2006

Using The Software Adapter To Connect Legacy Simulation Models To The Rti

Deepak Kumar Rachapalli
University of Central Florida

ANALYZING THE COMMUNITY STRUCTURE OF WEB-LIKE NETWORKS: MODELS AND ALGORITHMS

by

AUREL CAMI
B.S. University of Tirana, 1995;
B.S. Middle East Technical University, 1999

Major Professor: Narsingh Deo

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2005

ABSTRACT

This dissertation investigates the *community* structure of *web-like* networks (*i.e.,* large, random, real-life networks such as the World Wide Web and the Internet). Recently, it has been shown that many such networks have a *locally dense and globally sparse* structure with certain small, dense subgraphs occurring much more frequently than they do in the classical Erdös-Rényi random graphs. This peculiarity—which is commonly referred to as community structure—has been observed in seemingly unrelated networks such as the Web, email networks, citation networks, biological networks, *etc*. The pervasiveness of this phenomenon has led many researchers to believe that such cohesive groups of nodes might represent meaningful entities. For example, in the Web such tightly-knit groups of nodes might represent pages with a common topic, geographical location, *etc*., while in the neural networks they might represent evolved computational units.

The notion of *community* has emerged in an effort to formalize the empirical observation of the locally dense globally sparse topology of web-like networks. In the broadest sense, a community in a web-like network is defined as a group of nodes that induces a dense subgraph which is sparsely linked with the rest of the network. Due to a wide array of envisioned applications, ranging from crawlers and search engines to network security and network compression, there has recently been a widespread interest in finding efficient community-mining algorithms.

In this dissertation, the community structure of web-like networks is investigated by a combination of analytical and computational techniques: First, we consider the problem of *modeling* the web-like networks. In the recent years, many new random graph models have been

proposed to account for some recently discovered properties of web-like networks that distinguish them from the classical random graphs. The vast majority of these random graph models take into account only the *addition* of new nodes and edges. Yet, several empirical observations indicate that *deletion* of nodes and edges occurs frequently in web-like networks. Inspired by such observations, we propose and analyze two dynamic random graph models that combine node and edge addition with a *uniform* and a *preferential* deletion of nodes, respectively. In both cases, we find that the random graphs generated by such models follow power-law degree distributions (in agreement with the degree distribution of many web-like networks).

Second, we develop a framework for evaluating the degree to which the fundamental nature of communities is captured by some relevant graph theoretic concepts. This framework consists in estimating the concentration in web-like networks of a subgraph proposed as definition of community using sampling techniques and then deducing the statistical significance of such concentration by contrasting with appropriately defined random graphs. We apply this methodology to investigate the suitability in defining community of two graph concepts—alliances and near-cliques—and we also analyze the computational complexity of various community-mining problems under these definitions of community. Assuming the definition of community as a global defensive alliance, or a global offensive alliance we prove—using transformations from the *dominating set* problem—that finding optimal communities is an NP-complete problem.

These and other similar complexity results coupled with the fact that many web-like networks are huge, indicate that it is unlikely that fast, exact sequential algorithms for mining communities may be found. To handle this difficulty we adopt an *algorithmic* definition of

community and a simpler version of the community-mining problem, namely: *find the largest community to which a given set of seed nodes belong*. We propose several greedy algorithms for this problem: The first proposed algorithm starts out with a set of seed nodes—the initial community—and then repeatedly selects some nodes from community's neighborhood and pulls them in the community. In each step, the algorithm uses *clustering coefficient*—a parameter that measures the fraction of the neighbors of a node that are neighbors themselves—to decide which nodes from the neighborhood should be pulled in the community. This algorithm has time complexity of order $O(nd_{\max}^2)$, where n denotes the number of nodes visited by the algorithm and $d_{\max}$ is the maximum degree encountered. Thus, assuming a power-law degree distribution this algorithm is expected to run in near-linear time. The proposed algorithm achieved good accuracy when tested on some real and computer-generated networks: The fraction of community nodes classified correctly is generally above 80% and often above 90% .

A second algorithm based on a *generalized* clustering coefficient, where not only the first neighborhood is taken into account but also the second, the third, *etc.,* is also proposed. This algorithm achieves a better accuracy than the first one but also runs slower. Finally, a *randomized* version of the second algorithm which improves the time complexity without affecting the accuracy significantly, is proposed.

The main target application of the proposed algorithms is focused crawling—the selective search for web pages that are relevant to a pre-defined topic.

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Narsingh Deo, for introducing me to the beautiful area of random graphs, and for his continued mentorship and encouragement. Without his support this dissertation would not have been possible. I would also like to thank the members of my research committee, professors Ronald Dutton, Charles Hughes, Sheau-Dong Lang and Gary Richardson for their help and advice during the process of writing the manuscript.

I wish to thank my fellow graduate students in the Center for Parallel Computation, especially Hemant Balakrishnan, for many stimulating discussions and insightful comments. I am indebted to the faculty and the staff of the Computer Science Department for providing the environment and the resources which made this effort possible.

Special thanks are due to my wife Besa for her love and support, my parents Bilbil and Re for their faith and inspiration, and my daughter Iris whose smile makes everything worthwhile.

TABLE OF CONTENTS

LIST OF FIGURES viii

LIST OF TABLES x

LIST OF SYMBOLS xi

1. INTRODUCTION 1

1.1. Terminology and Basic Definitions 6

1.2. The Concept of Random Graph 10

1.3. Some Experimental Studies of Web-like Networks 13

2. RANDOM GRAPH MODELS 17

2.1. Static Random Graphs 20

2.2. Dynamic Random Graphs 29

2.3. Some Techniques for Analyzing Dynamic Random Graphs 42

3. PROPOSED BIRTH-DEATH DYNAMIC RANDOM GRAPH MODEL 58

3.1. Number of Nodes 60

3.2. Number of Edges 62

3.3. Degree Distribution in the First Neighborhood of the Deleted Node 64

3.4. Degree Distribution 66

4. THE NOTION OF COMMUNITY 70

4.1. Some Graph-Theoretic Problems Related to Community-Mining 70

4.2. Graph-theoretic Definitions of Community 72

4.3. Computational Complexity of Community Mining 76

5. COUNTING COMMUNITIES IN WEB-LIKE NETWORKS 85

5.1. Subgraph Counting in Dynamic Random Graph Models 85
5.2. Counting Communities by Trawling 92
5.3. Estimating the Density of Communities by Sampling 94
6. EXISTING ALGORITHMS FOR COMMUNITY MINING 107
6.1. Algorithms Based on Hierarchical Clustering 107
6.2. Algorithms Based on Spectral Analysis 110
6.3. Algorithms based on Flows 115
6.4. Other community-mining algorithms 120
7. PROPOSED ALGORITHMS FOR COMMUNITY MINING 122
7.1. Description of the Algorithms 122
7.2. Experimental Results 126
8. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS 134
APPENDIX: WEB RESOURCES AND SOFTWARE TOOLS 137
LIST OF REFERENCES 142

LIST OF FIGURES

Figure 1.1: The elements of the binomial probability space $G_{3,0.4}$ 11

Figure 2.1. The elements of the uniform probability space $\mathcal{G}_{4,5}$. 20

Figure 2.2. A small-world random graph with $n = 8$, $k = 2$ and (a) $p = 0$; (b) $p = 0.5$; (c) $p = 1$. 23

Figure 2.3. (a) Enumeration of the elements of the space $\mathcal{G}_{4,1,1}$; (b) The single graph that belongs to the space $\mathcal{G}_{4,4}$ but not to the space $\mathcal{G}_{4,1,1}$. 24

Figure 2.4. The two simple graphs on 6 nodes having degree sequence $\{2,2,1,1,2,2\}$. 26

Figure 2.5. A realization of the LCD model. 33

Figure 2.6. Two realizations of KR-1 model: (a) $\alpha = 0.4$; (b) $\alpha = 3$. 34

Figure 2.7. Two realizations of the DM-2 model: (a) $\alpha = 0.5$; (b) $\alpha = 4$. 35

Figure 2.8. A realization of BB model. 36

Figure 2.9. A realization of the KR-2 model. 37

Figure 2.10. Evolution of degree in the BA model. 45

Figure 2.11. Log-log plot of the degree distribution in the BA model. 46

Figure 2.12. Log-log plot of the degree distribution in the BB model. 48

Figure 2.13. Log-log plot of the degree distribution in the KR-1 model with $\alpha = 0.2$. 56

Figure 3.1. A small graph illustrating the probability distribution used in the preferential deletion model. 59

Figure 3.2. Growth in the number of nodes of graph G_t with the number of steps t, for three different values of the birth probability p. 61

Figure 3.3. Growth in the number of edges of graph G_t with the number of steps t, for three different values of the birth probability p. 64

Figure 3.4. The expected number of neighbors of degree k of a node chosen for deletion. 65

Figure 3.5. Log-log plot of the cumulative degree distribution of the graph generated by the preferential deletion model. 69

Figure 4.1. Construction of an instance of GDA from an instance of DS. 80

Figure 4.2. A graph G', a global defensive alliance S' in G' (nodes surrounded by squares) and a *non-component* node (surrounded by a circle) that has only one neighbor in S' which is a *component* node. 82

Figure 7.1. A graph consisting of two complete graphs K_{10} plus an edge that links them together: (a) initial configuration (b) after one step of the algorithm FIND-COMMUNITY. 126

Figure 7.2. Discovering the two communities of Zachary's karate club network: (a) Node 1 (white) is chosen as the seed for the first community; (b) The white nodes represent the community found by the algorithm; (c) Node 34 (white) is chosen as the seed for the second community; (d) The white nodes represent the community found by the algorithm. 127

Figure 7.3. Fraction of correctly classified nodes versus the ratio d_{in}/d_{out} 130

Figure 7.4. Fraction of correctly classified nodes versus the relative size of the set of seed nodes. 131

Figure 7.5. Robustness of the algorithm to different sets of seed nodes. 131

LIST OF TABLES

Table 1.1: Some parameters of selected real-world networks 16

Table 5.1: Number of feasible permutations for the best-, average-, and worst-case sample inputs 100

Table 5.2: Some parameters of the FOLDOC network 102

LIST OF SYMBOLS

Graphs and random graphs

$C(u)$	The clustering coefficient of node u
$C(G)$	The clustering coefficient of graph G
C_i	A cycle of length i
$C_{i,j}$	A bipartite core
$d(u), d_u$	The degree of node u
$d^-(u),\ d^+(u)$	The in- and out-degree of node u
$d_t(u)$	The degree of node u in the graph G_t
$d_t^-(u), d_t^+(u)$	The in-degree and out-degree of node u in the graph G_t
$d(u,w)$	The distance between nodes u and w
$\mathrm{dist}(G)$	The average distance of graph G
$\mathrm{diam}(G)$	The diameter of graph G
$\overline{d}(G)$	The average degree of graph G
$\overline{d}_t$	The average degree of graph G_t
$\overline{d}_t^{(1)}$	The average degree of a random neighbor of a random node
$D(V,A)$	A directed graph with set of nodes V and set of arcs A
$\mathcal{G}_{n,p}$	The probability space of the binomial random graph model
$\mathcal{G}_{n,m}$	The probability space of the uniform random graph model
$\mathcal{G}_{n,k,p}$	The probability space of the small-world random graph model
$\mathcal{G}_t$	The t^{th} probability space of a dynamic random graph model
$G(V,E)$	An undirected graph with set of nodes V and set of edges E
$G_{n,p}$	A binomial random graph; random element of the space $\mathcal{G}_{n,p}$
$G_{n,m}$	A uniform random graph; random element of the space $\mathcal{G}_{n,m}$

