ output: graph (V, E), where V is a set of nodes and E is a set of edges
 3  n_0 ← m + 2                                  ▷ initial number of nodes
 4  V ← {1, 2, ... n_0}
 5  E ← {(1, n_0)} ∪ {(i, i + 1) | 1 ≤ i ≤ n_0 − 1}
 6  for i ← 1 to n_0
 7        do PS[i] ← Σ_{k=1}^{i} d[k]                ▷ prefix sum array
 8  for i ← n_0 + 1 to n
 9        do V ← V ∪ {i}
10           e ← 0
11           while e < m
12                 do r ← RandomNumber(1, PS[i − 1])
13                    v ← FIND-CNODE(r, PS, i − 1)
14                    if (i, v) ∉ E
15                       then E ← E ∪ {(i, v)}
16                            e ← e + 1
17                            for j ← v to i
18                                 do PS[k] ← Σ_{k=1}^{j} d[k]
19  return (V, E)
```

Figure 6.1: Algorithm PREF-ATTACH

```
FIND-CNODE(r, PS, high)
 1  ▷ input: random number r, prefix sum array PS, highest subscript of the array PS high
 2  ▷ output: node corresponding to the random number r
 3  low ← 1
 4  while high < low
 5        do mid ← (high + low)/2
 6           if r ≤ PS[low]
 7              then return low
 8           elseif low = mid
 9              then return high
10           elseif r ≤ PS[mid]
11              then high ← mid
12           else low ← mid
13  return low
```

Figure 6.2: Algorithm FIND-CNODE

Node i	$degree[i]$	$PS[i]$
1	2	2
2	3	5
3	2	7
4	2	9
5	1	10

Table 6.2: Degree and PrefixSum for each node i = 1 to 5 after adding node 5.

Now, the second edge $m = 2$ is added to node $i = 5$. Suppose a random number selected is $r = 5$ and node corresponding to it is $v = 2$. But as there already exists an edge between nodes 5 and 2, it is ignored and a random number r is selected again until there does not exist an edge between the corresponding node v and node i. Suppose the next number selected is $r = 6$ and the corresponding node $v = 3$. Then, as there does not exist an edge between nodes 5 and 3, an edge $(5, 3)$ is added to the edge set E. The modified sets V, E, degrees of nodes, and PS are as shown below:

$$\begin{aligned} V &= \{1, 2, 3, 4, 5\} \\ E &= \{(1, 2),\ (2, 3),\ (3, 4),\ (4, 1),\ (5, 2),\ (5, 3)\} \end{aligned}$$

Node i	$degree[i]$	$PS[i]$
1	2	2
2	3	5
3	2	8
4	2	10
5	2	12

Table 6.3: Degree and PrefixSum for each node i = 1 to 5 after adding second edge at node 5.

The similar process is repeated for creating nodes 5 and 6, and edges incident upon them. The graph generated by this algorithm is undirected. To circumvent the problem of an infinite while loop in the case when all nodes are connected to a new node, we require $n_0 > m$. The definition of

clustering coefficient used here is the definition of global clustering coefficient $C_g(G)$ or transitivity $T(G)$, the ratio of total number of triangles in a graph to the total number of possible triangles in the graph.

6.2 Empirical Results

We tested the above algorithm for a number of nodes varying from $n = 1000$ to 5000 in a step of 1000 and a number of edges being added with each node from $m = 2$ to 5. We generated graphs 30 times for each case and measured average clustering coefficient and standard deviation.

As expected for graphs with power-law degree distribution, our result also show that for a given m, as the number of nodes increases, the clustering coefficient tends to decrease. Also for a given number of nodes n, the mean clustering coefficient tends to increase as we increase m, the number of edges being added with each node.

n	m = 2	m = 3	m = 4	m = 5
1000	0.0095	0.0166	0.0226	0.0282
2000	0.0056	0.0097	0.0132	0.0169
3000	0.0038	0.0070	0.0098	0.0123
4000	0.0032	0.0056	0.0079	0.0098
5000	0.0026	0.0047	0.0065	0.0083

Table 6.4: Clustering coefficient versus number of nodes n and m

6.3 Future Work

As shown by Cooper and Frieze [9], as the number of edges added with a node m changes, the power (coefficient α) of the graph also changes. So power may be adjusted potentially to get the desired clustering coefficient. We plan to carry out more experiments to observe and may be utilize this fact to tune the clustering coefficient.

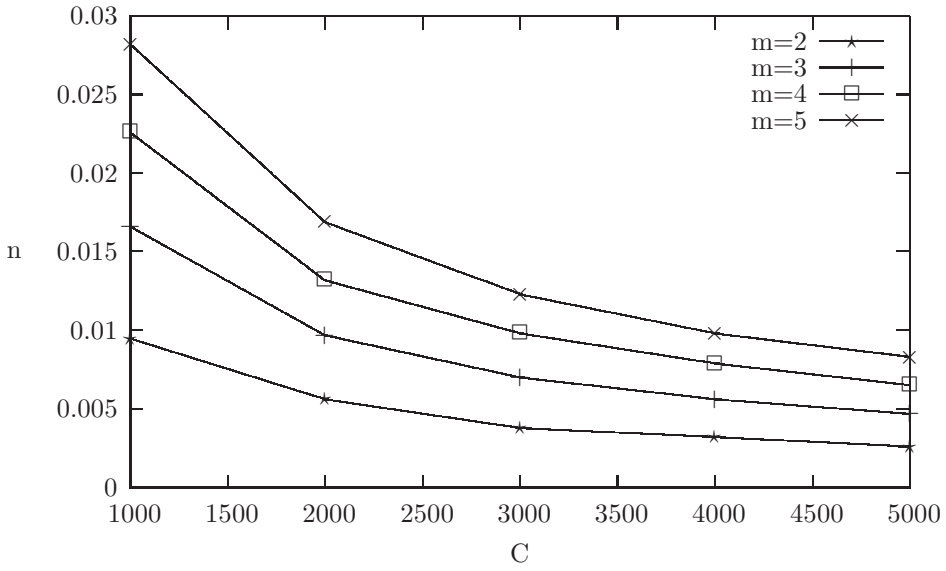

Figure 6.3: Clustering coefficient versus number of nodes.

Chapter 7

Newman's Algorithm

7.1 Algorithm

Newman [17] proposes an algorithm to generate a graph with tunable clustering coefficient and given degree sequence. It is based on the configuration model [25]. In this algorithm, for each node v, the number of triangles $t[v]$ it is part of and the number of single edges $s[v]$ incident on it that do not participate in any triangle are given as input. Here $t[v]$ and $s[v]$ are such that $\sum_{v \in V} t[v]$ is a multiple of 3 and $\sum_{v \in V} s[v]$ is a multiple of 2. As the algorithm is based on the configuration model, each node v can be seen as consisting of $s[v]$ stubs and $t[v]$ corners. Then a triangle is created by selecting three corners and the corresponding nodes u, v, and w at random and joining them. This process is repeated until each corner is part of a unique triangle. An edge is created by selecting two stubs and the corresponding nodes u and v at random and creating an edge between them. This process is repeated until each stub is part of a unique edge.

As we choose unconnected stubs or corners to form edges or triangles, we keep track of the number of unconnected stubs and corners for each node v. The number of *unconnected stubs* at a node v is the difference between the target single edges $s[v]$ and the current single edges at that node. It is also known as the *residual single edge degree* $rs[v]$. Similarly, the number of *unconnected corners* at a node v is the difference between the target triangle degree $t[v]$ and the current triangle degree at that node. It is know as the *residual triangle degree* $rt[v]$.

In the configuration model, the number of edges between nodes u and v is the number of their

stubs that are connected to each other. Hence it can produce multiple edges and self-loops. But here we are interested in simple graphs without multiple edges and self-loops. So, we discard self-loops and merge multiple edges.

7.2 Inspiration for four versions

Newman [17] does not specify a few important details, such as the probability with which a node should be chosen to form triangles and single edges, and whether it always chooses three distinct nodes for creating triangles and two distinct nodes for single edges or not.

Each node v can be chosen with an equal probability $\frac{1}{n}$ to form triangles and single edges (Algorithm Conf-1) or each node v can be chosen to form triangles with probability proportional to its residual triangle degree $rt[v]$ and to form single edges with probability proportional to its residual single edge degree $rs[v]$ (Algorithm Conf-2). Both of these algorithms choose three distinct nodes for creating triangles and two distinct nodes for creating edges. They also stick to the input triangle and single edge degree sequence. So, at any time, node v is allowed to be a part of a triangle or a single edge only if its residual triangle degree $rt[v]$ or residual single edge degree $rs[v]$ is greater than 0 respectively. Now consider a situation when $rt[u] = 2$, $rt[v] = 1$ and all $rt[w] = 0$ for all other nodes w. In this case, we can not choose three distinct nodes for creating a triangle but we still have some unconnected corners. Hence the algorithm tries to create triangles and fails to terminate. A similar situation may occur while creating single edges if residual single edge degree is greater than 0 for one node and 0 for all others.

To avoid an infinite loop, algorithms should break after some predefined number of unsuccessful trials (as in algorithms Conf-1 and Conf-2) or they should not stick to the input triangle and single edge degree sequence. In the later case, each node v is chosen for creating triangles and single edges with probability proportional to its triangle degree $t[v]$ and single edge degree $s[v]$, hence, it is allowed to be part of triangles and single edges even if its residual triangle degree $rt[v]$ and $rs[v]$ is equal to 0 or negative (as in algorithms Throw-1 and Throw-2). Throw-1 chooses three distinct nodes for creating triangles and two distinct nodes for creating edges respectively, while, for algorithm Throw-2, the nodes chosen to create triangles and edges need not be distinct.

All four algorithms take the number of nodes n, the array t, which holds the number of triangles each node is part of, and the array s, which holds the number of edges for each node that do not

participate in triangles as input. These arrays constitute the triangle degree sequence and the single edges degree sequence. The algorithms start with n nodes and an empty edge set E. Initially, all nodes are isolated. So the current degree of each node $v \in V$ is 0, and the residual triangle degree $rt[v]$ and residual single edge degree $rs[v]$ of each node v are the same as the input triangle degree $t[v]$ and single edge degree $s[v]$, respectively. The total number of triangles that need to be created is given by $T = \frac{\sum_{v \in V} t[v]}{3}$, and the total the number of single edges that need to be created is given by $M = \frac{\sum_{v \in V} s[v]}{2}$. The next steps are to create T triangles and M edges. CONF-1, CONF-2, THROW-1 and THROW-2 differ in these steps. The definition of clustering coefficient used for evaluation is global clustering coefficient $C_g(G)$ or transitivity $T(G)$ in this Chapter and the next Chapter.

7.3 Configuration Algorithm-1 CONF-1

As shown in Figure 7.1 CONF-1, first chooses three distinct nodes u, v, and w at random. Here, each node is chosen with the same probability $\frac{1}{n}$. This process is repeated until T triangles are created. For creating single edges, it chooses two distinct nodes u and v at random. Again here, each node is chosen with the same probability $\frac{1}{n}$. This process is repeated until M edges are created.

Time taken by lines 7 and 8 are $O(n) = O(|V|)$. The number of triangles to create T and the number of single edges to create M are together bounded by the number of edges E. Hence, total running time of CONF-1 is bounded by $O(|V| + |E|)$.

7.4 Configuration Algorithm-2 CONF-2

As shown in Figure 7.2, CONF-2 first chooses three distinct nodes u, v, and w with probability proportional to their residual triangle degree. Here, each node v is chosen with probability $\frac{rt[v]}{\sum_{x \in V} rt[x]}$. This process is repeated until T triangles are created. For creating single edges, CONF-2 chooses two distinct nodes u and v with probability proportional to their residual single edge degree. Again, each node v is chosen with probability $\frac{rs[v]}{\sum_{x \in V} rs[x]}$. This process is repeated until M edges are created.

Time taken by lines 7 and 8 is $O(n) = O(|V|)$. The number of triangles to create T and the number of single edges to create M are together bounded by the number of edges $|E|$. In loops

CONF-1(n, s, t)

1 $\triangleright$ input: number of nodes n, single edge and triangle degree sequence arrays s and t
2 $\triangleright$ output: graph (V, E), where V is a set of nodes and E a is set of edges
3 $V \leftarrow \{1,\ 2,\ \ldots\ n\}$
4 $E \leftarrow \phi$
5 $rt \leftarrow t$ $\triangleright$ residual triangle degree array
6 $rs \leftarrow s$ $\triangleright$ residual single edge degree array
7 $T \leftarrow \frac{\sum_{i=1}^{n} t[i]}{3}$ $\triangleright$ number of triangles to create
8 $M \leftarrow \frac{\sum_{i=1}^{n} s[i]}{2}$ $\triangleright$ number of single edges to create
9 **while** $T > 0$
10 **do** Choose three distinct nodes $u,\ v,\ w$ from 1 to n with equal probability $(\frac{1}{n})$
11 **if** $rt[u] > 0$, $rt[v] > 0$ and $rt[w] > 0$
12 **then** $E \leftarrow E\ \cup\ \{(u,\ v),\ (v,\ w),\ (u,\ w)\}$
13 $rt[u] \leftarrow rt[u] - 1$
14 $rt[v] \leftarrow rt[v] - 1$
15 $rt[w] \leftarrow rt[w] - 1$
16 $T \leftarrow T - 1$
17 **while** $M > 0$
18 **do** Choose two distinct nodes $u,\ v$ from 1 to n
19 **if** $rs[u] > 0$ and $rs[v] > 0$
20 **then** $E \leftarrow E\ \cup\ \{(u,\ v)\}$
21 $rs[u] \leftarrow rs[u] - 1$
22 $rs[v] \leftarrow rs[v] - 1$
23 $M \leftarrow M - 1$
24 **return** (V, E)

Figure 7.1: Algorithm CONF-1

CONF-2(n, s, t)

1 $\triangleright$ input: number of nodes n, single edge and triangle degree sequence arrays s and t
2 $\triangleright$ output: graph (V, E), where V is a set of nodes and E a is set of edges
3 $V \leftarrow \{1,\ 2,\ \ldots\ n\}$
4 $E \leftarrow \phi$
5 $rt \leftarrow t$ $\triangleright$ residual triangle degree array
6 $rs \leftarrow s$ $\triangleright$ residual single edge degree array
7 $T \leftarrow \frac{\sum_{i=1}^{n} t[i]}{3}$ $\triangleright$ number of triangles to create
8 $M \leftarrow \frac{\sum_{i=1}^{n} s[i]}{2}$ $\triangleright$ number of single edges to create
9 **while** $T > 0$
10 **do** Choose three distinct nodes
11 u with probability $\frac{rt[u]}{\sum_{i=1}^{n} rt[i]}$
12 v with probability $\frac{rt[v]}{\sum_{i=1}^{n} rt[i]}$
13 w with probability $\frac{rt[w]}{\sum_{i=1}^{n} rt[i]}$
14 $E \leftarrow E\ \cup\ \{(u,\ v),\ (v,\ w),\ (u,\ w)\}$
15 $rt[u] \leftarrow rt[u] - 1$
16 $rt[v] \leftarrow rt[v] - 1$
17 $rt[w] \leftarrow rt[w] - 1$
18 $T \leftarrow T - 1$
19 **while** $M > 0$
20 **do** Choose two distinct nodes
21 u with probability $\frac{rs[u]}{\sum_{i=1}^{n} rs[i]}$
22 v with probability $\frac{rs[v]}{\sum_{i=1}^{n} rs[i]}$
23 $E \leftarrow E\ \cup\ \{(u,\ v)\}$
24 $rs[u] \leftarrow rs[u] - 1$
25 $rs[v] \leftarrow rs[v] - 1$
26 $M \leftarrow M - 1$
27 **return** (V, E)

Figure 7.2: Algorithm CONF-2

starting at lines 9 and 20, nodes are chosen with given probability using the cumulative distribution function (CDF). The CDF requires prefix sum arrays of residual triangle and single edge degrees. A random number is generated using this CDF and then the node corresponding to the random number generated is found using binary search on this array. The time taken to update the prefix sum arrays is bounded by $O(n) = O(|V|)$, and the time taken by the binary search is bounded by $O(\log n) = O(\log |V|)$. Hence, the total running time of CONF-2 is bounded by $O(|V||E|)$.

For the simplicity, we assume that each triangle requires two edges at each of its three nodes. It may not be true in the case when two or more triangles share an edge or when a triangle shares an edge with a single-edge. Assuming each triangle requires two edges at each node, for every $v \in V$, we have $d[v] \approx 2t[v] + s[v]$. Each node v can be part of $\binom{d[v]}{2}$ triples (or triangles at the most), hence the total number of triples in the generated graph is

$$\tau(G) = \sum_{v=1}^{n} \binom{d[v]}{2}. \tag{7.1}$$

Since, each node v has $t[v]$ triangles and as each triangle is counted three times for three nodes, the actual number of triangles in the generated graph is given by

$$\Delta(G) = \frac{\sum_{i=1}^{n} t[v]}{3}. \tag{7.2}$$

The *anticipated clustering coefficient* of the generated graph is

$$C_a(G) = \frac{3 \times \Delta(G)}{\tau(G)}. \tag{7.3}$$

7.5 Probabilistic Algorithm-1 THROW-1

As shown in Figure7.3, THROW-1 first chooses three nodes u, v and w, not necessarily distinct, with probability proportional to their triangle degrees and creates a triangle among them. Here, each node v is chosen with probability $\frac{t[v]}{\sum_{x \in V} t[x]}$. This process is repeated until T triangles are created. For creating single edges, it chooses two nodes u and v, not necessarily distinct, with probability proportional to their single edges degrees and creates an edge between them. Again here, each node

THROW-1(n, s, t)

1 $\triangleright$ input: number of nodes n, single edge and triangle degree sequence arrays s and t
2 $\triangleright$ output: graph (V, E), where V is a set of nodes and E a is set of edges
3 $V \leftarrow \{1,\ 2,\ \ldots\ n\}$
4 $E \leftarrow \phi$
5 $T \leftarrow \frac{\sum_{i=1}^{n} t[i]}{3}$ $\triangleright$ number of single edges to create
6 $M \leftarrow \frac{\sum_{i=1}^{n} s[i]}{2}$ $\triangleright$ number of triangles to create
7 **while** $T > 0$
8 **do** Choose three nodes
9 u with probability $\frac{t[u]}{\sum_{i=1}^{n} t[i]}$
10 v with probability $\frac{t[v]}{\sum_{i=1}^{n} t[i]}$
11 w with probability $\frac{t[w]}{\sum_{i=1}^{n} t[i]}$
12 $E \leftarrow E\ \cup\ \{(u,\ v),\ (v,\ w),\ (u,\ w)\}$
13 $T \leftarrow T - 1$
14 **while** $M > 0$
15 **do** Choose two nodes
16 u with probability $\frac{s[u]}{\sum_{i=1}^{n} s[i]}$
17 v with probability $\frac{s[v]}{\sum_{i=1}^{n} s[i]}$
18 $E \leftarrow E\ \cup\ \{(u,\ v)\}$
19 $M \leftarrow M - 1$
20 **return** (V, E)

Figure 7.3: Algorithm THROW-1

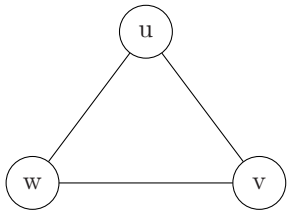

Figure 7.4: Algorithm THROW-1 creating triangles case a.

v is chosen with the probability $\frac{s[v]}{\sum_{x \in V} s[x]}$. This process is repeated until M edges are created. This algorithm just throws T triangles and M single edges and does not directly address triangle and single edge degrees of nodes.

Time taken by lines 5 and 6 is $O(n) = O(|V|)$. The number of triangles to create T and the number of single edges to create M are together bounded by the number of edges $|E|$. In loops starting at lines 7 and 14, nodes are chosen using the CDF (as explained in the discussion of CONF-2). Here, the prefix sum array remains constant and the time taken by binary search is bounded by $O(\log n) = O(\log |V|)$. Hence, the total running time of THROW-1 is bounded by $O(|E| \log |V| + |V|)$.

7.5.1 Throwing triangles only

Let us first analyze algorithm THROW-1 for the simple case when we only throw triangles, hence, for all $v \in V$, $t[v] = t$, where t is a constant, and $s[v] = 0$. This algorithm does not generate a regular graph of degree $2t$. We first calculate the expected degree distribution for the generated graph.

In this algorithm, we choose three nodes u, v, and w and create a triangle among them. Here, each node v is chosen with probability $\frac{t[v]}{\sum_{x \in V} t[x]}$ but as each node has the same triangle degree, each is chosen with probability equal to $\frac{1}{n}$. This process is repeated $T = \frac{\sum_{v \in V} t[v]}{3} = \frac{nt}{3}$ times to create T triangles. No single edges are added.

Let us see what happens when we throw a triangle. Here, while selecting three nodes, the algorithm does not ensure that they are distinct, i.e, there may be any of the following three cases:

a. Three distinct nodes u, v and w are chosen, which creates a triangle among u, v and w. The degrees of all three nodes u, v, and w may increase by 2 as shown in Figure 7.4.

b. One node u is chosen twice and a node v is chosen once, which creates an edge (u, v) and a

Figure 7.5: Algorithm THROW-1 creating triangles case b.

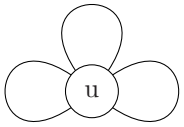

Figure 7.6: Algorithm THROW-1 creating triangles case c.

self-loop at node u as shown in Figure 7.5. We discard self-loops, and if there does not already exist an edge between nodes u and v, then the degrees of nodes u and v both may increase by 1. It there already exists and edge between nodes u and v then degrees of nodes u and v do not change.

c. The same node u is chosen three times, and hence there are three self-loops at node u as shown in Figure 7.6. Again we discard self-loops, and the degree of node u does not change.

Theorem 7.5.1. *For algorithm* THROW-1*, if each node has triangle degree equal to t and single edge degree equal to* 0*, then the probability that node v has degree equal to k is*

$$
\begin{aligned}
Pr[d[v]=k] &= \left[\left(1-\frac{1}{n}\right)^3+\left(\frac{1}{n}\right)^3\right]^T && \text{if } k=0\\
&\approx 3^{k'}\binom{T}{k'}\left(\frac{1}{n}\right)^{k'}\left(1-\frac{1}{n}\right)^{3T-3k'}A && \text{if } k>0 \text{ is even}\\
&\approx 3^{k'+1}\binom{T}{k'}\binom{T-k'}{1}\left(\frac{1}{n}\right)^{k'+1}\left(1-\frac{1}{n}\right)^{3T-3(k'+1)}A\,(B+D) && \text{if } k \text{ is odd,}
\end{aligned}
$$

where

$$\begin{aligned} k' &= \left\lfloor \frac{k}{2} \right\rfloor \\ A &= \left[\left(1 - \frac{1}{n}\right) \left(1 - \frac{2}{n}\right) \ldots \left(1 - \frac{2k'}{n}\right) \right] \\ B &= \left(1 - \frac{(2k'+1)}{n}\right) \left(\frac{2k'}{n}\right) \\ D &= \left(1 - \frac{(2k'+1)}{n}\right) \left(\frac{2}{n}\right). \end{aligned}$$

Proof. As seen earlier, when a triangle is created at node v, it could have one of the three possible degrees (2 in case of case-a triangle, 1 in case of case-b triangle, and 0 in case of case-c triangle). So, in this proof, we will first look at the probabilities that a node v has degrees 0, 1, and 2 to get an intuition about probability that a node v has degree equal to k.