$G_{n,c,d_{in},d_{out}}$	A random graph with known community structure
$G_t = (V_t, E_t)$	The t^{th} random graph in a dynamic random graph model
K_n	The complete graph
$K_{i,j}$	The complete bipartite graph
m	The number of edges (size) of a graph
m_t	The number of edges of graph G_t
M	The maximum size of a simple graph, *i.e.*, $M = n(n-1)/2$
n	The number of nodes (order) of a graph
n_t	The number of nodes of graph G_t
$N_{t,k}$	The number of nodes of degree k in the graph G_t
$N_{t,k}^{(1)}$	The number of degree k neighbors of the node chosen for deletion in step t
$N_{t,k}^{-}, N_{t,k}^{+}$	The number of nodes of in-degree (out-degree) k in the graph G_t
N_{t,C_i}	The number of cycles of length i in the graph G_t
N_{t,P_i}	The number of paths of length i in the graph G_t
$N_{t,K_{i,j}}$	The number of complete bipartite subgraphs $K_{i,j}$ in the graph G_t
P_i	A path of length i
$r(u,w)$	The distance in a metric space between two points u, w
$R_t(w)$	A quantity used in the LCD random graph model
t	Variable denoting the time-step in dynamic random graphs
$T(G)$	The transitivity of graph G
$\Omega_{\mathcal{D}}$	The set of graphs with degree sequence $\mathcal{D}$
$[V]^2$	The set of all potential edges of a simple graph
W_n	The wheel graph on n nodes

Probability

a.a.s.	Asymptotically almost surely

a.e.	Almost every
$B(p)$	Bernoulli distribution
$Bi(n,p)$	Binomial distribution
$N(\mu,\sigma^2)$	Normal (Gaussian) distribution
pdf	Probability distribution function of a random variable
$\mathbb{E}(X)$	The expectation of random variable X
$\mathbb{I}_A$	The indicator function of event A
$\mathbb{P}_t(u)$	The probability of selecting node u at time t
$\mathbb{P}(A)$	The probability of event A
$\mathbb{P}(k)$	The (asymptotic) pdf of a parameter of a random graph
$\mathbb{P}_t(k)$	The degree distribution of a parameter of the graph G_t
$\mathbb{P}_t(w,k)$	The distribution of the degree of node w in the graph G_t
r	The Pearson correlation coefficient

Parameters of random graph models and special symbols

α	Exponent of the degree in the KR-1 model
γ	The exponent of a power-law distribution *i.e.,* $\mathbb{P}(k) \sim k^{-\gamma}$
Γ	The gamma function; $\Gamma(n) = (n-1)!$
$\delta(k-a) = \delta_{ka}$	The Kronecker delta function
∂	Partial derivative
ε	The number of new edges added in each step
λ	The initial attractiveness in the DM-1 model
μ	The initial attractiveness for outgoing arcs
η_s	The fitness of node s in the BB model
p,q	Parameters denoting probability in some random graph models

Asymptotics

$a_n = O(b_n)$	Big O
$a_n = \Theta(b_n)$	Same order of magnitude
$a_n \sim b_n$	Asymptotic equality

1. INTRODUCTION

With the dramatic growth of the World Wide Web (Web) and the Internet, the study of large, random networks has acquired new prominence. Recent empirical studies have shown statistical similarities between these two and other complex, real-life networks such as the network of phone calls, power-distribution networks, citation network, science-collaboration network, movie-actor collaboration network, the network of sexual contacts, neural networks, and various infrastructure networks [AB02, New03b, BFT01]. The term *web-like* is used in this dissertation to refer to the real-life networks cited above and others that are statistically similar. This term is preferred over the most widely used term *scale-free* because that the emphasis in this dissertation is not on any scale-free property of web-like networks. Viewed as large, random graphs in which birth and death of nodes and links are taking place, they differ from the classical Erdös-Rényi (ER) random graphs [ER59, ER60] in significant ways. Most notably, for many web-like networks the proportion of nodes with degree k decreases as $k^{-\gamma}$ (*i.e.*, as a scale-free power-law) while in the random graph $G_{n,p}$ this proportion follows approximately a Poisson distribution. Moreover, web-like networks exhibit a significantly greater degree of clustering than $G_{n,p}$.

This dissertation is concerned with another recently discovered characteristic of web-like networks. This characteristic—known as the *community* structure—pertains to the fact that certain small, dense subgraphs occur in web-like networks much more frequently than they do in the ER random graphs. Such dense subgraphs have been found in seemingly unrelated networks such as the Web [KRRT99, DKMR01], email networks [GDDG03], citation networks [ADDG04], biological networks [GN02, MSIK02], *etc*. The notion of *community* has emerged in

an effort to formalize these empirical findings. In the broadest sense, a community in a web-like network has been defined as a group of nodes that induces a dense subgraph which is sparsely linked with the rest of the network. A community has also been defined in graph-theoretic terms (*e.g.*, *complete bipartite subgraph* [KRRT99], or *defensive alliance* [FLG00, HHH03, RCCL04]) as well as algorithmically (*e.g.*, the *hubs-and-authorities* communities produced by the HITS algorithm [Kle99]).

The pervasiveness of community structure in web-like networks, has led researchers to believe that such cohesive groups of nodes might represent meaningful entities. For example, in the Web such tightly-knit groups of nodes might represent pages with a common topic, geographical location, *etc*., while in the neural networks they might represent evolved computational units.

Currently, there is a widespread interest in finding efficient algorithms for mining communities. This interest stems from a wide array of envisioned applications for such algorithms, as outlined next:

a) Web applications: The ongoing rapid and apparently chaotic growth of the Web has posed unprecedented scaling and algorithmic challenges for Web-related applications such as crawlers and search engines [DG03, Hen03]. Two prominent such challenges are:

- *Increasing the coverage and maintaining the currency of search engine indices*: It has been known for some time that search engines cover only a fraction of the Web. For instance, in 2000, no search engine covered more than 16% of the Web and the top 11 search engines combined covered about 50% of the Web [LG00]. Exhaustive crawling is, in fact, becoming increasingly unattainable due to the huge size and dynamic content of the Web [CBD99, DG01b, Hen03]. To address such issues, focused (or,

topical) crawlers [CBD99, DCLG00, MPS04] have been proposed as an alternative to general-purpose crawlers. Guided by community-mining algorithms, such crawlers would selectively seek out pages that are relevant to a pre-defined topic, thereby improving the coverage and the currency of the indices.

- *Reducing the number and increasing the relevance of hyperlinks returned to a user query*: A user searching the Web can be overwhelmed by the large number of results returned by a search engine. In addition, queries are often prone to ambiguity: some of the returned results are completely unrelated. The *PageRank* [PBMW98] algorithm took a first step to remedy these issues by assigning a prestige value to each web site and sorting the responses by the prestige value before returning them. Obviously, more needs to be done. For instance, if the search engine could group the responses along the lines of different topics[1], then the user could quickly jump to the desired specific topic. This application calls for algorithms to *cluster* (a subgraph of) the Web into communities.

b) Network security: The design of algorithms for quarantining (containing) the propagation of cyber attacks has become an important research area for network security. Quarantining techniques, often consist in deploying software agents which can block the propagation of malicious code [DN03]. A major challenge is to select a subset of nodes in the network where these software agents may be deployed in order to maximize the efficiency of quarantining. Characterization of the community structure of a cyber-graph may be utilized to design efficient quarantining strategies, for example by deploying the software agents in the sparse regions of the cyber-graph.

---

[1] A search-engine which does that can be found at *http://clusty.com*.

c) Network compression: Due to the massiveness of many important real-life networks, the compression of networks has emerged as an important research problem [DL98, AM01, RG03, LDC04]. Recent compression techniques for the Web graph take into consideration some of the recently discovered properties of this network, including the community structure. For example, Raghavan and Garcia-Molina [RG03] have proposed a two-level scheme for representing the Web graph consisting of: (i) a set of lower-level graphs, each of which encodes the interconnections within a small subset of pages; and (ii) a top-level directed graph, consisting of "super-nodes" and "super-edges". The grouping of web pages into super-nodes is guided by some empirical observations in the Web, such as *domain locality*—tendency of web pages to point more to other pages in the same domain—and the high probability that web pages with similar adjacency lists are topically related. Extending these ideas, one could argue that a better characterization of the community structure in the Web could lead to a more efficient method to group pages into super-nodes which, in turn, could improve the compression technique.

Besides the above applications of community-mining, there are numerous others such as automatic re-population of topic taxonomies with newer and more relevant pages [CDAR98], Web-filtering (*e.g.,* identification of hate or pornographic websites) [DVGB03], selective advertising [RC02], assisting search engines in handling Web spamming [FMN04], *etc*.

In this dissertation, the community structure in web-like networks is investigated by a combination of analytical and computational techniques:

First, we consider the problem of modeling the web-like networks. In the recent years, many new random graph models have been proposed to account for the newly-discovered properties of such networks [DC05d]. The vast majority of these models take into account only the *addition* of new nodes and edges. Yet, several empirical observations indicate that the *deletion* of nodes and

edges occurs frequently in web-like networks [VPV02, BBKT04]. Inspired by such observations we propose and analyze two dynamic random graph models that combine node and edge addition with a *uniform* and a *preferential* deletion of nodes, respectively [DC05a, DC05c]. In both cases, we find that the random graphs generated by the proposed models follow power-law degree distributions (in agreement with the degree distribution of many web-like networks).

Second, we analyze the expected number of certain small subgraphs—such as defensive alliances on three and four nodes—in various random graphs models. Our findings show that while in the binomial random graph $G_{n,p}$ the expected density of such subgraphs is very close to zero, in some new *dynamic* random graph models it is much larger. These findings converge with the results we have obtained by computing the number of communities in some crawls of the Web [BCD05], via sampling.

Next, we investigate the computational complexity of community mining under various definitions of community. Assuming the definition of community as a global defensive alliance, or global offensive alliance [KHH04] we prove—using transformations from the *dominating set* problem [GJ79]—that finding optimal communities is an NP-complete problem.

These complexity results and similar ones obtained by other authors [FTT04], coupled with the fact that many web-like networks are huge, indicate that it is unlikely that fast, exact sequential algorithms for mining communities may be found. To handle this difficulty we adopt an *algorithmic* definition of community and a simpler version of the community-mining problem, namely: *find the largest community to which a given set of seed nodes belong*. We propose several greedy algorithms for this problem [DC05b]. The first proposed algorithm starts out with a set of seed nodes, and then repeatedly selects some nodes from community's neighborhood and places them in the community. In each step, this algorithm uses *clustering*

coefficient [WS98]—a parameter that measures the fraction of the neighbors of a node that are neighbors themselves—to decide which nodes from the neighborhood should be pulled in the community. This algorithm has time complexity of order $O(nd_{\max}^2)$, where n is the number of nodes visited by the algorithm and $d_{\max}$ is the maximum degree encountered. Thus, assuming a power-law degree distribution this algorithm is expected to run in near-linear time. This conclusion is supported by our timing results. The proposed algorithm achieved good accuracy when tested on some real and computer-generated networks: the fraction of community nodes classified correctly is generally above 80% and often above 90% .

A second algorithm based on a *generalized* clustering coefficient, where not only the first neighborhood is taken into account but also the second, the third, *etc.,* is also proposed. This algorithm achieves a better accuracy than the first one but also runs slower. Finally, a *randomized* version of the second algorithm which improves the time complexity without affecting the accuracy significantly, is proposed.

1.1. Terminology and Basic Definitions

This dissertation adheres to the standard terminology of graph theory (*e.g.,* [Deo74, Die00]). A description of all the symbols used in this dissertation is provided in the List of Symbols and some graph-theoretic concepts which have received substantial attention recently, are briefly discussed next.

Degree distribution and degree correlation: The *degree distribution* of a graph is the probability distribution function of the degree of a node chosen uniformly at random. The symbol γ is used to denote the exponent of a power-law distribution—which arises frequently in

web-like networks. *Degree correlation* is a measure of the mixing patterns according to degree, *i.e.*, it indicates whether high-degree nodes are linked more often to other high-degree nodes or to small-degree ones. Borrowing terminology from sociology and ecology, the networks where the former case is true have been called *assortative*, while the networks where the later case is true have been called *disassortative* [New03b]. Newman [New03a] has proposed using the Pearson correlation coefficient r of the degrees at either end of a randomly chosen edge, to quantify the degree correlation of a network. This number would be positive for assortative networks and negative for disassortative ones.