A node v can have degree equal to 0, if it is not chosen in any of the T triangles or chosen three times for a particular triangle as in case c. The probability that a nodes is chosen three times while creating a triangle is $\left(\frac{1}{n}\right)^3$ and the probability that a node is not chosen while creating a triangle is $\left(1 - \frac{1}{n}\right)^3$. So,

$$Pr[d[v] = 0] = \left[\left(1 - \frac{1}{n}\right)^3 + \left(\frac{1}{n}\right)^3 \right]^T.$$

A node v can have degree equal to 1, if it is part of exactly one of the T triangles, where one node has been chosen twice as in case b. There are $\binom{T}{1}$ ways to choose one triangle to be created at node v. Suppose, node v is chosen twice, then the probability of choosing node v twice is $\frac{1}{n^2}$ and choosing any other node $u \in V - \{v\}$ is $\left(1 - \frac{1}{n}\right)$ while creating a triangle. There are three such permutations possible, v, v, u and, v, u, v and, u, v, v. Hence the probability that node v is chosen exactly twice while creating a triangle is $3\left(\frac{1}{n}\right)^2 \left(1 - \frac{1}{n}\right)$. If node v is chosen once, then, while creating a triangle, the probability of choosing node v is $\frac{1}{n}$ and choosing any node $u \in V - \{v\}$ is $\left(1 - \frac{1}{n}\right)$ and choosing u again is $\frac{1}{n}$. Here also three permutations are possible. Hence the probability that node v is chosen once and some other node u is chosen twice while creating a triangle is $3\left(\frac{1}{n}\right)^2 \left(1 - \frac{1}{n}\right)$. In all six cases, node v has degree equal to 1. The probability that node v does not participate in any of the other $T - 1$ triangles is $\left(1 - \frac{1}{n}\right)^{3T-3}$. So,

$$Pr[d[v] = 1] \approx 6\binom{T}{1} \left(\frac{1}{n}\right)^2 \left(1 - \frac{1}{n}\right) \left(1 - \frac{1}{n}\right)^{3T-3}.$$

A node v can have degree equal to 2, if it is part of exactly one of the T triangles, where three distinct nodes have been chosen as in case a. There are $\binom{T}{1}$ ways to choose one triangle to be created at node v. The probability of choosing node v is $\frac{1}{n}$. Then probability of choosing some node $u \in V - \{v\}$ is $\left(1 - \frac{1}{n}\right)$ and then choosing some other node $w \in V - \{u, v\}$ is $\left(1 - \frac{2}{n}\right)$. Suppose, nodes u, v and w are chosen then there are three permutations possible, u, v, w and, v, w, u and, w, u, v. In all three cases, node v has degree equal to 2. The probability that node v does not participate in any of the other $T - 1$ triangles is $\left(1 - \frac{1}{n}\right)^{3T-3}$. So,

$$Pr[d[v] = 2] \approx 3\binom{T}{1}\left(\frac{1}{n}\right)\left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right)\left(1 - \frac{1}{n}\right)^{3T-3}.$$

In general, if a node v has even degree $k > 0$, then the most prominent case is when it is part of $k' = \frac{k}{2}$ case-a triangles. There could be other cases when there are fewer than $\frac{k}{2}$ triangles at node v and the reminder of the degrees come from the edges created due to case-b triangles, but the probability of this happening is quite low because the probability of multiple case-b triangles at node v is quite low due to the factor of $\left(\frac{1}{n}\right)^2$ in each case-b triangle. Hence we ignore it. Figure 7.7 show the most prominent case when $d[v] = 6$. The probability that a node v has even degree $k > 0$ is

$$Pr[d[v] = k] \approx 3^{k'}\binom{T}{k'}\left(\frac{1}{n}\right)^{k'} A \left(1 - \frac{1}{n}\right)^{3T-3k'},$$

where

$$\begin{aligned} k' &= \left\lfloor \frac{k}{2} \right\rfloor \\ A &= \left[\left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right)\ldots\left(1 - \frac{2k'}{n}\right)\right]. \end{aligned}$$

There are $\binom{T}{k'}$ ways to choose k' out of T triangles to be created at node v. Also as explained earlier there are three permutations for node v chosen for creating a case-a triangle and hence a factor of 3^k appears in the equation. The probability that node v is chosen once exactly in k' triangles is $\left(\frac{1}{n}\right)^n$ and the probability that node v does not participate in any of the other $(T - k')$ triangles is $\left(1 - \frac{1}{n}\right)^{3T-3k'}$. Factor A represents the probability that none of the triangles share an edge.

Similarly, if a node v has odd degree k there are two most prominent cases as shown in Figure 7.7:

1. When it is part of $\left\lfloor \frac{k}{2} + 1 \right\rfloor$ case-a triangles where two of these triangles share an edge. The

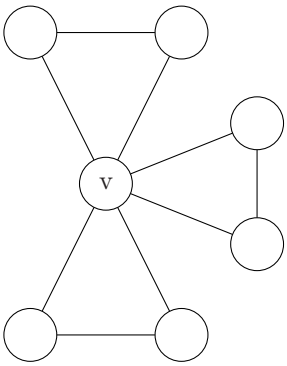

Figure 7.7: The most prominent case when node v has even degree $d[v] = 6$ for algorithm THROW-1

probability of this case is

$$E \approx 3^{k'+1}\binom{T}{k'}\left(\frac{1}{n}\right)^{k'} A\binom{T-k'}{1}\left(\frac{1}{n}\right)\left(\frac{2k'}{n}\right)\left(1-\frac{(2k'+1)}{n}\right)\left(1-\frac{1}{n}\right)^{3T-3(k'+1)},$$

where

$$\begin{aligned} k' &= \left\lfloor \frac{k}{2} \right\rfloor \\ A &= \left[\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\ldots\left(1-\frac{2k'}{n}\right)\right]. \end{aligned}$$

There are 3 permutations for node v chosen for each of the $k'+1$ case-a triangles so $3^{k'+1}$ appears in the equation. There are $\binom{T}{k'}$ ways to create k' of T non-overlapping triangles at node v and the probability that none of these k' triangles share an edge is given by factor A. There are $\binom{T-k'}{1}$ ways to create a $(k'+1)^{\text{th}}$ triangle at node v. The probability that node v is chosen in this $(k'+1)^{\text{th}}$ triangle is $\frac{1}{n}$. The probability that it shares an edge with any of the k' triangles is same as the probability that one of the $2k'$ neighbors (from k' triangles) of v is selected in $(k'+1)^{\text{th}}$ triangle which is $\left(\frac{2k'}{n}\right)$. The probability that $(k'+1)^{\text{th}}$ triangle does not share any other edge with any of the other triangles is $\left(1-\frac{(2k'+1)}{n}\right)$. $\left(1-\frac{1}{n}\right)^{3T-3(k'+1)}$ is the probability that node v is not chosen in any of the remaining $T-(k'+1)$ triangles.

2. When it is part of $\left\lfloor \frac{k}{2} \right\rfloor$ case-a triangles and one edge comes from a case-b triangle which is

$$F \approx 3^{k'}\binom{T}{k'}\left(\frac{1}{n}\right)^{k'} A \times 6\binom{T-k'}{1}\left(\frac{1}{n}\right)^{2}\left(1-\frac{(2k'+1)}{n}\right)\left(1-\frac{1}{n}\right)^{3T-3(k'+1)},$$

where

$$\begin{aligned} k' &= \left\lfloor \frac{k}{2} \right\rfloor \\ A &= \left[\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\ldots\left(1-\frac{2k'}{n}\right)\right]. \end{aligned}$$

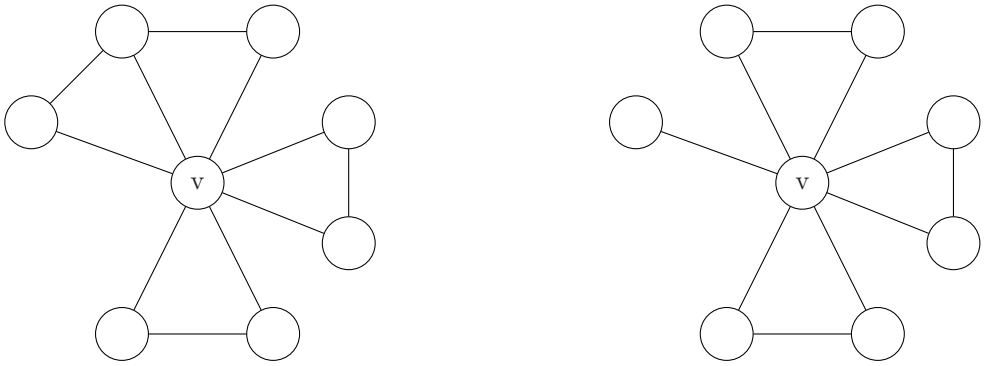

Figure 7.8: The most prominent case when a node v has odd degree $d[v] = 7$ for algorithm THROW-1

As explained earlier, $3^{k'}\binom{T}{k'}\left(\frac{1}{n}\right)^{k'} A$ is the probability that there are k' case-a triangles at node v and none of them share an edge. There are $\binom{T-k'}{1}$ ways to create a $(k'+1)^{\text{th}}$ case-b triangle at node v. The probability that node v is chosen in this $(k'+1)^{\text{th}}$ triangle is $\frac{1}{n}$. The probability that some other node $u \in V - N(v)$ is chosen in $(k'+1)^{\text{th}}$ triangle is $\left(1 - \frac{(2k'+1)}{n}\right)$ where $N(v)$ is the set of neighbors of node v. The probability that the v or u is chosen twice as required in case-b triangle is $\frac{2}{n}$. $\left(1-\frac{1}{n}\right)^{3T-3(k'+1)}$ is the probability that node v is not chosen in any of the remaining $T-(k'+1)$ triangles.

There could be many other cases but the probability of those cases is quite low and hence we ignore them.

So the probability that a node v has an odd degree k is

$$\begin{aligned} Pr[d[v]=k] &\approx E+F \\ &= 3^{k'+1}\binom{T}{k'}\binom{T-k'}{1}\left(\frac{1}{n}\right)^{k'+1}\left[\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\cdots\left(1-\frac{2k'}{n}\right)\right] \\ &\quad \left(1-\frac{(2k'+1)}{n}\right)\left(\frac{2k'}{n}+\frac{2}{n}\right)\left(1-\frac{1}{n}\right)^{3T-3(k'+1)} \\ &= 3^{k'+1}\binom{T}{k'}\binom{T-k'}{1}\left(\frac{1}{n}\right)^{k'+1}\left(1-\frac{1}{n}\right)^{3T-3(k'+1)} A\,(B+D) \end{aligned}$$

□

To calculate the expected clustering coefficient of the generated graph, we estimate the number

of triangles in a graph by counting triangles at each node using its degree and possible configurations that could produce that degree.

Theorem 7.5.2. *For algorithm* THROW-1*, if each node has triangle degree equal to* t *and single edge degree equal to* 0*, then the expected clustering coefficient of the generated graph* G *can be estimated as*

$$E[C_g(G)] \approx \frac{\sum_{k=0}^{4t} E[\triangle(k)]}{\sum_{k=0}^{4t} Pr[d[v]=k] \times \binom{k}{2}},$$

where

$$\begin{aligned}
E[\triangle(k)] &\approx 3^{k'}\binom{T}{k'}\left(\frac{1}{n}\right)^{k'}\left(1-\frac{1}{n}\right)^{3T-3k'} Ak' && \text{if } k \text{ is even}\\
&\approx 3^{k'+1}\binom{T}{k'}\binom{T-k'}{1}\left(\frac{1}{n}\right)^{k'+1}\left(1-\frac{1}{n}\right)^{3T-3(k'+1)} A\left(B(k'+1)+Dk'\right) && \text{if } k \text{ is odd}\\
k' &= \left\lfloor \frac{k}{2} \right\rfloor\\
A &= \left[\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\ldots\left(1-\frac{2k'}{n}\right)\right]\\
B &= \left(1-\frac{(2k'+1)}{n}\right)\left(\frac{2k'}{n}\right)\\
D &= \left(1-\frac{(2k'+1)}{n}\right)\left(\frac{2}{n}\right).
\end{aligned}$$

Proof. If the degree k of a node is even then the number of triangle at that node is $\frac{k}{2}$ and if the degree k of a node is odd then the number of triangles at that node can be $\lfloor \frac{k}{2} \rfloor$ or $\lfloor \frac{k}{2} \rfloor + 1$ as explained in Theorem 7.5.1.

Hence, the expected number of triangles at nodes with degree equal to k is

$$\begin{aligned}
E[\triangle(k)] &\approx 3^{k'}\binom{T}{k'}\left(\frac{1}{n}\right)^{k'}\left(1-\frac{1}{n}\right)^{3T-3k'} Ak' && \text{if } k \text{ is even}\\
&\approx 3^{k'+1}\binom{T}{k'}\binom{T-k'}{1}\left(\frac{1}{n}\right)^{k'+1}\left(1-\frac{1}{n}\right)^{3T-3(k'+1)} A\left(B(k'+1)+Dk'\right) && \text{if } k \text{ is odd,}
\end{aligned}$$

where

$$\begin{aligned} k' &= \left\lfloor \frac{k}{2} \right\rfloor \\ A &= \left[\left(1 - \frac{1}{n}\right) \left(1 - \frac{2}{n}\right) \ldots \left(1 - \frac{2k'}{n}\right) \right] \\ B &= \left(1 - \frac{(2k'+1)}{n}\right) \left(\frac{2k'}{n}\right) \\ D &= \left(1 - \frac{(2k'+1)}{n}\right) \left(\frac{2}{n}\right) \end{aligned}$$

Maximum number of triangles at nodes with degree equal to k is

$$max(\triangle(k)) = Pr[d[v] = k] \times \binom{k}{2}$$

The graph generated by this algorithm is not a regular graph with degree $2t$ but the probability that a node v has degree equal to $k > 4t$ is quite low. Hence, while calculating the expected clustering coefficient, we consider nodes with degree between 0 to $4t$. The expected value of the clustering coefficient of graph is given by

$$E[C_g(G)] \approx \frac{\sum_{k=0}^{4t} E[\triangle(k)]}{\sum_{k=0}^{4t} \max(\triangle(k))}$$

□

In the above calculation for degree and clustering coefficient, we have ignored some of the configurations and hence we expect the experimental clustering coefficient to be higher than the analytical clustering coefficient. Theorem 7.5.3 gives an analytical value of the expected clustering coefficient without ignoring any possible configuration. Hence, the values of analytical clustering coefficient are obtained using results in Theorem 7.5.3 for all experiments.