Clustering coefficient: This parameter, which was first introduced by Watts and Strogatz [WS98], measures the fraction of the neighbors of a node that are neighbors themselves. Clustering coefficient has attracted substantial attention recently, in part due to the surprising discovery that in many web-like networks the value of this parameter is much higher than in the classical random graphs. The clustering coefficient of a node u is given by

$$C(u) = \frac{\text{no. of edges between the neighbors of } u}{d_u(d_u - 1)/2}$$

Note that this definition is not valid for nodes with degree less than two; the clustering coefficient of such nodes is usually taken to be zero.

Two different approaches have been proposed to extend the definition of clustering coefficient to the whole graph: The first approach proposes a parameter called the *clustering coefficient* of graph G and denoted by $C(G)$ which is given by

$$C(G) = \frac{\sum_{v \in V} C(v)}{n}$$

The second approach proposes a parameter known as *transitivity* of the graph G and denoted by $T(G)$. This parameter was first proposed by Barrat and Weight [BW00] as an easier-

to-compute approximation of the clustering coefficient of a small-world network [WS98] and is defined as

$$T(G) = \frac{3 \times \text{number of triangles}}{\text{number of paths of length two}}$$

The factor of three in the numerator ensures that T lies in the range $[0, 1]$. Note that the definition of transitivity $T(G)$, unlike that of clustering coefficient $C(G)$, does not exclude the nodes with degree less than two.

Small-world property: The *distance* $d(u, v)$ between two nodes u and v is the length of the shortest path between them. If such a path does not exist, then $d(u, v)$ is taken to be ∞. The *average distance* of a graph is given by

$$\text{dist}(G) = \frac{\sum_{u,v \in V} d(u, v)}{n(n-1)/2}$$

To avoid an infinite value, the average distance of a disconnected graph may be computed by first finding the average distance of each connected component separately and then taking the average of these values. The *diameter* of a graph is defined as

$$\text{diam}(G) = \max_{u,v \in V} \{d(u, v)\}$$

For a disconnected graph, the diameter may also be defined as the maximum of the diameters of its connected components. A graph is said to satisfy the *small-world* property if its diameter is of order $O(\log n)$ [WS98].

Connectedness and the giant component: An undirected graph is said to be *connected* if there is a path between every pair of nodes. A graph that is disconnected may be partitioned into *connected components* which are connected, pairwise-disjoint subgraphs. A graph is said to have a *giant component* [JKLP93] if it contains a connected component of size εn, for some $\varepsilon > 0$,

while all other components have size of order $O(\log n)$. A graph is said to be *k-connected*, if every pair of nodes can be connected by at least k edge-disjoint paths. A directed graph is said to be *strongly connected* if for every pair of nodes u, v there is a directed path from u to v and another one from v to u.

Robustness: In order to function properly, many real networks such as the Internet or the energy power grids, must be connected. Hence, in many practical cases it is important to have a measure of the fraction of nodes of a connected network that must be removed in order to break the network into two or more components. This fraction has been called the *robustness* (or, resilience) of a network. The nodes may be removed randomly or based on some strategy. The former case corresponds to random network failures whereas the latter corresponds to failure due to malicious attacks.

In addition to the above parameters that have been studied widely in the context of web-like networks, a few others have also received some attention: The *spectra* of a graph are the eigenvalues of its adjacency matrix; they can provide important information about the structure of the graph. The *betweenness centrality* or *load* of a node is defined as the number of shortest paths passing through that node; the *betweenness* of an edge is defined analogously.

Next, we provide a brief introduction to the notion of random graph, which plays a central role in this dissertation.

1.2. The Concept of Random Graph

A random graph model may be specified in two ways: (i) through an algorithmic definition; and (ii) through a formal, mathematical definition. These two methods are illustrated next by taking as an example the classical random graph model $G_{n,p}$.

The first method consists of providing an algorithm (or, procedure) that generates an instance of the random graph $G_{n,p}$. This procedure is defined as follows: First, enumerate the two-element subsets of the set $[n]$ as $1,\ldots,M$ where $M = n(n-1)/2$. Then, let $X_1,\ldots,X_M$ be independent Bernoulli random variables with parameter p and join with an edge the pair of nodes in the i^{th} subset if and only if $X_i = 1$.

The second method consists of explicitly defining the *probability space* $\mathcal{G}_{n,p}$, whose elements are random graphs with set of nodes $V = [n]$. One way of doing this, is the following [Die00]: Let $[V]^2$ be the set of 2-element subsets of V, *i.e.,* $[V]^2$ is the set of all potential edges of an undirected, simple graph on V. For every potential edge $e \in [V]^2$, let $\Omega_e := \{0_e, 1_e\}$ be a probability space for which the measure is specified as: $\mathbb{P}_e(\{1_e\}) := p$ and $\mathbb{P}_e(\{0_e\}) := 1-p$. Then, the probability space $\mathcal{G}_{n,p}$ is defined as the product space of all the spaces Ω_e:

$$\mathcal{G}_{n,p} := \prod_{e \in [V]^2} \Omega_e$$

Thus, formally an element of $\mathcal{G}_{n,p}$ is a map assigning to every $e \in [V]^2$ either 0_e or 1_e, and the probability measure $\mathbb{P}$ on $\mathcal{G}_{n,p}$ is the product measure of all the measures $\mathbb{P}_e$. In practice, each point $\omega \in \mathcal{G}_{n,p}$ is assumed to represent a graph on V with edge set $\{e \mid \omega(e) = 1_e\}$. Each element of the space $\mathcal{G}_{n,p}$ is called a *random graph* on V with edge-probability p.

Example 1.1: Assume that $n = 3$ and $p = 0.4$. The potential edges of a simple graph on the nodes 1, 2, 3 may be enumerated as $e_1 = (1,2)$, $e_2 = (1,3)$, $e_3 = (2,3)$. Figure 1.1 shows the elements of the space $\mathcal{G}_{3,0.4}$ (in one-to-one correspondence with the simple graphs on three nodes) and the probability assigned to each these elements. As an illustration, the probability assigned to the element $0_{e_1}0_{e_2}0_{e_3}$ of the space $G_{3,0.4}$ is:

$$\mathbb{P}(0_{e_1}0_{e_2}0_{e_3}) = \mathbb{P}_{e_1}(0_{e_1})\mathbb{P}_{e_2}(0_{e_2})\mathbb{P}_{e_3}(0_{e_3}) = (.4)(.4)(.4) = .064$$

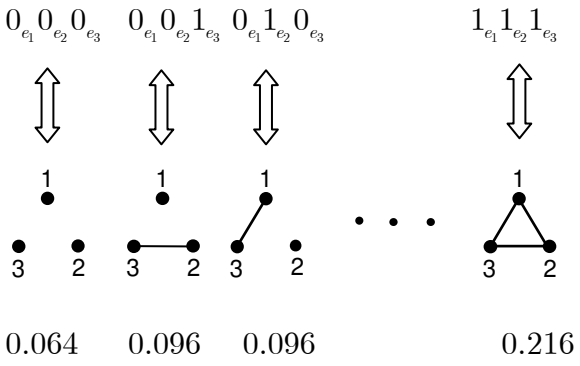


Figure 1.1: The elements of the binomial probability space $G_{3,0.4}$

The probability assigned to each of the remaining elements of this space is obtained similarly. ■

Having defined the probability space of a random graph model, one can talk about such probabilistic concepts as events, random variables, moments, convergence, *etc*.

Events: Any set of graphs on $V = [n]$ is an *event* in $\mathcal{G}_{n,p}$. In particular, for every $e \in [V]^2$ the set A_e of all graphs $G_{n,p}$ having e as an edge is an event: the event that e is an edge $G_{n,p}$. It is straightforward to show that the events A_e are *independent* and occur with probability p ([Die00], p. 231). As another example, let H be a given graph of order n and size m and denote by A_H the event that H is a subgraph of $G_{n,p}$. Since A_H consists of those graphs in the space $\mathcal{G}_{n,p}$ that contain all the m edges of H, it follows that $\mathbb{P}(A_H) = p^m$.

Random variables and moments: A random variable X defined on the space $\mathcal{G}_{n,p}$ is a function $X : \mathcal{G}_{n,p} \to \mathbb{R}$. Thus, the graph parameters discussed earlier such as degree of a random node, average degree, degree correlation, average distance, diameter, robustness, clustering coefficient, *etc*., are examples of random variables on $\mathcal{G}_{n,p}$. Ideally, one would like to know the *probability distribution function* of each such graph parameter. In practice, this is often difficult to achieve; in such cases one usually resorts to the study of the first and the second moments of these random variables. As an example, let X denote the number of triangles—*i.e*., cycles of length three. It may be shown that $\mathbb{E}(X) = n(n-1)(n-2)p^3/6$. This follows immediately from the following two observations: (i) each fixed triangle is a subgraph of $G_{n,p}$ with probability p^3; (ii) there are $n(n-1)(n-2)$ distinct three-element sequences with elements from $[n]$ and each triangle is specified by 6 of these sequences.

Asymptotics: Of particular interest in the study of random graph models is the asymptotic case $n \to \infty$. An event A is said to happen *asymptotically almost surely* (*a.a.s.*) if $\mathbb{P}(A) \to 1$ as $n \to \infty$. Let, for instance, $\overline{A_e}$ be the event that e is not an edge of $G_{n,p}$, for some fixed $e \in [V]^2$. From an earlier observation if follows that $P(\overline{A_e}) = 1 - p$. If $p = p(n) = 1/n$, then the event $\overline{A_e}$ happens *a.a.s.*

Critical functions: A *graph property* $\mathcal{P}$ is formally defined as a set of graphs closed under isomorphism. For instance, the property "G is connected" consists of all connected graphs. Given a graph property $\mathcal{P}$, it is said that *almost every (a.e.) graph* has $\mathcal{P}$, if $\mathbb{P}(G_{n,p} \in \mathcal{P}) \to 1$ as $n \to \infty$. The most interesting cases in the study of the space $\mathcal{G}_{n,p}$ arise

when $p = p(n)$ is a decreasing function of n. A real function $t = t(n)$ is called a *critical* (or, *threshold*) function for a graph property $\mathcal{P}$, if

$$\lim_{n\to\infty} \mathbb{P}\left(G_{n,p} \in \mathcal{P}\right) = \begin{cases} 0 & \text{if } \lim_{n\to\infty} p(n)/t(n) = 0 \\ 1 & \text{if } \lim_{n\to\infty} p(n)/t(n) = \infty. \end{cases}$$

In the next chapter, several other random graph models are discussed.

1.3. Some Experimental Studies of Web-like Networks

We conclude Chapter 1 with a brief review of some empirical studies of the Web, the Internet and other selected web-like networks.

The Web: This network can be modeled as a graph at two different levels: At the *web page* level, nodes stand for web pages, while arcs stand for hyperlinks between web pages. At the *web site* level, nodes stand for web sites—which, generally, comprise many web pages. Two web sites are joined by an arc if and only if there is at least a pair of web pages—one in each web site—that are joined by an hyperlink. Unless otherwise indicated, the experimental results cited below relate to the graph model of the Web at the page level.

First, it has been found that both in- and out-degree of the Web graph follow power-law distributions with exponents 2.1 and 2.4, respectively [DKMR01, BKMR00]. In terms of connectedness, it has been found that the Web has an interesting structure—called the "bow-tie" [BKMR00]—essentially consisting of a large strongly connected component (the *core*) and two other connected components that have only unidirectional links to and from the core, respectively. The size of connected and biconnected components follows a power-law distribution [BKMR00, AH01, DKMR01]. Further, the Web graph satisfies the "small-world"

property, *i.e.,* it has a small diameter (*e.g.,* the value 19 has been reported as an estimate for the diameter of the whole Web in [AJB99] and the value 28 as an estimate for the diameter of the strongly connected component in [BKMR00]). In addition, the Web graph has been found to contain large quantities of some signature subgraphs such as *complete bipartite cores* and *webrings* [KRRT99], certain subgraphs on 3 or 4 vertices (*e.g.,* triples of nodes where each pair is linked with two arcs oriented in opposite directions) [MSIK02], *etc.* Finally, it has been shown that the Web displays a *fractal-like self-similarity*: certain sub-regions of the Web display the same characteristics as the Web itself [DKMR01]. This self-similarity is both *distributional* and *structural* and is displayed at various scales: First, if a subgraph induced by a sufficiently large set of web pages that form a *thematically unified cluster* (TUC)—a set of web pages sharing a common theme, such as content, geographical location, *etc*.—is fixed, then several parameters of that subgraph such as degree, or the size of connected components, follow *power-law* distributions. In addition, many such TUCs have a "bow-tie" structure and contain large numbers of small bipartite cores. Second, if the Web graph is modeled at the level of *web sites*, then the same structure and distributions are observed (with approximately the same constants).

The Internet: The Internet can also be modeled at two levels: microscopic and macroscopic. In the *Microscopic Internet graph*, nodes stand for routers and hosts, while edges represent communication links. The *Macroscopic Internet graph* can be thought of as a contraction of the Microscopic Internet graph: here, each node represents an autonomous system (which incorporates a number of routers). Two nodes in the *Macroscopic Internet graph* are adjacent if there is at least one pair of routers (belonging to different autonomous systems) that can communicate. Both of these graphs have a power-law degree distribution [FFF99, GT00]. Further, it has been shown [YJB02] that the Macroscopic Internet graph has clustering

coefficient between 0.18 and 0.3 and average distance between 3.70 and 3.77. Similar values for clustering coefficient and average distance were found by another study [VPV02] where some additional parameters, such as *node betweenness* and *degree correlations* were also studied.

Next, we summarize the salient properties of some other web-like networks. Many of these networks are more naturally modeled as *undirected* graphs, while others as *directed* graphs. For each network below we have indicated what the nodes and edges (arcs) represent. For a more elaborate description of these networks and additional examples the reader is referred to the surveys [DM02, AB02, New03b].

- *Citation network:* nodes – *published articles*; arcs – *citations* of one article from another.
- *Food-web network:* nodes – *species*; arcs – *prey/predator* relationships.
- *Movie-actors network:* nodes – *actors*; edges – *collaboration* in a movie.
- *Neural networks:* nodes – *neurons*; edges – *synaptic connections*.
- *Peer-to-peer networks:* nodes – *computers*; edges – *file-sharing* between computers
- *Phone-call network:* nodes – *phone numbers*; arcs – *completed calls* during a fixed period
- *Science collaboration network:* nodes – *scientists*; edges – *collaboration* between scientists
- *Word co-occurrences network:* nodes – *words*; edges – *co-occurrence* of words in consecutive positions or one word apart in a sentence.

Table 1.1 summarizes the known properties of these networks. From the data in this table one can observe three common characteristics of web-like networks: (1) the *average distance* is generally small *i.e.,* these networks satisfy the *small-world* property; (2) the *clustering coefficient* is significantly greater than zero; and (3) the *degree distribution* generally follows a power-law with exponent that falls between 2 and 3 (in the case of directed networks the same is true for

both in- and out-degrees). Furthermore, the networks shown in Table 1.1 and many other web-like networks are *sparse i.e.,* the number of edges is of the same order as the number of nodes (or, put differently, their average degree is small).

Table 1.1: Some parameters of selected real-world networks

Network	n	m	$d(G)$	$\mathrm{dist}(G)$	γ	T	r
Peer-to-peer	880	1296	1.47	4.28	2.1	0.011	–0.366
Citation	783339	6716198	8.57	–	3.0/–	–	
Math collab.	253339	496489	3.92	7.57	–	0.34	0.12
Movie actors	449913	25516482	113.43	3.48	2.3	0.78	0.208
Phone calls	47×10^6	80×10^6	3.16	–	2.1	–	–
Word co-occ.	460902	17×10^6	70.13	–	2.7	0.44	–
Marine food	135	598	4.43	2.05	–	0.23	–0.263
Neural netw.	307	2359	7.68	3.97	–	0.28	–0.226

A – sign indicates that data is not available

It is natural to ask that all mathematical models of web-like networks should, at the very least, satisfy the properties above. Several new random graph models displaying these properties have been discovered recently. We discuss these models in the next chapter.

2. RANDOM GRAPH MODELS

The ubiquity and the increasing importance of web-like networks have spawned a truly cross-disciplinary research aimed at understanding their fundamental properties and functions.

Two groups of questions are of main interest: First, we would like to know the *graph structure* of these networks. Some of the simplest questions that may be asked for each network are: Is it sparse or dense? What does its degree sequence look like? How many connected components does the network have, and what are their sizes? If a network is connected, how robust is it, *i.e.,* what fraction of nodes must be removed to break the network into disconnected components? What is the typical distance between two nodes? *etc*. Second, we need to understand how the structure of these networks affects the behavior of dynamic processes that occur on them. For instance, we would like to know how social networks facilitate or constrain the spread of diseases, or how the properties of the Internet can be exploited to devise efficient strategies for containing the spread of viruses and worms, *etc*. Answering such questions precisely has proven to be hard because virtually all web-like networks are *dynamic*, *i.e.*, their sets of nodes and edges change continuously due to the birth of new nodes and edges or due to the death of existing ones and because these networks are generally *enormous*.

Given such dynamic and massive structures, is there any hope for researchers to gain some insight into their function and structural properties? The key to answering this question has turned out to be the use of nondeterministic methods. In particular, an iterative interplay between *experimental data* and *modeling*—where both data and models are *statistical* in nature—has emerged as a promising tool in advancing our understanding of web-like networks. This interplay unfolds as follows: first, a small number of experimentally found properties of real networks are used as the basis for the design of a mathematical model that displays all of these

properties; next, this model is investigated analytically to obtain additional properties; finally, the newly-derived properties are validated against the real-world data and the whole process is repeated in order to obtain models that are as accurate as possible. Correct models of web-like networks serve two major purposes: (i) first, they can provide an insight into the basic processes responsible for the structure of such networks; and (ii) they can be used as tested to study the behavior of dynamic processes occurring on web-like networks and the performance of various network algorithms.

Nondeterministic models of networks can be traced back to the 1950's with the introduction of the *classical random graphs* $G_{n,m}$ by Erdös and Rényi [ER59, ER60] and $G_{n,p}$ by Gilbert [Gil59]. For several decades, these random graphs have been studied intensively both for their theoretical interest and as the only sensible and rigorous approach in modeling large, random, real-life networks. During those years, detailed topological data on web-like networks was generally unavailable and the computational power to analyze such networks was insufficient. Therefore, a comparison between real networks and the classical random graph $G_{n,p}$ was difficult. In recent years, the situation has changed: the computerization of data acquisition (*e.g.*, obtaining the structure of the Web via *crawling*), as well as the availability of high computational power and efficient algorithms have allowed researchers to carry out experiments on large data sets extracted from real-life networks. The results of these experiments have made it clear that classical random graphs differ significantly from web-like networks, especially in the (a) *degree distribution*; (b) *clustering coefficient*; and (c) *community structure*, as explained in Chapter 1.

Beginning with the *small-world* model by Watts and Strogatz [WS98] and the *preferential attachment* model by Barabási *et al*. [BAJ99], many new random graph models have

been defined and studied in the recent years in an effort to explain these new empirical findings. Currently, the work in this area is growing rapidly and may be grouped into three main categories: (1) experimental study of real networks; (2) analysis of the new random graph models using heuristic and/or rigorous techniques; and (3) design of new network algorithms. The surveys by Dorogovtsev and Mendes [DM02] and Albert and Barabási [AB02] summarized the initial work in the field. Bollobás and Riordan [BR03b] surveyed the initial rigorous mathematical results in this area. The list of 429 references in the more recent survey by Newman [New03b] is an indicator of the rapid growth of the field. A number of books [Bar02, Buc02, Wat03, DM03, BFS03] have also appeared on this topic.

This chapter is devoted to the discussion of various random graph models and some techniques for analyzing them. Random graph models may be classified into two groups: (1) *static* (also known as explicit or off-line); and (2) *dynamic* (also known as recursive or on-line). The difference between the two groups may be explained as follows: In a static model, the set of nodes is fixed at the beginning of the algorithm that defines it (the set of edges may change). The random graph $G_{n,p}$, described in the previous chapter, is an example of a static model. On the other hand, in a dynamic model, the sets of both nodes and edges may change during the course of the defining algorithm. Several examples of dynamic random graph models—which have emerged as more likely candidates for modeling web-like networks accurately—will be presented in this chapter. However, for completeness, we first provide a short discussion of three static models: (i) the classical random graphs; (ii) the small-world graphs; and (iii) the random graphs with given degree distribution.

2.1. Static Random Graphs

Classical random graphs

The definition of random graph model $G_{n,p}$ (known as the *binomial* model) was shown in Section 1.2. There is an equivalent model, known as the *uniform* model and denoted by $G_{n,m}$ [ER59, ER60], which is formally defined as follows: Let $\mathcal{G}_{n,m}$ be the set of all undirected, simple and labeled graphs of order n and size m; this set clearly has $\binom{M}{m}$ elements, where $M = n(n-1)/2$. To turn $G_{n,m}$ into a probability space, each of its elements is assigned a probability of $1/|\mathcal{G}_{n,m}|$. Any element of the probability space $\mathcal{G}_{n,m}$ is called a (uniform) random graph and is denoted by $G_{n,m}$.

Example 2.1: Assume that $n = 4$ and $m = 5$. It follows that $M = 6$ and $|\mathcal{G}_{4,5}| = \binom{6}{5} = 6$. The elements of the space $\mathcal{G}_{4,5}$ are shown in Figure 2.1, below:

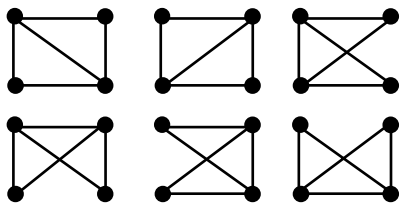


Figure 2.1. The elements of the uniform probability space $\mathcal{G}_{4,5}$.

Each of these six elements is assigned a probability of $1/6$ in the probability space $\mathcal{G}_{4,5}$. ■

Additional examples and further discussion of classical random graphs at an introductory level may be found in [BBSR05].

The following algorithm generates a random graph $G_{n,m}$: Beginning with n isolated nodes, add one by one m edges chosen independently, uniformly at random (avoiding self-loops and parallel edges).

It has been shown that the random graphs $G_{n,p}$ and $G_{n,m}$ are essentially the same for $m = pM$ [Bol79]. Thus, choosing one of these two models to work with is a matter of convenience.

Many papers and a number of books have been written on classical random graphs. The book by Palmer [Pal85] is a gentle introduction to the area; the book by Bollobás [Bol85] provides an in-depth analysis of the properties of random graphs and the book by Janson *et al*. [JLR00] is a comprehensive treatment that also includes the major recent developments.

Next, we describe by informal arguments some of the most salient properties of $G_{n,p}$ and compare them with the corresponding properties of web-like networks. First, consider the *degree distribution* of $G_{n,p}$. Let v be a node selected uniformly at random from $G_{n,p}$ and denote by $\mathbb{P}(k)$ the probability distribution function of the random variable $d(v)$. Since each of the remaining $n-1$ nodes of $G_{n,p}$ can independently be a neighbor of v with probability p, it follows that $d(v)$ has a binomial distribution with parameters $(n-1)$ and p, *i.e.,* $\mathbb{P}(k) = \binom{n}{k} p^k (1-p)^{n-k}$. This distribution is clearly quite different from the power-law degree distribution observed in many web-like networks.

Next, consider the *clustering coefficient* of $G_{n,p}$. The expected number of neighbors of a node v that are neighbors themselves is $d_v(d_v - 1)p/2$, *i.e.*, the expected clustering coefficient of each node is p. Therefore, $\mathbb{E}\big(C(G_{n,p})\big) = p = \mathbb{E}\big(d(G_{n,p})\big)/n$, implying that the clustering

coefficient of $G_{n,p}$ becomes vanishingly small when n grows very large with the average degree $d(G_{n,p})$ remaining constant. In fact, it has been observed the clustering coefficient of many web-like networks is 10^2-10^3 times larger than the clustering coefficient of a classical random graph of the same order [AB02].