7.5.2 Throwing triangles and single edges

Now we analyze THROW-1 for the case when triangles and single edges are thrown, i.e., for all $v \in V$, $t[v] = t$, a constant and $s[v] = s$, a constant. We assume that a triple at node u, v, w is independent of a triangle at nodes u, v, and w which may not be true.

Theorem 7.5.3. *For algorithm* THROW-1*, if each node $v \in V$ has triangle degree equal to t and single edge degree equal to s, then the expected value of the clustering coefficient of the output graph is*

$$E[C_g(G)] \approx \frac{1-\left(1-\frac{6}{n^3}\right)^T+\left(1-\frac{6}{n^3}\right)^T P^3}{1-\left(1-\frac{6}{n^3}\right)^T+\left(1-\frac{6}{n^3}\right)^T P^2},$$

where

$$\begin{aligned} P &= 1-\left(1-\frac{6}{n^2}\right)^T+\left(1-\frac{6}{n^2}\right)^T\left(1-\left(1-\frac{2}{n^2}\right)^M\right) \\ T &= \frac{nt}{3} \\ M &= \frac{ns}{2}. \end{aligned}$$

Proof. Let

$$\begin{aligned} Y(u,v) &= \begin{cases} 1 & \text{if } (u,v) \in E; \\ 0 & \text{otherwise;} \end{cases} \\ X(u,v,w) &= \begin{cases} 1 & \text{if } (u,v),(v,w),(u,w) \in E; \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

By definition, the clustering coefficient is

$$C_g(G) = \frac{\sum_{distinct\ u,v,w \in V} X(u,v,w)}{\sum_{distinct\ u,v,w \in\ V} Y(u,v)Y(u,w)}.$$

Assuming that a triple at node u, v, w is independent of a triangle at nodes u, v, and w which means $Y(u,v)Y(u,w)$ is independent of $X(u,v,w)$, so the expected value of clustering coefficient is

given by

$$
\begin{aligned}
E[C_g(G)] &\approx \frac{E\left[\sum_{distinct\ u,v,w \in V} X(u,v,w)\right]}{E\left[\sum_{distinct\ u,v,w \in V} Y(u,v)Y(u,w)\right]} \\
&= \frac{\sum_{distinct\ u,v,w \in V} E[X(u,v,w)]}{\sum_{distinct\ u,v,w \in V} E[Y(u,v)Y(u,w)]} \\
&= \frac{E[X(u,v,w)]}{E[Y(u,v)Y(u,w)]},
\end{aligned}
$$

where u, v, and w are any three distinct nodes.

As each node $v \in V$ has an equal single edge degree s, the probability of choosing a node v while creating a single edge is $\frac{1}{n}$. The probability that nodes u and v are chosen while creating a single edge is $\frac{1}{n^2}$ and there are two permutations possible, u and v, and v and u. Hence, the probability that an edge is created between nodes u and v, while creating a single edge is $\frac{2}{n^2}$. Hence, the probability that an edge is not created between nodes u and v, while creating a single edge is $\left(1 - \frac{2}{n^2}\right)$. The probability that an edge is not created between nodes u and v, while creating M single edges is $\left(1 - \frac{2}{n^2}\right)^M$. So, the probability that an edge is created between nodes u and v, while creating at least one of M single edges is $1 - \left(1 - \frac{2}{n^2}\right)^M$.

Similarly, every node $v \in V$ has an equal triangle degree t, the probability of choosing a node v while creating a triangle is $\frac{1}{n}$. Similar to the probability that an edge is created between nodes u and v while creating a single edge, the probability that an edge is created between nodes u and v while creating a triangle edge is $\frac{2}{n^2}$ and it could be any of the three edges of a triangle. Hence, the probability that an edge is created between nodes u and v while creating a triangle is $\frac{6}{n^2}$. The probability that an edge is not created between nodes u and v, while creating a triangle is $\left(1 - \frac{6}{n^2}\right)$. Hence, the probability that an edge is not created between nodes u and v, while creating T triangles is $\left(1 - \frac{6}{n^2}\right)^T$ and the probability that an edge is created between nodes u and v, while creating at least one of T triangles is $1 - \left(1 - \frac{6}{n^2}\right)^T$.

$E[Y(u,v)]$ is the probability that an edge is created between nodes u and v by any of T triangles

or any of M single edges. Hence,

$$\begin{aligned} P &= E[Y(u,v)] \\ &= \left(1-\left(1-\frac{6}{n^2}\right)^T\right)+\left(1-\frac{6}{n^2}\right)^T\left(1-\left(1-\frac{2}{n^2}\right)^M\right) \end{aligned}$$

The probability that nodes u, v, and w are chosen while creating a triangle is $\frac{1}{\binom{n}{3}} \approx \frac{6}{n^3}$. Similar to the above cases, the probability that nodes u, v, and w are chosen while creating at least one of T triangles, is $1-\left(1-\frac{6}{n^3}\right)^T$ and the probability that a triangle is created between nodes u, v and w by edges created between these nodes (due to triangles and single edges) is P^3. Hence,

$$E[X(u,v,w)] = \left(1-\left(1-\frac{6}{n^3}\right)^T\right)+\left(1-\frac{6}{n^3}\right)^T P^3$$

There are edges between nodes u and v, and nodes u and w, if they were created by edges or a triangle between these three nodes. Hence,

$$E[Y(u,v)Y(u,w)] = \left(1-\left(1-\frac{6}{n^3}\right)^T\right)+\left(1-\frac{6}{n^3}\right)^T P^2$$

□

As this theorem, does not ignore any configurations, we use this theorem to calculate the values of analytical clustering coefficient for algorithm THROW-1 in experimental results.

7.6 Probabilistic Algorithm-2 THROW-2

As shown in Figure 7.9, THROW-2 first chooses three distinct nodes u, v and w with probability proportional to their triangle degree and creates a triangle among them. Here, each node v is chosen with probability $\frac{t[v]}{\sum_{x\in V} t[x]}$. This process is repeated until T triangles are created. For creating single edges, it chooses two distinct nodes u and v with probability proportional to their single edge degree and creates an edge between them. Again, each node v is chosen with probability $\frac{s[v]}{\sum_{x\in V} s[x]}$. This process is repeated until M edges are created. Similar to THROW-1, THROW-2 also just throws T triangles and M single edges and does not directly address triangle and single edge degrees of nodes.

Similar to THROW-1, the total running time of THROW-2 is also bounded by $O(|E|\log|V|+|V|)$.

THROW-2(n, s, t)

1 $\triangleright$ input: number of nodes n, single edge and triangle degree sequence arrays s and t
2 $\triangleright$ output: graph (V, E), where V is a set of nodes and E a is set of edges
3 $V \leftarrow \{1,\ 2,\ \ldots\ n\}$
4 $E \leftarrow \phi$
5 $T \leftarrow \frac{\sum_{i=1}^{n} t[i]}{3}$ $\triangleright$ number of single edges to create
6 $M \leftarrow \frac{\sum_{i=1}^{n} s[i]}{2}$ $\triangleright$ number of triangles to create
7 **while** $T > 0$
8 **do** Choose three distinct nodes
9 u with probability $\frac{t[u]}{\sum_{i=1}^{n} t[i]}$
10 v with probability $\frac{t[v]}{\sum_{i=1}^{n} t[i]}$
11 w with probability $\frac{t[w]}{\sum_{i=1}^{n} t[i]}$
12 $E \leftarrow E\ \cup\ \{(u,\ v),\ (v,\ w),\ (u,\ w)\}$
13 $T \leftarrow T - 1$
14 **while** $M > 0$
15 **do** Choose two distinct nodes
16 u with probability $\frac{s[u]}{\sum_{i=1}^{n} s[i]}$
17 v with probability $\frac{s[v]}{\sum_{i=1}^{n} s[i]}$
18 $E \leftarrow E\ \cup\ \{(u,\ v)\}$
19 $M \leftarrow M - 1$
20 **return** (V, E)

Figure 7.9: Algorithm THROW-2

7.6.1 Throwing triangles only

Let us analyze algorithm THROW-2 for the simple case when we only throw triangles, hence, for all $v \in V$, $t[v] = t$, where t is a constant, and $s[v] = 0$. Here also, this algorithm does not generate a regular graph of degree $2t$. Hence, lets us first calculate the expected degree distribution for the generated graph.

Theorem 7.6.1. *For algorithm* THROW-2*, if each node has triangle degree equal to t and single edge degree equal to 0, then the probability that node v has degree equal to k is*

$$\begin{aligned}
Pr[d[v] = k] &= \left(1 - \frac{1}{n}\right)^{3T} && \text{if } k \text{ is } 0 \\
&= 0 && \text{if } k \text{ is } 1 \\
&\approx 3^{k'} \binom{T}{k'} \left(\frac{1}{n}\right)^{k'} \left(1 - \frac{1}{n}\right)^{3T-3k'} A && \text{if } k \text{ is even} \\
&\approx 3^{k'+1} \binom{T}{k'} \binom{T-k'}{1} \left(\frac{1}{n}\right)^{k'+1} \left(1 - \frac{1}{n}\right)^{3T-3(k'+1)} \left(1 - \frac{(2k'+1)}{n}\right) \left(\frac{2k'}{n}\right) A && \text{if } k \text{ is odd,}
\end{aligned}$$

where

$$\begin{aligned}
k' &= \left\lfloor \frac{k}{2} \right\rfloor \\
A &= \left[\left(1 - \frac{1}{n}\right) \left(1 - \frac{2}{n}\right) \ldots \left(1 - \frac{2k'}{n}\right) \right].
\end{aligned}$$

Proof. In this algorithm, we choose three distinct nodes u, v and w and create a triangle among them. Here, each node v is chosen with probability $\frac{t[v]}{\sum_{x \in V} t[x]}$ but as each node have same triangle degree, each node is chosen with the same probability equal to $\frac{1}{n}$. This process is repeated $T = \frac{\sum_{v \in V} t[v]}{3} = \frac{nt}{3}$ times to create T triangles. No single edges are added. To find the expected clustering coefficient of a output graph, we first find the probability that a randomly chosen node v has degree k

As we choose three distinct nodes, a node v can have degree equal to 0, if it is not chosen in any of the T triangles. The probability that node v is not chosen while creating a triangle is $\left(1 - \frac{1}{n}\right)^3$. So, the probability that node v is not chosen in any of the T triangles and hence the probability that its degree is 0 is

$$Pr(d[v] = 0) = \left(1 - \frac{1}{n}\right)^{3T} \tag{7.4}$$

As we choose three distinct nodes, no node v can have degree equal to 1.

$$Pr(d[v] = 1) = 0 \tag{7.5}$$

As we choose three distinct nodes, a node v can have degree equal to 2, if it is part of exactly one triangle. There are $\binom{T}{1}$ ways to choose one out of T triangles to be created at node v. The probability of choosing node v is $\frac{1}{n}$. Here, as algorithm always chooses three distinct nodes to create a triangle, node v will not be chosen again in the same triangle. Suppose, nodes u, v and w are chosen then there are three permutations possible, u, v, w, v, w, u and w, u, v. In all three cases, node v have degree equal to 2. The probability that node v does not participate in any of the other $T-1$ triangles is $\left(1-\frac{1}{n}\right)^{3T-3}$. So,

$$Pr(d[v] = 2) = 3\binom{T}{1}\left(\frac{1}{n}\right)\left(1-\frac{1}{n}\right)^{3T-3} \tag{7.6}$$

In, general, if a node v has even degree $k > 0$, then the most prominent case is when it is part of $\frac{k}{2}$ triangles at node v. There are other configurations possible where triangles share some edges at node v but for simplicity we consider that none of the triangles share an edge at node v. Figure 7.10 show the most prominent case when $d[v] = 6$. The probability that a node v has even degree $k > 0$ is

$$Pr[d[v] = k] \approx 3^{k'}\binom{T}{k'}\left(\frac{1}{n}\right)^{k'} A\left(1-\frac{1}{n}\right)^{3T-3k'},$$

where

$$\begin{aligned} k' &= \left\lfloor \frac{k}{2} \right\rfloor \\ A &= \left[\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\ldots\left(1-\frac{2k'}{n}\right)\right]. \end{aligned}$$

There are $\binom{T}{k'}$ ways to choose k' out of T triangles to be created at node v. Also as explained earlier there are three permutations for node v chosen for creating a triangle and hence a factor of 3^k appears in the equation. The probability that node v is chosen in k' triangles is $\left(\frac{1}{n}\right)^n$ and the probability that node v does not participate in any of the other $(T-k')$ triangles is $\left(1-\frac{1}{n}\right)^{3T-3k'}$. Factor A represents the probability that none of the triangles share an edge.

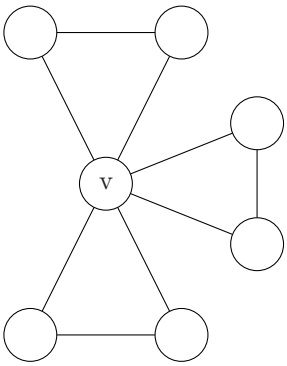

Figure 7.10: The most prominent case when node v has even degree $d[v] = 6$ for algorithm THROW-2

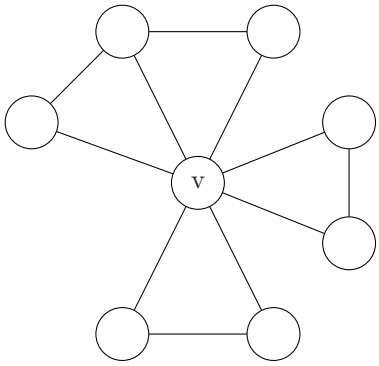

Figure 7.11: The most prominent case when a node v has odd degree $d[v] = 7$ for algorithm THROW-2

Similarly, if a node v has odd degree k, then it is part of $\left\lfloor \frac{k}{2} + 1 \right\rfloor$ triangles where exactly two of these triangles share an edge. Figure 7.11 show the most prominent case when $d[v] = 6$. Again, there are many more configuration possible but for simplicity we ignore them. The probability of this case is

$$Pr(d[v] = k) \approx 3^{k'+1} \binom{T}{k'} \left(\frac{1}{n}\right)^{k'} A \binom{T-k'}{1} \left(\frac{1}{n}\right) \left(\frac{2k'}{n}\right) \left(1 - \frac{(2k'+1)}{n}\right) \left(1 - \frac{1}{n}\right)^{3T-3(k'+1)},$$

where

$$\begin{aligned} k' &= \left\lfloor \frac{k}{2} \right\rfloor \\ A &= \left[\left(1 - \frac{1}{n}\right) \left(1 - \frac{2}{n}\right) \ldots \left(1 - \frac{2k'}{n}\right) \right]. \end{aligned}$$

There are 3 permutations for node v chosen for each of the $k' + 1$ triangles so $3^{k'+1}$ appears in the equation. There are $\binom{T}{k'}$ ways to create k' of T non-overlapping triangles at node v and the

probability that none of these k' triangles share an edge is given by factor A. There are $\binom{T-k'}{1}$ ways to create a $(k'+1)^{\text{th}}$ triangle at node v. The probability that node v is chosen in this $(k'+1)^{\text{th}}$ triangle is $\frac{1}{n}$. The probability that it shares an edge with any of the k' triangles is same as the probability that one of the $2k'$ neighbors (from k' triangles) of v is selected in $(k'+1)^{\text{th}}$ triangle which is $\left(\frac{2k'}{n}\right)$. The probability that $(k'+1)^{\text{th}}$ triangle does not share any other edge with any of the other triangles is $\left(1-\frac{(2k'+1)}{n}\right)$. $\left(1-\frac{1}{n}\right)^{3T-3(k'+1)}$ is the probability that node v is not chosen in any of the remaining $T-(k'+1)$ triangles.

□

Again we estimate the number of triangles in a graph by counting triangles at each node using its degree and possible configuration that could produced that degree.

Theorem 7.6.2. *For algorithm* THROW-2*, if each node has triangle degree equal to* t *and single edge degree equal to* 0*, then the expected clustering coefficient of the generated graph* G *can be estimated as*

$$C_g(G) \approx \frac{\sum_{k=0}^{4t} E[\triangle(k)]}{\sum_{k=0}^{4t} Pr[deg(v)=k] \times \binom{k}{2}},$$

where

$$E[\triangle(k)] \approx 3^{k'}\binom{T}{k'}\left(\frac{1}{n}\right)^{k'}\left(1-\frac{1}{n}\right)^{3T-3k'} Ak' \quad \text{if } k \text{ is even}$$

$$\approx 3^{k'+1}\binom{T}{k'}\binom{T-k'}{1}\left(\frac{1}{n}\right)^{k'+1}\left(1-\frac{1}{n}\right)^{3T-3(k'+1)}\left(1-\frac{(2k'+1)}{n}\right)\left(\frac{2k'}{n}\right)A(k'+1) \quad \text{if } k \text{ is odd,}$$

$$k' = \left\lfloor \frac{k}{2} \right\rfloor$$

$$A = \left[\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\ldots\left(1-\frac{2k'}{n}\right)\right].$$

Proof. If the degree k of node is even then the number of triangle at that node is $\frac{k}{2}$ as shown in Figure 6.11 and if the degree k of node is odd then the number of triangle at that node can be $\left\lfloor \frac{k}{2} \right\rfloor + 1$ as shown in Figure 6.12.

Expected number of triangles at nodes with degree equal to k is

$$
\begin{aligned}
E[\triangle(k)] &\approx 3^{k'}\binom{T}{k'}\left(\frac{1}{n}\right)^{k'}\left(1-\frac{1}{n}\right)^{3T-3k'} Ak' && \text{if } k \text{ is even} \\
&\approx 3^{k'+1}\binom{T}{k'}\binom{T-k'}{1}\left(\frac{1}{n}\right)^{k'+1}\left(1-\frac{1}{n}\right)^{3T-3(k'+1)}\left(1-\frac{(2k'+1)}{n}\right)\left(\frac{2k'}{n}\right)A(k'+1) \\
& && \text{if } k \text{ is odd,}
\end{aligned}
$$

where

$$
\begin{aligned}
k' &= \left\lfloor \frac{k}{2} \right\rfloor \\
A &= \left[\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\ldots\left(1-\frac{2k'}{n}\right)\right].
\end{aligned}
$$

Maximum number of triangles at nodes with degree equal to k is

$$
max(\triangle(k)) = Pr(d[v]=k)\times\binom{k}{2}
$$

The clustering coefficient of graph is given by

$$
C_g(G) \approx \frac{\sum_{k=0}^{4t} E[\triangle(k)]}{\sum_{k=0}^{4t} max(\triangle(k))}
$$

□

In above calculation for degree and clustering coefficient, we have ignored some of the configurations and hence we expect the experimental clustering coefficient to be higher than analytical clustering coefficient. Theorem 7.6.3 gives an analytical value of the expected clustering coefficient without ignoring any possible configuration. Hence, the values of analytical clustering coefficient are obtained using results in Theorem 7.6.3 for all experiments.

7.6.2 Throwing triangles and single edges

Now we analyze THROW-1 for the case when triangles and single edges are thrown i.e. for all $v \in V$, $t[v] = t = constant$ and $s[v] = s = constant$. We assume that a triple at node u, v, w is independent of a triangle at nodes u, v, and w which may not actually be true.

Theorem 7.6.3. *For algorithm* THROW-2*, if each node $v \in V$ has triangle degree equal to t and single edge degree equal to s, then the expected value of the clustering coefficient of the output graph is*

$$E[C_g(G)] \approx \frac{1-\left(1-\frac{1}{\binom{n}{3}}\right)^T+\left(1-\frac{1}{\binom{n}{3}}\right)^T P^3}{1-\left(1-\frac{1}{\binom{n}{3}}\right)^T+\left(1-\frac{1}{\binom{n}{3}}\right)^T P^2},$$

where

$$\begin{aligned} P &= 1-\left(1-\frac{3}{\binom{n}{2}}\right)^T+\left(1-\frac{3}{\binom{n}{2}}\right)^T\left(1-\left(1-\frac{1}{\binom{n}{2}}\right)^M\right) \\ T &= \frac{nt}{3} \\ M &= \frac{ns}{2}. \end{aligned}$$

Proof. Let

$$\begin{aligned} Y(u,v) &= \begin{cases} 1 & \text{if } (u,v)\in E; \\ 0 & \text{otherwise;} \end{cases} \\ X(u,v,w) &= \begin{cases} 1 & \text{if } (u,v),(v,w),(u,w)\in E; \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

By definition, the clustering coefficient is

$$C_g(G) = \frac{\sum_{distinct\ u,v,w\in V} X(u,v,w)}{\sum_{distinct\ u,v,w\in\ V} Y(u,v)Y(u,w)}.$$

Assuming that a triple at node u, v, w is independent of a triangle at nodes u, v, and w which means $Y(u,v)Y(u,w)$ is independent of $X(u,v,w)$, so the expected value of clustering coefficient is given by

$$
\begin{aligned}
E[C_g(G)] &\approx \frac{E\left[\sum_{distinct\ u,v,w \in V} X(u,v,w)\right]}{E\left[\sum_{distinct\ u,v,w \in V} Y(u,v)Y(u,w)\right]} \\
&= \frac{\sum_{distinct\ u,v,w \in V} E[X(u,v,w)]}{\sum_{distinct\ u,v,w \in V} E[Y(u,v)Y(u,w)]} \\
&= \frac{E[X(u,v,w)]}{E[Y(u,v)Y(u,w)]},
\end{aligned}
$$

where u, v, and w are any three distinct nodes.

The probability that an edge is created between nodes u and v, while creating a single edge is $\frac{1}{\binom{n}{2}}$. Hence, the probability that an edge is not created between nodes u and v, while creating a single edge is $\left(1 - \frac{1}{\binom{n}{2}}\right)$. The probability that an edge is not created between nodes u and v, while creating M single edges is $\left(1 - \frac{1}{\binom{n}{2}}\right)^M$. So, the probability that an edge is created between nodes u and v, while creating at least one of M single edges is $1 - \left(1 - \frac{1}{\binom{n}{2}}\right)^M$.

Similarly, the probability that an edge is created between nodes u and v while creating a triangle edge is $\frac{1}{\binom{n}{2}}$. It could be any of the three edges. Hence, the probability that an edge is created between nodes u and v while creating a triangle is $\frac{3}{\binom{n}{2}}$. So, the probability that an edge is not created between nodes u and v while creating a triangle is $\left(1 - \frac{3}{\binom{n}{2}}\right)$. Hence, the probability that an edge is not created between nodes u and v, while creating T triangles is $\left(1 - \frac{3}{\binom{n}{2}}\right)^T$ and the probability that an edge is created between nodes u and v, while creating at least one of T triangles is $1 - \left(1 - \frac{3}{\binom{n}{2}}\right)^T$.

$E[Y(u,v)]$ is the probability that an edge is created between nodes u and v by any of T triangles or any of M single edges. Hence,

$$\begin{aligned} P &= E[Y(u,v)] \\ &= \left(1-\left(1-\frac{3}{\binom{n}{2}}\right)^T\right)+\left(1-\frac{3}{\binom{n}{2}}\right)^T\left(1-\left(1-\frac{1}{\binom{n}{2}}\right)^M\right) \end{aligned}$$

The probability that nodes u, v, and w are chosen while creating a triangle is $\frac{1}{\binom{n}{3}} \approx \frac{6}{n^3}$. Similar to the above cases, the probability that nodes u, v and w are chosen while creating at least one of T triangles, is $1-\left(1-\frac{1}{\binom{n}{3}}\right)^T$ and the probability that a triangle is created between nodes u, v and w by edges created between these nodes (due to triangles and single edges) is P^3. Hence,

$$E[X(u,v,w)] = \left(1-\left(1-\frac{1}{\binom{n}{3}}\right)^T\right)+\left(1-\frac{1}{\binom{n}{3}}\right)^T P^3$$

There are edges between nodes u and v, and nodes u and w, if they were created by edges or a triangle between these three nodes. Hence,

$$E[Y(u,v)Y(u,w)] = \left(1-\left(1-\frac{1}{\binom{n}{3}}\right)^T\right)+\left(1-\frac{1}{\binom{n}{3}}\right)^T P^2$$

□

The degree sequence of the graphs generated by algorithms THROW-1 and THROW-2 is not regular. To find out the probability that a node v has degree equal to k, we look at the most prominent cases when node v could have degree equal to k. If we assume that each triangle contributes two edges for each node, most prominent cases are when node v with degree equal to k can be part of $x = 0$ to k' triangles. So for node v, $2x$ edges comes from triangles and $y = (k-2x)$ comes from single edges. There are many other cases when node v could have degree equal to k but the probability for those cases is quite low and hence we ignore those cases. We also tried to bound the probability for the ignored cases but we did not succeed. So, if we assume that none of the triangles and single edges share an edge then the degree distribution of the output graph can be estimated as below.