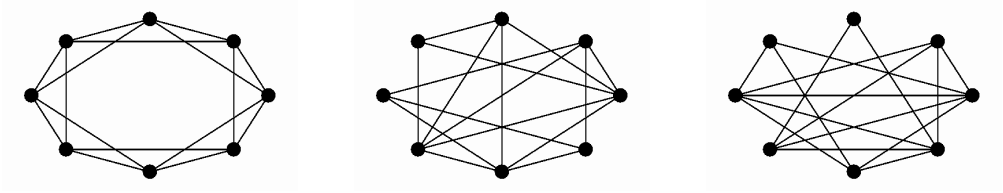
Now, consider the *degree correlation* defined as the Pearson correlation coefficient r of the degrees at either end of a randomly chosen edge. Since the edges of $G_{n,p}$ are placed independently of the degrees of the two ends, it follows that $r = 0$. On the other hand, a number of web-like networks have been found to have nonzero degree correlations (Table 1.1).

Having pointed out some differences between classical random graphs and web-like networks, we note that $G_{n,p}$ has an important property in common with web-like networks: In many ranges of p, $G_{n,p}$ satisfies the small-world property, *e.g.*, [Bur74, Bol90, Luc98, CL01].

Small-world graphs

Watts and Strogatz [WS98] analyzed the following interpolation between regular and random graphs. Consider n nodes $v_1,\ldots,v_n$ which are spread equidistantly along the curve of a ring. Assume that each node is linked with an edge to each of its k nearest neighbors on either side (Figure 2.2(a)). The resulting graph is called the *k-nearest neighbor regular lattice* [WS98] (it is the same as the *Harary graph* $H_{2k,n}$, which is the smallest *k*-connected simple graph of order n and size nk). Now, pick any node v_i and perform the following *rewiring* procedure for each of the k edges incident on v_i in the clockwise sense: with probability p, reattach this edge so that it joins v_i to a node chosen uniformly at random among the remaining $n-1$ nodes (disallowing parallel edges); with probability $1-p$ leave the edge in place. The procedure

above is repeated by moving clockwise around the ring, considering each node in turn until one lap is completed. This construction, known as the *WS rewiring* algorithm, allows one to "tune" the graph between regularity ($p = 0$) and disorder ($p = 1$) and thereby to study the region $0 \leq p \leq 1$. An illustration of small-world graph is shown in Figure 2.2.



(a) (b) (c)

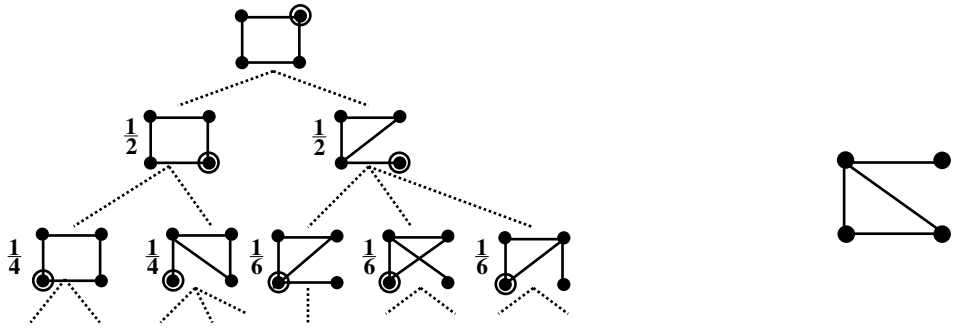
Figure 2.2. A small-world random graph with $n = 8$, $k = 2$ and (a) $p = 0$; (b) $p = 0.5$; (c) $p = 1$.

Newman and Watts [NW99] introduced a slightly-different version of the small-world model—the *edge-addition* algorithm—where new edges are repeatedly added between random pairs of nodes, instead of existing edges being rewired.

It should be mentioned that the random graph obtained at the end of the WS procedure with $p = 1$ is not the same as $G_{n,m}$, because after the rewiring has been completed, every node of the resulting random graph will have degree at least k. A better understanding of the relationship between the small-world model and the classical random graph model may be achieved by looking at the mathematical definition of small-world model: Let $\mathcal{G}_{n,k,p}$ denote the probability space of the small-world model (after the rewiring has been completed) with parameters n, k and p. The following example shows that the uniform random graph space

$\mathcal{G}_{n,nk}$, and the small-world graph space $\mathcal{G}_{n,k,1}$, differ both in the number of their elements and in the probabilities they assign to equal graphs.

Example 2.2: Consider the small-world probability space $\mathcal{G}_{4,1,1}$. It is straightforward to enumerate all possible graphs that may arise during the WS edge-rewiring procedure by using a tree, as shown in Figure 2.3(a). Each level of this tree depicts the rewiring of the edges of a single node (shown encircled). The number to the left of each graph denotes the probability that this graph will arise during the WS procedure. By completing the last two levels of the tree in Figure 2.3(a) one may see that the space $\mathcal{G}_{4,1,1}$ has 14 elements. Furthermore, the probabilities associated with these elements are obtained, respectively, by dividing $9,15,10,10,10,12,6,12,6,12,18,12,6,6$ by 144. On the other hand the classical random graph space $\mathcal{G}_{4,4}$ consists of 15 graphs, each with an assigned probability of $1/15$.



(a) (b)

Figure 2.3. (a) Enumeration of the elements of the space $\mathcal{G}_{4,1,1}$; (b) The single graph that belongs to the space $\mathcal{G}_{4,4}$ but not to the space $\mathcal{G}_{4,1,1}$.

It is easy to verify that the graph shown in Figure 2.3(b) is the only graph that belongs to $\mathcal{G}_{4,4}$ but not to $\mathcal{G}_{4,1,1}$. ■

Watts and Strogatz [WS98] looked at two properties of the small-world model—the average distance and the clustering coefficient—for $0 \leq p \leq 1$. Their findings may be summarized as follows:

(i) Total order, $p = 0$: The average distance of *k*-nearest neighbor lattice grows as $n/4k$ *i.e., linearly* with n and hence this graph does not display the *small-world* property. Further, the clustering coefficient of this graph is asymptotically close to $3/4$, *i.e.,* the graph is highly clustered.

(ii) Total disorder, $p = 1$: In this case, it was demonstrated experimentally that the average distance of the resulting graph grows logarithmically with n, *i.e.,* the graph has the *small-world* property. Further, the clustering coefficient in this case approaches zero as $n \to \infty$.

(iii) From order to disorder, $0 < p < 1$: The above two extreme cases indicate that large average distance is associated with large clustering coefficient, and small average distance with small clustering coefficient. Surprisingly, it was found [WS98] that there is a wide range of p ($0.01 < p < 0.1$) where the average distance is small whereas the clustering coefficient is large. The above authors proposed the term *"small-world network"* to refer to networks that have small average distance and large clustering coefficient. The explanation offered for the small-world phenomenon is that only a few rewired edges are sufficient to decrease the average distance significantly without affecting much the clustering coefficient. It turned out that this fact had been know in the random graph community long before Watts and Strogatz rediscovered it (see [BR02]). However, the paper by Watts and Strogatz drew a lot of attention and many additional papers on the properties of small-world networks have been published (*e.g.*, [BA99b, NW99, BW00, MN00, NMW00, Kle00, AKS02]).

Despite the fact that, for appropriate values of p, the small-world random graph model exhibits two of the salient properties of web-like networks, it was observed [BW00] that for all values of p, the degree distribution is essentially a binomial one and hence it differs substantially from the power-law degree distribution of web-like networks.

Random graphs with given degree distribution

A sequence of positive integers $\mathcal{D} = (d_1, d_2, \ldots, d_n)$ is said to be *graphic* if there is a graph having $\mathcal{D}$ as degree sequence. Characterizations of graphic sequences have been known for many years (*e.g.*, [EG60, Hak62]). Given a graphic sequence $\mathcal{D}$, denote by $\Omega_{\mathcal{D}}$ the set of all graphs with set of nodes $[n]$ and degree sequence $\mathcal{D}$. A *random graph with degree sequence* $\mathcal{D}$ is a graph chosen uniformly at random from $\Omega_{\mathcal{D}}$.

Example 2.3: Consider the degree sequence $\{2,2,1,1,2,2\}$. It may be seen that there are only two simple graphs on 6 nodes having this degree sequence. These graphs are shown in Figure 2.4:

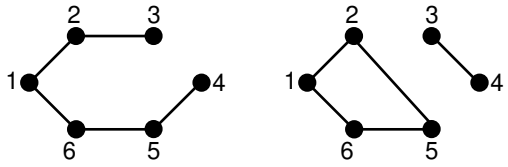


Figure 2.4. The two simple graphs on 6 nodes having degree sequence $\{2,2,1,1,2,2\}$.

Hence, the probability space $\Omega_{\{2,2,1,1,2,2\}}$ consists of these two graphs each associated with a probability of 0.5. ■

Because it is difficult to design an algorithm that generates random graphs with given degree sequence, it has become standard to work instead with *random configurations on a given*

degree sequence and use some lemmas that allow one to translate the results from one model to the other. The configuration model was introduced by Bender and Canfield [BC78] and refined by Bollobás [Bol85]. The following procedure generates a random configuration with n nodes on the degree sequence $\mathcal{D} = (d_1, \ldots, d_n)$: (1) form a set L containing d_i copies of node i for $i = 1, \ldots, n$, (2) choose a random matching of the elements of L. Each configuration represents an underlying *multigraph* (*i.e.,* a graph with self-loops and parallel edges) whose edges are defined by the pairs in the matching.

A slightly different model—random graphs *with given expected degrees*—was proposed by Chung and Lu [CL02]. Given a degree sequence $\mathcal{D} = (d_1, \ldots, d_n)$ a random graph from this family may be generated by carrying out the following steps: (i) Begin with n isolated nodes; and (ii) Join each pair of nodes (i, j) independently with an edge with probability proportional to $d_i d_j$.

The notion of random graph with given degree sequence may be extended to that of a random graph *with a given degree distribution*: Given an arbitrary discrete probability distribution $\mathbb{P}(k)$, a random graph on n nodes having $\mathbb{P}(k)$ as degree distribution, may be obtained by generating a random graph with degree sequence consisting of $\mathbb{P}(i)n$ nodes of degree i, for $i = 1, \ldots, n$.

Molloy and Reed [MR95] derived the following result about random graphs with given degree distribution: Let $\lambda_0, \lambda_1, \ldots$ be a given distribution. If $\sum_{k \geq 1} k(k-2)\lambda_k > 0$, then a random graph G with the given degree distribution, *a.s.* has a giant component. Otherwise, if $\sum_{k \geq 1} k(k-2)\lambda_k < 0$ then *a.s.* all the connected components of G are of size less than

$O(\log n)$. In a sequel paper [MR98], the same authors analyzed the size of the giant component in the case when $\sum_{k\geq 1} k(k-2)\lambda_k > 0$.

Motivated by the empirical observations of power-law degree distribution in some web-like networks, Aiello *et al.* [ACL01], proposed and analyzed a model of random graphs with given *power-law* degree distribution. By applying the results of [MR95, MR98] and other techniques these authors obtained the expected distribution of the size of connected components in such random graphs.

Newman *et al*. [NSW01] proposed a method for analyzing random graphs with given degree distribution based on the formalism of *generating functions* [Wil90]. Among other things, they re-derived the result of [MR95] cited earlier and showed how their technique may be also applied to *directed* and *bipartite* graphs.

Cohen *et al*. studied the robustness of random graphs with given power-law degree distribution under random node removal [CEBH00] and intentional attack [CEBH01]. A similar study was carried out by Callaway *et al*. in [CNSW00]. Schwartz *et al.* [SCBB02] studied the robustness of *directed* random graphs with given degree distribution, while Cooper and Frieze [CF04] studied the size of the strongly connected component in such graphs.