Theorem 7.6.4. *For algorithms* THROW-1 *and* THROW-2*, if each node has triangle degree equal to t and single edge degree equal to s and none of the triangles and single edges share an edge, then the probability that node v has degree equal to k is*

$$Pr[d[v] = k] = \sum_{x=0}^{k'} 3^x \binom{T}{x} 2^y \binom{M}{y} \left(\frac{1}{n}\right)^{x+y} \left(1 - \frac{1}{n}\right)^{3(T-x)+2(M-y)} AB,$$

where

$$\begin{aligned} k' &= \left\lfloor \frac{k}{2} \right\rfloor \\ y &= k - 2x \\ A &= \left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right) \ldots \left(1 - \frac{2x}{n}\right) \\ B &= \left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right) \ldots \left(1 - \frac{y}{n}\right). \end{aligned}$$

Proof. As mentioned earlier to find the probability that node v has degree equal to k, we sum the probability for the most prominent cases, which are cases when node v with degree equal to k can be part of $x = 0$ to k' triangles. So for node v, $2x$ edges comes from triangles and $y = (k - 2x)$ comes from single edges. We ignore all other cases as the probability for these cases is quite low.

There are three permutations of the three nodes chosen for triangles and two permutations for the two nodes chosen for creating shingle edges, hence the factors of 3^x and 2^y in the equation. Now x triangles and y single edges can be chosen $\binom{T}{x}$ and $\binom{M}{y}$ ways respectively. As node v is part of x triangles and all nodes have same triangle degree, the probability of choosing it x times for creating triangles is $\left(\frac{1}{n}\right)^x$. Similarly node v is also part of y single edges and all nodes have same single edge degree, hence the probability of choosing it y times for creating single edges is $\left(\frac{1}{n}\right)^y$. Factors A and B represent the probabilities that none of the triangles share an edge and none of the single edges share an edge. □

7.7 Empirical Results

We conducted experiments to measure the clustering coefficient of a graph with $n = 1000$ nodes, where each node v has single edge degree $s[v] = 0$ and triangle degree $t[v] = 1$ to 9 for CONF-1,

CONF-2, THROW-1 and THROW-2. As each triangle contributes approximately two edges for each node v that is part of it, $d[v] \approx 2t[v]$.

The anticipated, experimental, and analytical clustering coefficients are as shown in Table 7.1 and Figure 7.12, when only triangles are created without any single edges. The results are as below:

1. For moderately dense graphs (up to average degree = 14), the values of the average experimental clustering coefficient for CONF-1 and CONF-2 are approximately the same as the corresponding anticipated or target value of the clustering coefficient.

2. For denser graphs (average degree > 14), the values of the average experimental clustering coefficient for CONF-1 and CONF-2 are higher than the corresponding anticipated or target clustering coefficient. The reason behind this is the existence of extra triangles (created by edges from triangles and single edges).

3. For THROW-1 and THROW-2, the values of the average experimental clustering coefficient are the same as the corresponding analytical value, as the analytical results account for extra triangles.

4. In general, though algorithms CONF-1 and CONF-2 choose nodes with different probabilities, they behave similarly. Also though algorithm THROW-2 always chooses three distinct nodes to create triangles and two distinct nodes to create single edges, and algorithm THROW-1 may choose a node more than once for creating triangles and single edges, they also behave similarly.

For all four algorithms, the clustering coefficient of the generated graph depends on the value of input single edge degree s and triangle degree t. For a graph of $n = 1000$ nodes, where each node $v \in V$ has $t[v] = s[v]$ and $s[v] = 2t[v]$, respectively, the results are as shown in Figures 7.13 and 7.14, respectively. Again, as each triangle contributes to two edges for each node v that it is part of, the average degree is $d = 2t[v] + s[v]$. We make these observations.

1. As expected for THROW-1 and THROW-2, the values of the average experimental clustering coefficient are almost the same as the corresponding analytical value.

2. For CONF-1 and CONF-2, initially the average clustering coefficients are almost the same as the corresponding anticipated values. But as the average degree increases, the difference between

$t[v]$	anticipated	Conf-1 Exp	Conf-2 Exp	Throw-1 Exp	Throw-1 Analy Theorem 7.5.3	Throw-2 Exp	Throw-2 Analy Theorem 7.6.3
1	1	1	1	0.337171	0.335328	0.337088	0.335552
2	0.333333	0.334702	0.334586	0.204913	0.203987	0.204736	0.204151
3	0.2	0.202994	0.202657	0.14939	0.148716	0.149154	0.148844
4	0.142857	0.147716	0.147412	0.119507	0.119002	0.119427	0.119109
5	0.111111	0.117919	0.117555	0.101147	0.100805	0.101078	0.100897
6	0.090909	0.099655	0.099238	0.089053	0.0887792	0.089036	0.088862
7	0.076923	0.087603	0.087213	0.080756	0.0805125	0.080742	0.0805885
8	0.066667	0.079292	0.078914	0.074856	0.0746493	0.074846	0.0747204
9	0.058823	0.073365	0.073013	0.070585	0.0704191	0.070579	0.0704866

Table 7.1: Clustering coefficient for Conf-1, Conf-2, Throw-1 and Throw-2 where $n = 1000$ and for each node v, $s[v] = 0$

the experimental clustering coefficient and the corresponding anticipated value increases because, as the average degree increases, the graph becomes more dense and hence creates more implicit triangles.

3. For higher degree, the experimental value of the clustering coefficient for Conf-1 and Conf-2 increases much more rapidly (the ratio of the experimental and the anticipated clustering coefficient) from the corresponding anticipated or analytical clustering coefficient for the case of $s[v] = 2t[v]$ than the case when $s[v] = t[v]$. We believe this is because, when $s[v] = 2t[v]$, Conf-1 and Conf-2 tries to create fewer triangles compared to the case when $s[v] = t[v]$. But as the density of the graph (the average degree) is the same, the ratio of implicitly created triangles to anticipated triangles is more for the case of $s[v] = 2t[v]$ than that of $s[v] = t[v]$.

4. It is also interesting that, for higher average degree (greater than 15), the values of the experimental clustering coefficient for Throw-1 and Throw-2 is slightly higher than that of Conf-1 and Conf-2 (which is the reverse for smaller degrees).

As discussed earlier, the degree distribution of the graphs generated by Throw-1 and Throw-2

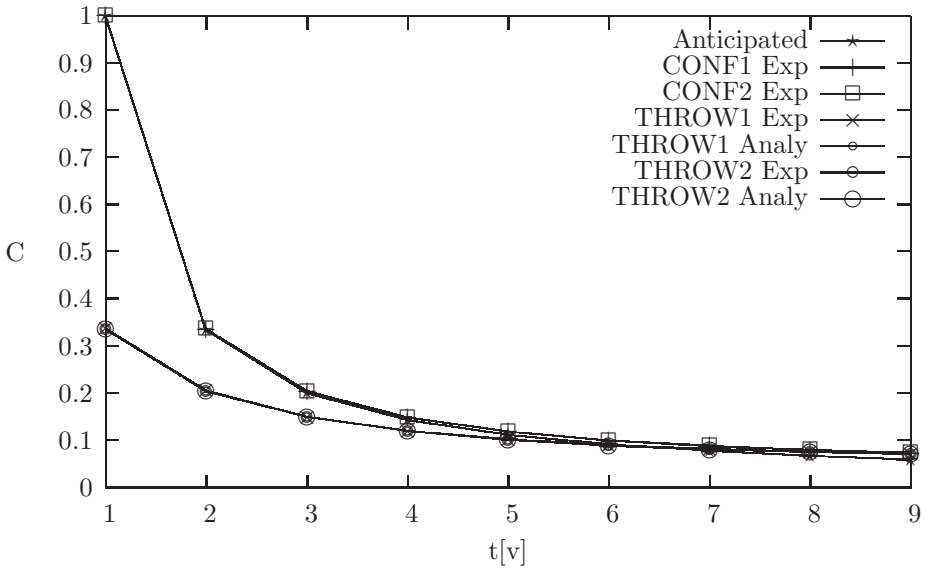

Figure 7.12: Clustering coefficient versus $t[v]$ for Conf-1, Conf-2, Throw-1, and Throw-2 where $n = 1000$ and for each node v, $s[v] = 0$.

is different from that of the input. However the approximate degree distribution can be estimated using analytical results given earlier in Theorem 7.6.4. The results for the graph with $n = 1000$ nodes, when $t[v] = 2, s[v] = 2$ and when $t[v] = 2, s[v] = 4$ for all nodes $v \in V$, is as shown in Figure 7.6.4. Here, we calculate probabilities up to degree k equal to twice the average degree $d[v] = 2t[v] + s[v]$. As the calculation considers the case when none of the triangles and single edges share an edge, it ignores some configurations and hence the analytical values are slightly smaller than the experimental values.

The experiments were performed on a 1.8 GHz, 1 GB Mac PowerPC G5. Table 7.2 shows the time (in seconds) taken by algorithms Conf-1, Conf-2, Throw-1, and Throw-2 to generate graphs with size $n = 1000$ and $n = 10000$ and, for each node v, $t[v] = 1$ and $s[v] = 1$. For a graph of size $n = 1000$, the time is the average over 1000 runs, and, for a graph of size $n = 10000$, the time is the average over 100 runs.

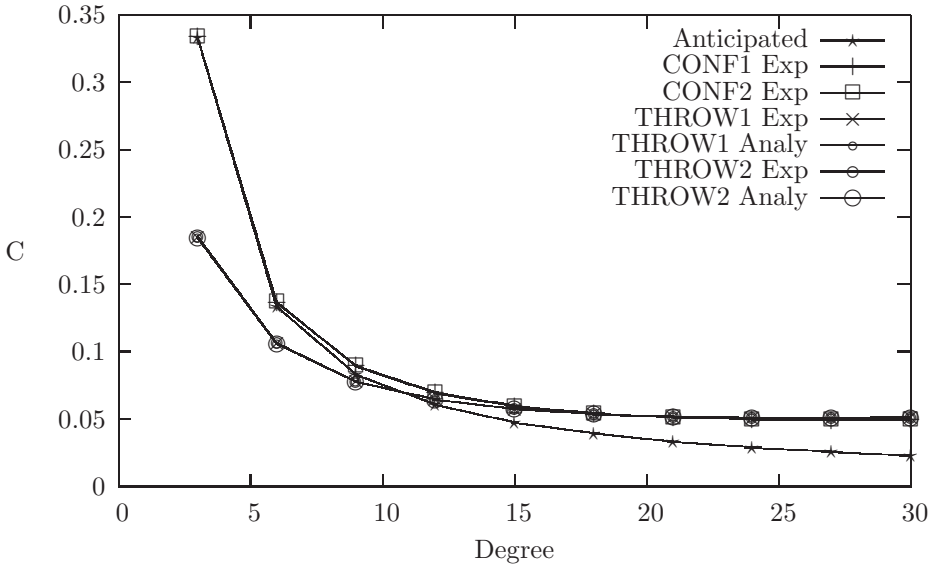

Figure 7.13: Clustering coefficient versus degree for CONF-1, CONF-2, THROW-1, and THROW-2 where $n = 1000$ and for each node $d[v] = 2t[v] + s[v]$ and $s[v] = t[v]$.

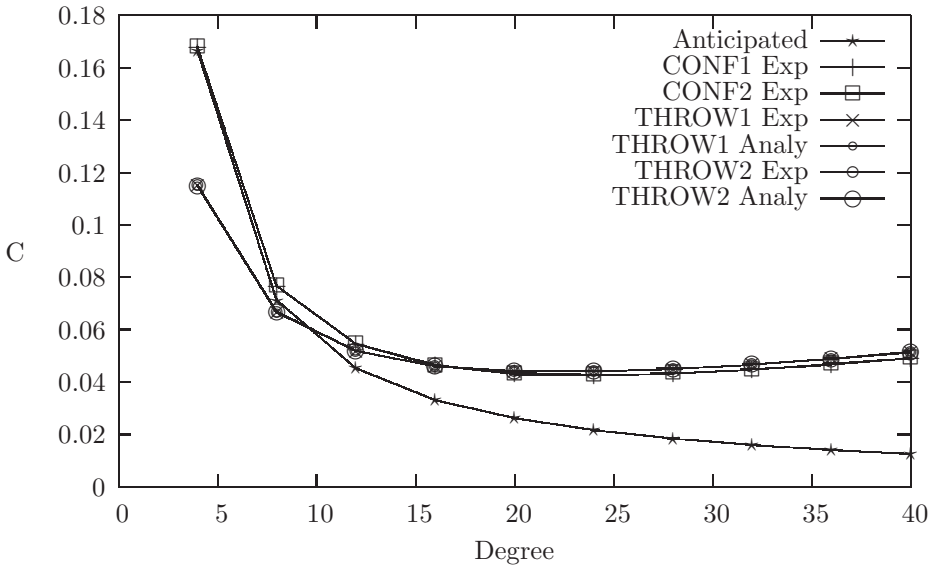

Figure 7.14: Clustering coefficient versus degree for CONF-1, CONF-2, THROW-1, and THROW-2 where $n = 1000$ and for each node $d[v] = 2t[v] + s[v]$ and $s[v] = 2t[v]$.