Chung and Lu have investigated some asymptotic properties of *graphs with given expected degrees*. In [CL02] they studied the connected components, in [CL03] the average distance, and with Vu in [CLV03] the spectra, of such graphs.

Random graphs with given degree distribution and their variants provide a convenient tool for modeling web-like networks. However, these models do not provide any insight into the elementary processes responsible for the structure or the evolution of web-like networks.

Next, we turn to the main topic of this chapter: the dynamic random graph models. These models not only generate random graphs that display the known properties of web-like networks, but also highlight some basic mechanisms that can potentially explain the evolution of such networks.

2.2. Dynamic Random Graphs

A dynamic random graph model is broadly defined as a *discrete-time graph process* $\{G_t(V_t, E_t)\}_{t\geq 1}$ where G_1 is a fixed, small graph and for all $t > 1$, the graph G_{t+1} is obtained by making small changes to the graph G_t, according to some stochastic rules. Unless otherwise indicated, it will be assumed that each node is labeled with the time-step during which it is born, *i.e.*, $V_t = \{1,\ldots,t\} = [t]$.

The underlying stochastic rule employed in virtually all dynamic random graph models has been some form of either *preferential attachment* [BA99a] or *copying* [KKRS00]. The ideas behind these two stochastic rules are explained next.

Preferential attachment: In the context of dynamic random graph models this rule was introduced by Barabási and Albert [BA99a]. It turned out that this stochastic rule has a long history and has been previously used in various other fields such as economics and biology [Mit04]. It can be explained as follows: Let $t+1$ be the only node and e_{t+1} the only edge added at $(t+1)^{th}$ time-step of a dynamic random graph model. Assume that e_{t+1} is incident on the node $t+1$ while the other end of e_{t+1} is chosen at random from the set of existing nodes V_t based on some probability measure $\mathbb{P}_{t+1} : V_t \to [0,1]$, where $\mathbb{P}_{t+1}(s)$ denotes the probability of node s being chosen. Such a probability measure is called a *preferential attachment rule* if

$\mathbb{P}_{t+1}(s)$ is *proportional* to the degree $d_t(s)$ of node s in the graph G_t. A dynamic random graph model which is based on a preferential attachment rule, is called a *preferential attachment model*.

Copying: This rule was introduced by Kumar *et al.* [KKRS00]. It defines a *directed* dynamic random graph model as follows: Let again $t+1$ and e_{t+1} be the only node and the only *directed arc*, respectively, added at $(t+1)^{th}$ step and assume that node $t+1$ is the *tail* of arc e_{t+1}. Let $s \in V_t$ be a node selected uniformly at random—referred to as the *prototype* of node $t+1$. According to the copying rule, with probability p the *head* of e_{t+1} is chosen uniformly at random from V_t, whereas with probability $q = 1 - p$ it is taken to be the *head* of the arc having the prototype node s as *tail* (*i.e.*, it is *copied* from the prototype s). The intuition behind this model comes from Web authoring: When an author decides to create a new web page, the author is likely to have some topic in mind. The choice of *prototype* represents the choice of the topic while *copying* reflects the intuition that a new web page about the topic will probably link to many pages that are already linked-to by existing other web pages, but it will probably also link to some new pages.

We have classified the dynamic random graph models according to two criteria:

(i) *Preferential attachment* vs. *copying* models: First, we have distinguished between the models that are based on some form of preferential attachment rule and those that are based on some form of copying.

(ii) *Birth-only* vs. *birth-death* models: Second, we have distinguished between the models in which only the addition of nodes and edges takes place (birth-only) and the models were both addition and deletion of nodes and edges takes place (birth-death).

With this classification scheme in mind, we are now ready to list the major dynamic random graph models of web-like networks. Note that the vast majority of these models fall in the category of birth-only, preferential attachment models.

Birth-only preferential attachment models

Barabási-Albert (BA) model [BA99a]. The procedure that generates a BA random graph, can be described as follows: Beginning with ε_0 isolated nodes, in each time-step a new node and $\varepsilon \leq \varepsilon_0$ new edges are added. The ε new edges are all incident on the new node. The other end of each new edge is chosen based on the following preferential attachment rule:

$$\mathbb{P}_{t+1}(s) = \frac{d_t(s)}{2\varepsilon t} \text{ for } 1 \leq s \leq t. \qquad (2.1)$$

There are two problems with the definition of this model: First, it starts out with a set V_1 of isolated nodes. Hence, for every node $s \in V_1$, the probability $\mathbb{P}_1(s)$ is equal to zero and thus the process can't get started. This difficulty may be sidestepped by using different approaches such as: (1) beginning with a small graph with non-isolated nodes (see LCD model below); and (2) modifying the preferential attachment rule (7) (see DM-1 model below). Another problem with the BA model is that it does not allow self-loops and parallel arcs. This restriction prevents the ε new edges from being added independently, which in turn makes the model difficult to analyze. The next two models are rigorous versions of the BA model:

LCD model [BRST01]. This model, defined by Bollobás *et al.* [BRST01], takes its name from the fact that it can be analyzed via Linearized Chord Diagrams (the definition of this construct has been omitted since we do not use it in this dissertation; see [BRST01]). The LCD model defines two related graph processes: one where a single edge is added and another where

several edges are added, in each step. The single-edge version is defined as follows: Beginning with an empty graph, or with a graph consisting of one node and a self-loop, in each step, a node together with an edge incident on the new node, are added. The other endpoint of the new edge is chosen based on the following preferential attachment rule:

$$\mathbb{P}_{t+1}(s) = \begin{cases} \dfrac{d_t(s)}{2t+1} & \text{for } 1 \leq s \leq t; \\ \dfrac{1}{2t+1} & \text{for } s = t+1. \end{cases} \tag{2.2}$$

The multi-edge version is defined in the same way as the single-edge one except that in each time-step, ε edges incident on the new node are added. These edges are added one by one, counting the previous edges as well as the 'outward half' of the edge being added, as already contributing to the degrees. The reason for this precise rule is that it yields the following equivalent procedure for defining the multi-edge version of LCD model: First, run the procedure for the single-edge version on a sequence of εt nodes and then *contract* each group of ε consecutive nodes into super-nodes. The advantage of having this alternative definition is that many results for the multi-edge version may be obtained by deriving them first for the single-edge one—which is easier to analyze—and then converting to the multi-edge version.

Figure 2.5 shows a realization of LCD model with $t = 100$ and $\varepsilon = 1$. Note that a few nodes have high degree (*e.g.*, $d(2) = 14$, $d(5) = 9$) while most of the nodes have small degree—the hallmark of a power-law degree distribution. Furthermore, the labels of the high-degree nodes are small, illustrating the so called "rich-gets-richer" phenomenon observed in the preferential attachment models.

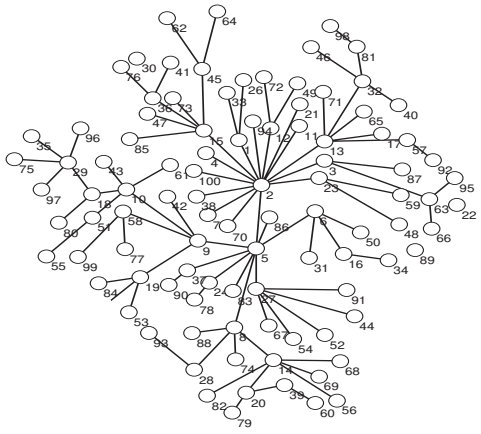


Figure 2.5. A realization of the LCD model.

The degree distribution in the models BA and LCD follows a power-law with exponent equal to 3 (see Section 2.3). Since in many web-like networks this exponent varies between 2 and 3, many parameterized forms of preferential attachment, aiming to generate exponents that fall in this range, have been proposed. Some of these models are shown next:

Dorogovtsev-Mendes-1 (DM-1) model [DMS00]. This model was introduced by [DMS00] and was also studied by [KRR01, BO04]. Beginning with an empty graph, in each time-step, a new node and ε arcs are added. The ε arcs introduced at step $(t+1)$ are attached as follows: For each arc, the *tail* is chosen uniformly at random among the $(t+1)$ existing nodes (*i.e.*, the new node is included in the selection), while the *head* is chosen via a preferential attachment rule based on *in-degree*. To avoid the first problem in the BA model, the DM-1 model employs a shifted linear preferential attachment rule given by

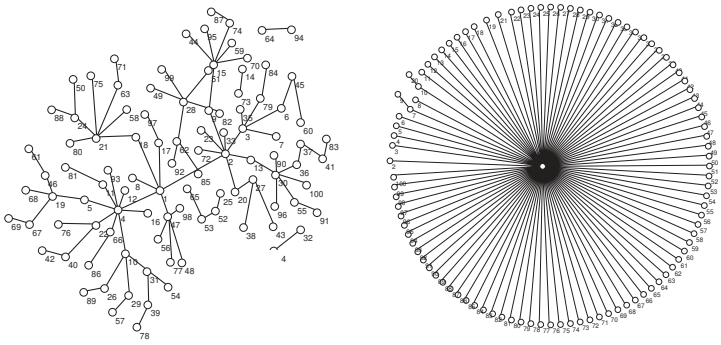
$$\mathbb{P}_{t+1}(s) = \frac{\lambda + d_t^-(s)}{\sum_{j \in V_t} \left(\lambda + d_t^-(j)\right)} \qquad (2.3)$$

for $1 \leq s \leq t$. The term $\lambda > 0$ is called the *initial attractiveness* of nodes. The DM-1 model permits *parallel* arcs and *self-loops* in order to avoid the second problem with the BA model mentioned earlier.

Krapivsky-Redner-1 (KR-1) model [KR01]. The preferential attachment rule in the KR-1 model is non-linear in $d_t(s)$ and is given by

$$\mathbb{P}_{t+1}(s) = \frac{d_t(s)^\alpha}{\sum_{j \in V_t} d_t(j)^\alpha} \qquad (2.4)$$

for $1 \leq s \leq t$. Krapivsky and Redner [KR01] discovered that for the KR-1 model with $\alpha \neq 1$, the asymptotic degree distribution of G_t does not follow a power-law. Figure 2.6 shows two realizations of KR-1 model with $t = 100$. It is visually clear that neither of these two graphs is likely to have a power-law degree distribution.



(a) (b)

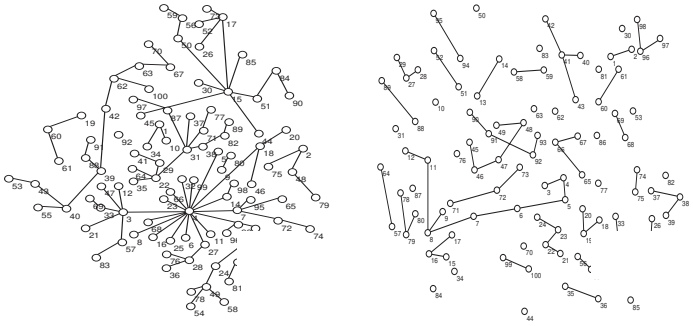
Figure 2.6. Two realizations of KR-1 model: (a) $\alpha = 0.4$; (b) $\alpha = 3$.

Dorogovtsev-Mendes-2 (DM-2) model [DM00a]. This model was inspired by a phenomenon observed in citation networks: old papers are generally cited less than new ones. In

order to incorporate this idea of *aging* of nodes, the preferential attachment rule is changed as follows:

$$\mathbb{P}_{t+1}(s) = \frac{d_t(s)(t-s)^{-\alpha}}{\sum_{j=1}^{t} d_t(j)(t-j)^{-\alpha}} \qquad (2.5)$$

Here $(t-s)$ denotes the age of node s at time t. Figure 2.7 shows two realizations of this model with two different values of the parameter α. In Section 2.3, it is shown that, for the DM-2 model, the degree distribution of G_t follows asymptotically a power-law with exponent that can become arbitrary large depending on the value of parameter α.



(a) (b)

Figure 2.7. Two realizations of the DM-2 model: (a) $\alpha = 0.5$; (b) $\alpha = 4$.

Bianconi-Barabasi (BB) model [BB01]. In this model, each node s is associated at the time of its birth with a random "fitness" coefficient, η_s, that remains constant. Fitness coefficients are drawn independently from a probability distribution $f(x)$. The preferential attachment rule may be modified in various ways to incorporate the fitness of nodes. The case considered in [BB01] is the one where fitness has a *multiplicative* effect on the degree, as follows:

$$\mathbb{P}_{t+1}(s) = \frac{\eta_s d_t(s)}{\sum_{j=1}^{t} \eta_j d_t(j)} \qquad (2.6)$$

for $1 \leq s \leq t$. Ergün and Rodgers [ER02] have studied alternative ways of incorporating fitness coefficients, *e.g.,* the case where fitness has an *additive* effect on the degree. A realization of BB model is depicted in Figure 2.8.

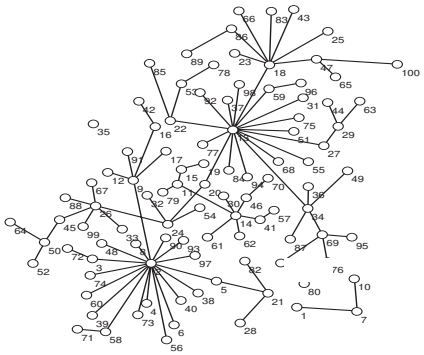


Figure 2.8. A realization of BB model.

In Section 2.3, it will be seen that degree distribution of this model follows a power-law with a logarithmic correction.

In the dynamic models discussed so far, each time-step adds a new node and one or more new edges—all incident on the new node. However, in real networks new edges might also be added between existing nodes (*e.g.*, *hyperlinks* are often added from an existing web page to another existing web page). The following model incorporates this idea:

Krapivsky-Redner-2 (KR-2) model [KR03]. In the KR-2 model, beginning with an isolated node, the growth in the network happens due to two distinct processes:

- With probability p, a new node and a new arc emanating from this node are added. The head of the new arc is chosen according to a shifted linear preferential attachment rule based on *in-degree*:

$$\mathbb{P}_{t+1}(s) = \frac{\lambda + d_t^-(s)}{\sum_{j \in V_t} \lambda + d_t^-(j)} \qquad (2.7)$$

- With probability $q = 1 - p$, a new arc is created between two existing nodes. The *tail* of this new arc is chosen based on *out-degree* while the head is chosen based on *in-degree*; more specifically, the probability that an arc is added between an existing node s_1 and another existing node s_2 is given by

$$\mathbb{P}_{t+1}(s_1, s_2) = \frac{\left[\lambda + d_t^-(s_2)\right]\left[\mu + d_t^+(s_1)\right]}{\sum_{i,j \in V_t} \left[\lambda + d_t^-(j)\right]\left[\mu + d_t^+(i)\right]} \qquad (2.8)$$

where $\lambda > 0$ and $\mu > -1$.

The preferential attachment rule given by the above equation is called *bilinear* [KR03]. A realization of the KR-2 model is shown in Figure 2.9.

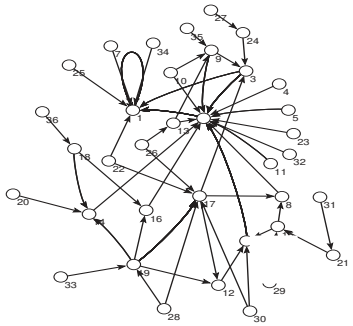


Figure 2.9. A realization of the KR-2 model.

In addition to the models given above, numerous other birth-only, preferential attachment models have been proposed [DM01a, YJBT01, ACL02, CF03, BBCR03, BBBC03, Vaz03, BO04, FFV04, BBPV04a, BBPV05]. All these models have been shown to generate graphs with power-law degree distribution.

Birth-only copying models

Kumar-et-al. (KKRS) model [KKRS00]. This model specifies a directed random graph and is parameterized by a *copy factor* $p \in (0,1)$ and a constant out-degree $d \geq 1$. In each time-step $t+1$, one new node and d new arcs having this node as tail, are added. The heads of the new arcs are chosen as follows: First, a node s—the *prototype*—is selected uniformly at random from V_t. Next, with probability p, the head of i^{th} new arc is chosen uniformly at random from V_t, whereas with probability $1-p$ this head is taken to be the head of the i^{th} arc leaving the prototype s.

Kumar *et al.* [KKRS00] have defined another version of the copying model, called *exponential growth copying*. This version has four parameters: (i) the growth factor p; (ii) the self-loop factor α; (iii) the tail copy factor α'; and (iv) the out-degree factor d. Beginning with a single node which has α directed self-loops, the number of nodes added during the $(t+1)^{th}$ step is pn_t. Thus, the graph G_{t+1} will have $(1+p)^{t+1}$ nodes. The number of arcs added in each step is $(d+\alpha)pn_t$. Each new nodes is born with α self-loops; this accounts for αpn_t new arcs. The remaining dpn_t new arcs are attached according to the following rules: For each new node $u \in V_t$, and each arc directed into u in the graph G_t, a new arc directed into u is added during $(t+1)^{th}$ step. The tail of each new arc is chosen as follows: (a) with probability $1-\alpha'$ it is

chosen uniformly at random from the pn_t new nodes; and (b) with probability α' it is chosen at random from the pn_{t-1} nodes created in the previous step according to a linear preferential attachment rule based on out-degree.

MFCS model [Ald04]. Aldous [Ald04] proposed a *metric copying framework* for defining dynamic random graph models. In this framework nodes are assumed to be points in a metric space, *i.e*., there is some real-valued distance $r(u,w)$ defined between any two nodes u,w. The graph G_t is constructed by adding one new node t to the graph G_t and (1) for each arc ij in the graph G_{t-1}, a copied arc tj is created with probability $p(r(t,i))$; (2) for each node i $(1 \le i < t)$, a new arc ti is created with probability $p(r(t,i))$. Here, $p:[0,\infty) \to [0,1]$ is assumed to be a real-valued function. Aldous [Ald04], studied the asymptotic properties of a particular metric copying model—the *mean field simple copying* (MFCS) model—in which distances of the metric space are assumed to be random numbers drawn from an exponential distribution and the function p is given by: $p(x) = \min(1, \alpha\lambda e^{-\lambda x}),\ 0 \le x < \infty$, where α and λ are parameters of the model. Aldous found that MFCS model displays some of the properties of web-like networks: a power-law degree distribution, an average distance that grows logarithmically with the order of the graph, and a high density of certain subgraphs such as complete bipartite graphs.

Birth-death models

Only a few birth-death dynamic models have been studied so far. To the best of our knowledge, all these models are based on preferential attachment. Three such models are shown next:

Dorogovtsev-Mendes-3 (DM-3) model [DM00c]. In this model, a new node and a new edge are added in each step according to a linear preferential attachment rule (as in the BA model). In addition, ε existing edges chosen uniformly at random are deleted.

Cooper-Frieze-Vera, *Chung-Lu* (CFV-CL) model [CFV04, CL04]. This model was introduced independently by Cooper *et al*. [CFV04] and Chung and Lu [CL04]. It combines the birth of nodes and edges with a uniform deletion of both nodes and edges as follows:

- With probability p_1, a new node and ε new edges incident on it are added. The other ends of the new edges are chosen at random from existing nodes based on a linear preferential attachment rule.
- With probability p_2, ε new edges are added. The probability that a new edge is added between two existing nodes is proportional to the product of their degrees.
- With probability p_3, a new node chosen uniformly at random and all edges incident on it are deleted.
- With probability $p_4 = 1 - p_1 - p_2 - p_3$, ε edges chosen uniformly at random are deleted.

Flaxman-et-al (FFV) model [FFFV04]. This model combines addition of nodes and edges with an "adversarial" deletion of nodes: In each time-step t, a new node and ε new edges incident on it are added. The other ends of the new edges are chosen based on linear preferential attachment. After the edges are added, an "adversary" may delete up to δt nodes, where δ is a constant.

Deo-Cami (DC) model [DC05]. In the DC model, the birth of nodes and edges is combined with a *preferential deletion* of nodes that favors the deletion of small-degree nodes. The motivation for this type of deletion comes from the Web and the Internet where it has been

observed that small-degree nodes die more frequently that high-degree ones [BBKT04, VPV02]. Beginning with a single node with a self-loop, in each step of this model either one of the following two processes can happen:

- With probability p, a new node and a new edge incident on it are added. The other end of the new edge is chosen based on preferential attachment.
- With probability $q = 1 - p$, an existing node and all the edges incident on it are deleted. The node deleted during $(t+1)^{th}$ step is chosen based on the following distribution:

$$\mathbb{P}_{t+1}(u) = \frac{n_t - d_t(u)}{n_t^2 - 2m_t} \qquad (2.9)$$

Before passing to the techniques for analyzing the dynamic models, it would be instructive to look at the nature of probability spaces associated with dynamic models. For concreteness, let us focus on the LCD model:

The procedure that generates an instance of the random graph G_t for the LCD model, described earlier in this section, suggests the following *inductive* definition of the probability space $\mathcal{G}_t$: The space $\mathcal{G}_1$ contains a single element—the graph G_1^1 consisting of a single node and a self-loop. The probability measure $\mathbb{P}_1$ of the space $\mathcal{G}_1$ assigns a probability of 1 to the graph G_1^1. Given the space $\mathcal{G}_1$, the two elements G_2^1 and G_2^2 of the space $\mathcal{G}_2$ are obtained by adding a new node to the graph G_1^1 together with a new edge which either joins nodes 1 and 2 (with probability 2/3) or is a self-loop of node 2 (with probability 1/3). Hence the probability measure $\mathbb{P}_2$ is defined by $\mathbb{P}_2\left(G_2^1\right) = 2/3$, $\mathbb{P}_2\left(G_2^2\right) = 1/3$. It is now easy to see that $|\mathcal{G}_t| = |\mathcal{G}_{t-1}| \, t$, therefore the probability space $\mathcal{G}_t$ has $t!$ elements. Given the definition of the probability measure $\mathbb{P}_{t-1}$ on $\mathcal{G}_{t-1}$, the probability measure $\mathbb{P}_t$ is obtained by applying:

$$\mathbb{P}_t\left(G_t^k\right) = \sum_{i=1}^{(t-1)!} \mathbb{P}\left(G_t^k \mid G_{t-1}^i\right) \mathbb{P}\left(G_{t-1}^i\right), \quad k = 1,\ldots,t! \qquad (2.10)$$

Typically, one is concerned with asymptotic properties of the space $\mathcal{G}_t$, as $t \to \infty$. The above inductive definition of $\mathcal{G}_t$ suggests that to study the probability space $\mathcal{G}_t$ one should take into account all the probability spaces $\mathcal{G}_k$ for $k < t$. In other words, it is to be expected that in order to derive results about $\mathcal{G}_t$, one should resort to the use of *difference equations*. In fact, such equations arise on a regular basis during the analysis of dynamic random graphs, as shown in the next section.

2.3. Some Techniques for Analyzing Dynamic Random Graphs

The techniques for analyzing dynamic random graph models can be broadly divided into two groups: (1) *heuristic*; and (2) *rigorous*. The techniques belonging to the first group allow one to *quickly* obtain approximate result. However, such results have to be constantly checked by other methods or numerical simulations. The techniques in the second group are mathematically rigorous and thus produce exact results. However applying them requires considerably more effort than applying the heuristic techniques. This section is devoted to the heuristic techniques for analyzing the dynamic random graph models.

Degree distribution

We begin with a detailed discussion of three heuristic techniques that have been widely used to analyze the degree distribution of several models: (A) the *continuous* method; (B) the *master equation* method; and (C) the *rate equation* method.