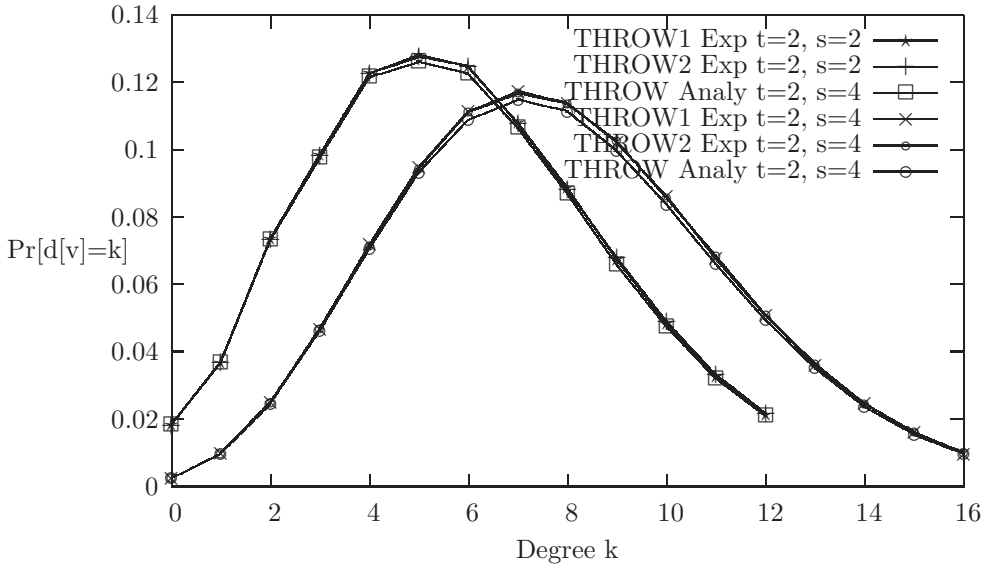

Figure 7.15: $Pr[d[v] = k]$ versus degree k for THROW-1 and THROW-2 where $n = 1000$.

Algorithm	$n = 1000$	$n = 10000$
CONF-1	0.069	3.26
CONF-2	0.067	1.23
THROW-1	0.061	0.77
THROW-2	0.062	0.79

Table 7.2: Time (in seconds) taken by algorithms CONF-1, CONF-2, THROW-1, and THROW-2 where for each node v, $t[v] = 1$ and $s[v] = 1$.

Chapter 8

Degree Sequence Algorithm DEG

8.1 Algorithm

We propose an algorithm DEG that generates a random graph with given degree sequence and clustering coefficient. It takes the number of nodes n, the degree sequence d and the clustering coefficient C as input and generates a simple random graph with these properties.

It is based on the configuration model, i.e., each node v can be seen as having a finite number of stubs coming out of it, which is equal to its degree $d[v]$. The next step is to estimate the number of triangles needed to achieve the given clustering coefficient C and then create those triangles. It is done by selecting three stubs at random and joining them to create a triangle. Once the estimated number of triangles is created, a pair of stubs are selected independently at random and joined to create additional edges.

As we aim here to generate simple graphs, we merge multiple edges into one. We also avoid any self-loops by selecting distinct nodes for creating edges and triangles.

As shown in Figure 8.1, DEG starts with n nodes and an empty edge set E. The total number of edges M needed in the graph (including those of triangles) is given by

$$M = \frac{\sum_{v=1}^{n} d[v]}{2}.$$

The number of triples at node v is $\tau(v) = \binom{d[v]}{2}$ and the total number of connected triples in the

DEG(n, d, C)

1 $\triangleright$ input: number of nodes n, degree sequence array d and target clustering coefficient C
2 $\triangleright$ output: graph (V, E), where V is set of nodes and E is set of edges
3 $V \leftarrow \{1,\ 2,\ \ldots,\ n\}$
4 $E \leftarrow \phi$
5 $M \leftarrow \frac{\sum_{x \in V} d[x]}{2}$ $\triangleright$ number of edges to create
6 $rd \leftarrow d$ $\triangleright$ residual degree array
7 $T \leftarrow \frac{C \times \sum_{i=1}^{n} \binom{d[i]}{2}}{3}$ $\triangleright$ number of triangles to create
8 **while** $T > 0$
9 **do** Choose three distinct nodes
10 u with probability $\frac{rd[u]}{\sum_{x \in V} rd[x]}$
11 v with probability $\frac{rd[v]}{\sum_{x \in V} rd[x]}$
12 w with probability $\frac{rd[w]}{\sum_{x \in V} rd[x]}$
13 Check $rd[u]$, $rd[v]$ and $rd[w]$
14 Add edges needed to create a triangle between u,v and w
15 update $rd[u]$, $rd[v]$, $rd[w]$, T and M appropriately.
16
17 **while** $M > 0$
18 **do** Choose two distinct nodes
19 u with probability $\frac{rd[u]}{\sum_{x \in V} rd[x]}$
20 v with probability $\frac{rd[v]}{\sum_{x \in V} rd[x]}$
21 **if** $(u,\ v) \notin E$
22 **then** $E \leftarrow E\ \cup\ \{(u,\ v)\}$
23 $rd[u] \leftarrow rd[u] - 1$
24 $rd[v] \leftarrow rd[v] - 1$
25 $M \leftarrow M - 1$
26 **return** (V, E)

Figure 8.1: Degree sequence algorithm DEG

graph $\tau(G) = \sum_{v \in V} \tau(v)$. Hence, the number of triangles needed to achieve input clustering coefficient C can be computed as follows

$$T = \frac{C \times \tau(G)}{3}.$$

For creating a triangle, DEG chooses three distinct nodes u, v, and w with probability proportional to their residual degree. If these three nodes have required residual degrees for edges missing from edge set E that is required to create a triangle among nodes u, v, and w, then those missing edges are created and $rd[u]$, $rd[v]$, $rd[w]$, T, and M are updated appropriately. Otherwise the node selection is ignored. For example, if $(u,\ v) \in E$ and $(u,\ w),\ (v,\ w) \notin E$ and $rd[w] \geq 2$, then the two missing edges $(u,\ w)$ and $(v,\ w)$ are added to the edge set E. Then $rd[u]$, $rd[v]$ and $rd[w]$ are decreased by 1, 1 and 2 respectively. The number of triangles to create T and the number of edges to add M are also decreased by 1 and 2, respectively.

An important point in the above step is that, when $(u,\ w),\ (v,\ w),\ (w,\ u) \in E$, no edges are added to the edge set E but the number of triangles that still need to be created T is decreased by 1 instead of an obvious action of ignoring this node selection because if we ignore the node selection, it leads to many more triangles than expected. The reason is that, when we explicitly create a triangle among nodes u, v, and w and if u and v have k common neighbors, then adding an edge $(u,\ v)$ creates k extra triangles. We call these extra triangles implicitly created triangles. We observe that, when we pick three nodes at random to create a triangle and if there already exists a triangle among them, then it is more likely that it is an implicitly created triangle than being an explicitly created triangle. So in this case we decrease the number of triangles that still needs to be created by 1 as we have encountered a triangle which we did not explicitly create.

Once the required number of triangles T is created, the next step is to create the remaining edges M. An edge is created by selecting two nodes u and v with probability proportional to their residual degree. If $(u,\ v) \notin E$, then an edge $(u,\ v)$ is added to the edge set E and $rd[u]$ and $rd[v]$ are decreased by 1. Also M, the number of edges that still need to be created, is decreased by 1.

Similar to CONF-2, the total running time of DEG is bounded by $O(|V||E|)$.

The algorithm selects nodes only if it has at least one unconnected stub. Consider nodes u, v and w such that $(u,\ v),\ (v,\ w) \in E$ and $rd[v] = 0$, $rd[u],\ rd[w] \geq 1$. Here, as the residual degree of node v is 0, it will never be selected for creating triangles. But a triangle can be created between nodes u, v and w by adding an edge $(u,\ w)$ which does not require node v to have any unconnected stub.

All such cases will be ignored by this algorithm. However, other algorithms that try to incorporate some structure in a graph do not generate all possible configurations with the same probability.

8.2 Experimental Results

We conducted the same experiments (Table 8.1) as for the four versions of Newman's algorithms. We want to generate a graph with $n = 1000$ nodes, where input degree for each node v is $d[v] = 2$ to 18 in steps of 2 and target clustering coefficient is as shown below (same as the anticipated value for CONF-1 and CONF-2 when we are only creating triangles). The results are as below:

1. For moderately dense graphs (up to average degree = 14), the value of the average experimental clustering coefficient for DEG is approximately the same as the corresponding anticipated or target value of the clustering coefficient.

2. For denser graphs (average degree > 14), the value of the average experimental clustering coefficient for DEG is higher than the corresponding anticipated or target clustering coefficient. The reason behind this is the existence of implicitly created triangles

3. The results are quite similar to that of CONF-1 and CONF-2.

Avg Degree	C Analytical	C Experimental	StdDeviation
2	1	1	0
4	0.333333	0.334601	6.978244×10^{-4}
6	0.2	0.202189	6.470735×10^{-4}
8	0.142857	0.146771	6.550684×10^{-4}
10	0.111111	0.116833	6.105926×10^{-4}
12	0.090909	0.098480	5.735758×10^{-4}
14	0.076923	0.086465	5.308188×10^{-4}
16	0.066667	0.078151	5.135681×10^{-4}
18	0.058823	0.072205	5.239204×10^{-4}

Table 8.1: Clustering Coefficient for DEG where $n = 1000$

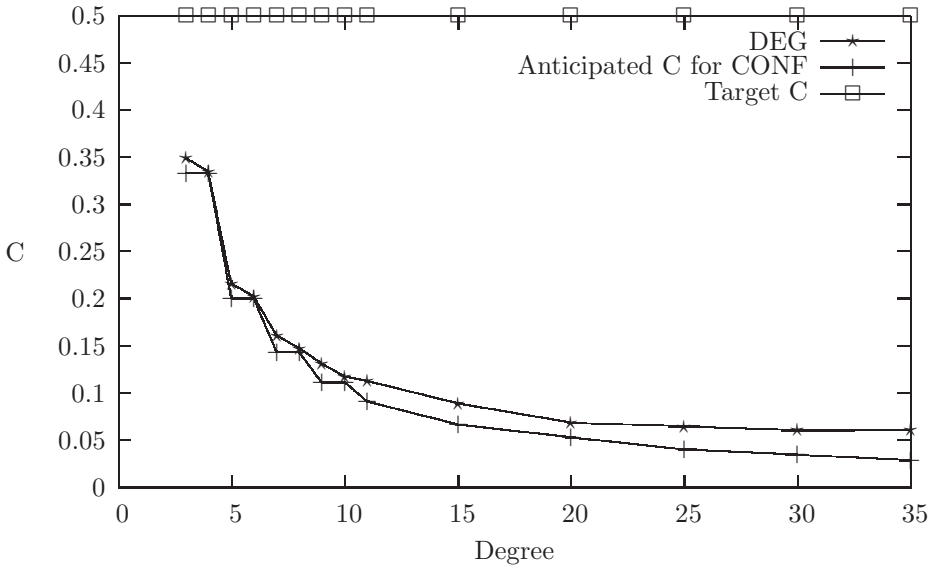

Figure 8.2: Clustering coefficient versus degree for DEG where n=1000.

To understand the combinations of degree sequence and clustering coefficient that are achievable, we conducted some experiments on DEG with varying degree. The target clustering coefficient was held constant at 0.5. Figure 8.2 shows the maximum clustering coefficient achieved. If we assume that each triangle requires two edges at each of three nodes, then the maximum possible triangles that can be produced at a node v with degree k is $\lfloor \frac{k}{2} \rfloor$ and the clustering coefficient is approximately given by

$$\frac{\sum_{v \in V} \left\lfloor \frac{k}{2} \right\rfloor}{\sum_{v \in V} \binom{k}{2}}$$

which is the same as the anticipated clustering coefficient for CONF-2 with similar input (for all nodes $v \in V$, $t[v] = \lfloor \frac{k}{2} \rfloor$ and $s[v] = k - \lfloor \frac{k}{2} \rfloor$). Figure 8.2 shows that for small degree, the clustering coefficient that could be achieved by DEG is slightly higher than the anticipated clustering coefficient for CONF-1 and CONF-2. However it is higher than the anticipated value for higher degree as the number of extra triangles (implicitly created triangles) is more as the graph becomes denser.

The experiments were performed on a 1.8 GHz, 1 GB Mac PowerPC G5. Table 8.2 shows the time (in seconds) taken by algorithm DEG to generate graphs with size $n = 1000$ and $n = 10000$

Algorithm	$n = 1000$	$n = 10000$
DEG	1.175	43.53

Table 8.2: Time (in seconds) taken by algorithm DEG where for each node v, $d[v] = 3$.

and, for each node v, $d[v] = 3$. For a graph of size $n = 1000$, the time is the average over 1000 runs, and, for a graph of size $n = 10000$, the time is the average over 100 runs.

Chapter 9

Experiments on the Real World Networks

We conducted experiments to see the clustering coefficient of the graphs generated by algorithms Conf-1, Conf-2, and Deg for a network of a symmetrized snapshot of the structure of the Internet at the level of autonomous systems [1] and a network representing the topology of the Western States Power Grid of the United States [39]. Results for the Internet and the power-grid networks are in Table 9.1.

As seen from Table 9.1, the clustering coefficient of the graph generated by Deg matches quite well with the actual clustering coefficient and the clustering coefficient of the graph generated by Conf-2 matches with the anticipated clustering coefficient for the power-grid network. Conf-1 and Conf-2 assumes that each triangle requires approximately two edges at each of its three nodes, which may not be true and hence the actual clustering coefficient for the power-grid graph is quite different from the anticipated clustering coefficient and results of Conf-1 and Conf-2. The result of Conf-1 differs from that Conf-2 as this graph is not a regular graph and Conf-1 selects each node with the same probability for creating triangles and and single-edges and hence it terminates quickly after the predefined number of trials. Algorithms Throw-1 and Throw-2 deviate from the input triangle and single-edge degree sequences and hence the result of these algorithms do not match with the actual or anticipated clustering coefficient.

For the Internet network, the result of Deg matches well with the actual clustering coefficient.

	Internet	Power-grid
n	11019	4941
$\|E\|$	31761	6594
T	88905	651
M	8951	5223
$avgD$	5.76477	2.66909
$maxD$	2359	19
$actualC$	0.03974	0.10315
$anticipatedC$	0.00029	0.05905
$CONF-1$	0.49823	0.09164
$CONF-2$	0.32030	0.06675
$THROW-1$	0.01473	0.04777
$THROW-2$	0.12813	0.05768
DEG	0.03823	0.10360

Table 9.1: Results for the graphs of the Internet and the power-grid, where n = the number of nodes, $|E|$ = the number of edges, T = the total number of triangles, M = the total number of single-edges, $avgD$ = the average degree, $maxD$ = the maximum degree.

But the result of CONF-1 and CONF-2 deviates much from the actual and anticipated clustering coefficients because it has $T = 88905$ number of triangles and $|E| = 31761$ number of edges. Here, the number of triangles are much more than the number of edges, which means many of these triangles share edges which violates the assumption of the algorithms CONF-1 and CONF-2. Here also, algorithms THROW-1 and THROW-2 deviate from the input triangle and single-edge degree sequences and hence the result of these algorithms do not match with the actual or anticipated clustering coefficient.

Chapter 10

Conclusion and Future Work

We have presented four versions of Newman's algorithm that take triangle and single edge degree sequences and generates a graph with tunable clustering coefficient and analyzed them for the case of a regular graph. The results for Conf-1 and Conf-2 are quite similar to each other, and the results of Throw-1 and Throw-2 are also similar. Conf-1 and Conf-2 generate a graph with a degree sequence and a clustering coefficient that is approximately the same as the input degree sequence (for all $v \in V$, $d[v] = 2t[v] + s[v]$) and anticipated clustering coefficient. However, as the graph becomes denser, the value of the experimental clustering coefficient is higher than the anticipated value. Throw-1 and Throw-2 do not generate a graph with the input degree sequence, but they maintain the average degree, and the analytical results give a way of predicting an exact clustering coefficient of the generated graph. The estimated degree distribution can also be calculated analytically. In the real world, it is difficult to get information about triangle and single edge degree sequence. Only information about degree sequence is available. We propose a new algorithm Deg that generates a graph with the input degree sequence and clustering coefficient. Results of this algorithm are quite similar to those of Conf-1 and Conf-2. The maximum clustering coefficient that could be achieved for the given degree sequence is approximately

$$\frac{\sum_{v \in V} \left\lfloor \frac{d[v]}{2} \right\rfloor}{\sum_{v \in V} \binom{d[v]}{2}},$$

assuming that each triangle requires two edges for each node. Hence, this formula gives a way to identify combinations of degree sequence and clustering coefficient that are nearly impossible to achieve algorithmically.

Though we have presented experimental and analytical results for regular graphs, our algorithms are generic enough to work on any degree distribution. While most of the algorithms available in the literature focus only on power-law degree distribution. In future, we would like to work on more theoretical and empirical results for any generic degree distribution. We give a naive approximation for the maximum clustering coefficient that could be achieved for given degree distribution but for dense graphs, we observe that often the value of the clustering coefficient for the generated graph is higher than the anticipated value for algorithms Conf-1, Conf-2, and Deg. In the future, we would also like to work on estimating the minimum clustering coefficient that could be achieved with the given degree sequence and modify our algorithm to match the anticipated clustering coefficient as closely as possible.

Some networks [36] represent strong community structure where density of edges is very high within a community which represent clique like structure where a few edges may be missing from a clique and these communities are connected with sparse connections. Our algorithms could be modified to incorporate various clique like structures instead of just triangles (3-clique). It would be interesting to observe the clustering coefficient of these networks as the clustering coefficient of each community is very high and hence the clustering coefficient of the resultant graph could be quite high as compared to other graphs. It may provide some intuition to model graphs with a very high clustering coefficients.

Some subgraphs occur more frequently in a network as compared to the randomized version of the same network. Such subgraphs are known as motifs in biological networks. It would also be interesting to modify and test our algorithms to generate random graphs with small subgraphs (with three or four nodes).

REFERENCES

[1] *The origin of power laws in Internet topologies revisited*, vol. 2, 2002.

[2] L. A. Adamic and B. A. Huberman, *Power-law distribution of the World Wide Web*, Science, 287 (2000), p. 2115.

[3] A. Barabasi and R. Albert, *Emergence of scaling in random networks*, Science, 286 (1999), pp. 509–512.

[4] A.-L. Barabasi, R. Albert, and H. Jeong, *Scale-free characteristics of random networks: The topology of the World-Wide Web*, Physica A: Statistical Mechanics and its Applications, 281 (2000), pp. 69–77.

[5] B. Bollobas and O. Riordan, *The diameter of a scale-free random graph*, Combinatorica, 24 (2004), pp. 5–34.

[6] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, *Graph structure in the web*, Computer Networks, 33 (2000), pp. 309–320.

[7] J. Camacho, R. Guimera, and L. Amaral, *Robust patterns in food web structure*, Physical Review Letters, 88 (2002).

[8] A. Clauset, C. R. Shalizi, and M. E. J. Newman, *Power-law distributions in empirical data*, SIAM Reviews, (2007), pp. 661–703.

[9] C. Cooper and A. Frieze, *A general model of web graphs*, Random Struct. Algorithms, 22 (2003), pp. 311–335.

[10] H. Ebel, L.-I. Mielsch, and S. Bornholdt, *Scale-free topology of e-mail networks*, Phys. Rev. E, 66 (2002), p. 035103.

[11] P. Erdős and A. Rényi, *On random graphs. I*, Publ. Math. Debrecen, 6 (1959), pp. 290–297.

[12] J. P. Gleeson, *Bond percolation on a class of clustered random networks*, Phys. Rev. E, 80 (2009), p. 036107.

[13] N. Guelzim, S. Bottani, P. Bourgine, and F. Kepes, *Topological and causal structure of the yeast transcriptional regulatory network*, Nature Genetics, 31 (2002), pp. 60–63.

[14] Q. Guo, T. Zhou, J.-G. Liu, W.-J. Bai, B.-H. Wang, and M. Zhao, *Growing scale-free small-world networks with tunable assortative coefficient*, Physica A: Statistical and Theoretical Physics, 371 (2006), pp. 814–822.

[15] W. Guo and S. B. Kraines, *A random network generator with finely tunable clustering coefficient for small-world social networks*, Computational Aspects of Social Networks, International Conference on, 0 (2009), pp. 10–17.

[16] P. Holme and B. J. Kim, *Growing scale-free networks with tunable clustering*, Phys. Rev. E, 65 (2002), p. 026107.

[17] N. M. E. J., *Random graphs with clustering*, Physical Review Letters, 103 (2009), p. 058701.

[18] H. Jeong, Z. Neda, and A. L. Barabasi, *Measuring preferential attachment for evolving networks*, Europhysics Letter 61, 61 (2003), pp. 567–572.

[19] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins, *The Web as a graph: Measurements, models, and methods*, Computing and Combinatorics: 5th Annual International Conference, COCOON'99, Tokyo, Japan, July 1999. Proceedings, (1999), pp. 1–17.

[20] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal, *Stochastic models for the web graph*, Proceedings of the 41st Annual Symposium on Foundations of Computer Science, (2000), pp. 57–65.

[21] V. Latora and M. Marchiori, *Efficient behavior of small-world networks*, Physical Review Letters, 87 (2001), p. 198701.

[22] J. Leskovec, *Graphs over time: Densification laws, shrinking diameters and possible explanations*, Knowledge Discovery in Data mining, (2005), pp. 177–187.

[23] J. Leskovec, J. Kleinberg, and C. Faloutsos, *Graph evolution: Densification and shrinking diameters*, ACM Trans. Knowl. Discov. Data, 1 (2007), p. 2.

[24] S. Milgram, *The small world problem*, Psychology Today, 2 (1967), pp. 60–67.

[25] M. Molloy, P. P. U. S. A, B. Reed, and E. Combinatoire, *A critical point for random graphs with a given degree sequence*, Random Structures and Algorithms, (1995), pp. 161–179.

[26] M. E. J. Newman, *Clustering and preferential attachment in growing networks*, Phys. Rev. E, 64 (2001), p. 025102.

[27] M. E. J. Newman, *Finding community structure in networks using the eigenvectors of matrices*, Physical Review E, (2006), p. 036104.

[28] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, *Random graphs with arbitrary degree distributions and their applications*, Physical Review, E 64 (2001), pp. 026118–1–026118–17.

[29] M. E. J. Newman and D. J. Watts, *Renormalization group analysis of the small-world network model*, Physics Letters A, 263 (1999), pp. 341–346.

[30] R. Pastor-Satorras, A. Vázquez, and A. Vespignani, *Dynamical and correlation properties of the Internet*, Physical Review Letters, 87 (2001), p. 258701.

[31] D. D. Price, *A general theory of bibliometric and other cumulative advantage processes*, Journal of the American Society for Information Science, 27 (1976), pp. 292–306.

[32] A. Rapoport and W. J. Horvath, *A study of a large sociogram*, Behavioral Science, 6 (1961), pp. 279–291.

[33] A. Reka and Barabási, *Statistical mechanics of complex networks*, Rev. Mod. Phys., 74 (2002), pp. 47–97.

[34] T. Schank and D. Wagner, *Approximating clustering coefficient and transitivity*, Journal of Graph Algorithms and Applications, 9 (2005), p. 2005.

[35] R. Solomonoff and A. Rapoport, *Connectivity of random nets*, Bulletin of Mathematical Biology, 13 (1951), pp. 107–117.

[36] A. M. Tsankov, C. R. Brown, M. C. Yu, M. Z. Win, P. A. Silver, and J. M. Casolari, *Communication between levels of transcriptional control improves robustness and adaptivity*, Mol. Syst. Biol., 2 (2006), p. 65.

[37] B. Wang, Z. Zhang, H. Tang, and Z. Xiu, *Evolving scale-free network model with tunable clustering*, International Journal of Modern Physics B, 19 (2005), p. 39513959.

[38] J. Wang, L. Rong, and L. Zhang, *Evolving small-world networks of the local world with tunable clustering*, Computing, Communication, Control and Management, ISECS International Colloquium on, 2 (2008), pp. 369–373.

[39] D. J. Watts and S. H. Strogatz, *Collective dynamics of 'small-world' networks*, Nature, 393 (1998), pp. 440–442.

[40] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner, *The structure of the nervous system of the nematode Caenorhabditis elegans*, Philosophical Transactions of the Royal Society of London. B, Biological Sciences, 314 (1986), pp. 1–340.