(*A*) *The continuous method:* This method has been used by several authors such as Barabasi *et al*. [BAJ99, BB01] and Dorogovtsev and Mendes [DM01B]. There are two main steps involved:

Step1: Let the random variable $d_t(s)$ denote the degree in G_t of an arbitrary node s, $1 \le s \le t$. Initially, an expression for $d_t(s)$ in terms of t, is obtained by deriving and solving a differential equation. One of the key assumptions of the continuous method is that the degree $d_t(s)$ may be approximated by its expected value. This approximation—known as the *mean-field* approach—means that for each node s, the pdf $\mathbb{P}_t(s,k)$ of the random variable $d_t(s)$ is highly concentrated around its mean *i.e.,* $\mathbb{P}_t(s,k) \approx \delta\left(k - \mathbb{E}\left(d_t(s)\right)\right)$. In addition, it is often assumed that it is safe to work in the continuous domain (as the length of the time-step tends to zero).

The mean-field assumption implies that

$$\Delta\left(d_t(s)\right) = d_{t+1}(s) - d_t(s) \approx \mathbb{E}\left(d_{t+1}(s)\right) - \mathbb{E}\left(d_t(s)\right) = \mathbb{E}\left(\Delta\left(d_t(s)\right)\right)$$

i.e., the change $\Delta\left(d_t(s)\right)$ may also approximated by its expected value. Note that $\Delta\left(d_t(s)\right)$ denotes the number of edges that link with node s during the $(t+1)^{th}$ step. Assuming that edges are added independently, it follows that $\Delta\left(d_t(s)\right)$ is a binomial random variable with parameters ε—the number of edges added in each step—and $p = \mathbb{P}_{t+1}(s)$. Thus, $\mathbb{E}\left(\Delta\left(d_t(s)\right)\right) = \varepsilon\mathbb{P}_{t+1}(s)$. Switching to the continuous domain, we get the following differential equation:

$$\frac{\partial d_t(s)}{\partial t} = \varepsilon\mathbb{P}_{t+1}(s). \qquad (2.11)$$

Step2: Having found a solution of Equation (2.11), the following approach is proposed in [BAJ99] to obtain $\mathbb{P}_t(k)$: first, find the cumulative density function $F_t(k)$ of $d_t(s)$ by using the formula $F_t(k) = \mathbb{P}\left(d_t(s) < k\right)$ and then derive $\mathbb{P}_t(k)$ by differentiating $F_t(k)$ with respect to k.

This approach is not very appealing, since it requires the differentiation of a discrete cumulative density function! A better method, proposed in [DM01b], is to use the *law of total probability* as follows:

$$\begin{aligned}\mathbb{P}_t(k) &= \int_0^t \mathbb{P}\big(d_t(v) = k \mid v = s\big)\mathbb{P}(v = s)\,ds \\ &= \frac{1}{t}\int_0^t \mathbb{P}\big(d_t(s) = k\big)\,ds \\ &= \frac{1}{t}\int_0^t \delta(k - d_t(s))ds,\end{aligned} \qquad (2.12)$$

where the last equality follows from mean-field assumption.

Next, we show how the continuous method can be applied to obtain the degree distribution for three dynamic random graph models: (i) BA model; (ii) BB model; and (iii) DM-1 model.

(i) Degree distribution in the BA model: The following derivation is similar to that in [BAJ99].

Step 1: Recall that the preferential attachment rule in the BA model is given by

$$\mathbb{P}_{t+1}(s) = \frac{d_t(s)}{\sum_{j\in V_t} d_t(j)} = \frac{d_t(s)}{2\varepsilon t}. \qquad (2.13)$$

Therefore, substituting for $\mathbb{P}_{t+1}(s)$ in Equation (2.11) we get:

$$\frac{\partial d_t(s)}{\partial t} = \frac{d_t(s)}{2t}. \qquad (2.14)$$

The solution of the *linear* differential Equation (2.14) with the boundary condition $d_s(s) = \varepsilon$ (each node is born with degree ε) is:

$$d_t(s) = \varepsilon\left(t/s\right)^{1/2} \quad \text{for } t \geq s. \qquad (2.15)$$

Figure 2.10 shows that there is a good agreement between the simulation data and the prediction of Equation (2.15). This figure depicts the evolution of the degree of nodes born during the time-steps 5 and 30. The lines correspond to the analytical prediction of Equation (2.15) while the data

points to the simulation of the BA model. The simulation results are averaged over 30 realizations of the BA model with $t = 100,000$.

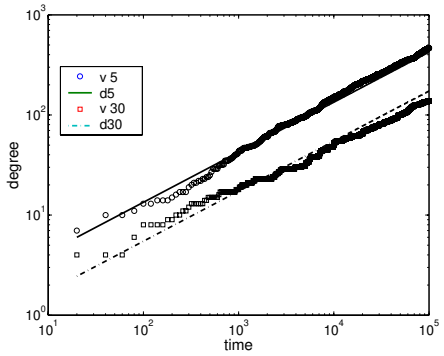


Figure 2.10. Evolution of degree in the BA model.

Note that the average degree of the graph G_t is asymptotically given by:

$$d(G_t) = \frac{2\varepsilon t}{\varepsilon_0 + t} \to 2\varepsilon \quad \text{as} \quad t \to \infty.$$

Step 2: For simplicity we consider the case $\varepsilon_0 = 1$.

$$\begin{aligned} \mathbb{P}_t(k) &= \frac{1}{t+1}\int_0^t \delta\left[k - \varepsilon(t/s)^{1/2}\right] ds \\ &= -\frac{1}{t+1}\left[\frac{1}{\frac{\partial d_t(s)}{\partial s}\Big|_{s=\varepsilon^2 t/k^2}}\right] \\ &= \frac{t}{t+1}\frac{2\varepsilon^2}{k^3} \to \frac{2\varepsilon^2}{k^3} \quad \text{as} \quad t \to \infty. \end{aligned} \tag{2.16}$$

Thus, the linear preferential attachment rule, leads to a power-law asymptotic degree distribution with exponent $\gamma = 3$. Figure 2.11 compares the result of Equation (2.16) with the data obtained by simulating the BA model. The three data sets correspond, left to right, to the cases $\varepsilon = 1, \varepsilon = 3,$ and $\varepsilon = 5$ while the line is the plot of the function k^{-3}.

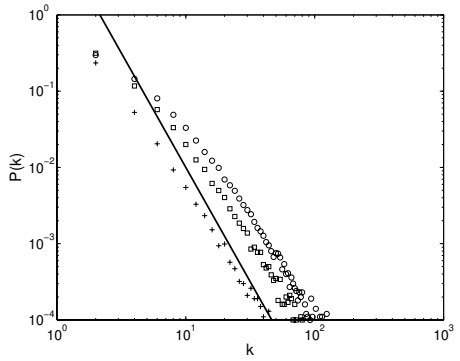


Figure 2.11. Log-log plot of the degree distribution in the BA model.

(ii) The degree distribution in the BB model: The following derivation is taken from [BB01].

Step 1: The differential equation becomes

$$\frac{\partial d_t(s)}{\partial t} = \varepsilon \mathbb{P}_{t+1}(s) = \varepsilon \frac{\eta_s d_t(s)}{\sum_i \eta_i d_t(i)}. \qquad (2.17)$$

In the continuous limit, the normalization coefficient $A_t = \sum_i \eta_i d_t(i)$ may be written as $A_t = \int_0^t \eta_s d_t(s) ds$. Note that since $0 < \eta_s < 1$, it follows that $A_t < \int_0^t d_t(s) ds = 2t$. Conjecturing that A_t grows linearly with t as $A_t = Ct$ and substituting for A_t in Equation (2.17) we obtain:

$$\frac{\partial d_t(s)}{\partial t} = \varepsilon \frac{\eta_s d_t(s)}{Ct} \qquad (2.18)$$

The solution of Equation (2.18) with boundary condition $d_s(s) = \varepsilon$ is

$$d_t(s) = \varepsilon \left(t/s\right)^{\frac{\eta_s}{C}}. \qquad (2.19)$$

By comparing equations (2.15) and (2.19), it may be seen that they differ only in that the exponent of t/s in the latter is a function of fitness, whereas in the former it is a constant. This *multi-scaling* phenomenon arises because of the introduction of *competition* in the network. If

the pdf $f(x)$ of the fitness coefficient is known, the constant C can be found from the following equation:

$$\int_0^{\eta_{max}} \left[x d_t(s) ds\right] f(x) dx = \mathbb{E}\left(A_t\right) = Ct$$

which by using Equation (2.19) becomes

$$\int_0^{\eta_{max}} \frac{x}{C-x} f(x) dx = 1 \qquad (2.20)$$

Step 2: Let's obtain $\mathbb{P}_k(\eta)$ in the case $\varepsilon_0 = \varepsilon = 1$:

$$\begin{aligned} P_k(\eta) &= \frac{1}{t+1} \int_0^t \delta(k - \varepsilon(t/s)^{\frac{\eta_s}{C}}) ds \\ &= -\frac{1}{t+1} \left[\frac{1}{\frac{\partial d_t(s)}{\partial s}\Big|_{s=tk^{-C/\eta}}} \right] \\ &\to \frac{C}{\eta} \frac{1}{k^{1+C/\eta}} \quad \text{as} \quad t \to \infty. \end{aligned}$$

Consider, as examples, two special cases of the fitness pdf $f(x)$: (i) If $f(x) = \delta(x-1)$ (all nodes have the same fitness 1), than the fitness model reduces to the previous BA model, as expected; (ii) In the case of a uniform distribution

$$f(x) = \begin{cases} 1 & \text{if} \quad 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases}$$

it may be determined, by solving Equation (2.20), that $C^* = 1.225$. Then, $\mathbb{P}(k)$ is obtained by

$$\mathbb{P}(k) = \int_0^1 \frac{C^*}{x} \frac{1}{k^{1+C^*/x}} dx \approx \frac{1}{(\log k) k^{1+C^*}}. \qquad (2.21)$$

Thus, in the case of a uniform fitness distribution, the BB model has a power-law distribution with exponent $\gamma = 2.225$ and a logarithmic correction. Figure 2.12 shows that there is a good match between the simulation data and the analytical prediction given by Equation (2.21).

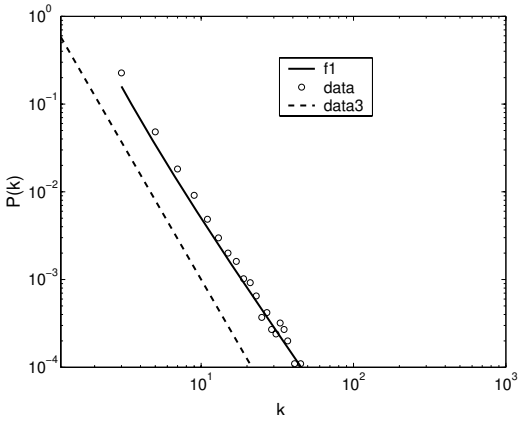


Figure 2.12. Log-log plot of the degree distribution in the BB model.

(iii) The degree distribution in the DM-2 model: The following derivation appears in [DM01b].

Step 1: The differential equation becomes

$$\frac{\partial d_t(s)}{\partial t} = \frac{d_t(s)(t-s)^{-\alpha}}{\sum_i d_t(i)(t-i)^{-\alpha}} \quad \text{for} \quad \alpha \geq 0. \qquad (2.22)$$

The boundary condition is $d_s(s) = 1$, because each node is born with degree one. This equation is difficult to solve for an arbitrary α and thus we can't proceed as in the previous two cases.

It may be shown (see [DM01b]) that if $d_t(s) \sim (s/t)^{\beta}$, then the asymptotic degree distribution follows a power-law with exponent γ, where

$$\gamma(\beta - 1) = 1. \qquad (2.23)$$

Based on the just stated result, the solution of Equation (2.22) is searched in the form

$$d_t(s) = f(s/t). \qquad (2.24)$$

Now, let $\xi = s/t$. Then Equation (2.22) becomes

$$-\xi(1-\xi)^{\alpha} \frac{d \ln f(\xi)}{d\xi} = \frac{1}{\int_0^1 f(\zeta)(1-\zeta)^{-\alpha} d\zeta}, \qquad (2.25)$$

and $f(1) = 1$. To proceed, we solve for β in

$$-\xi(1-\xi)^{\alpha}\frac{d\ln f(\xi)}{d\xi}=\beta, \qquad (2.26)$$

and replace the left-hand side of Equation (2.25) with the obtained value of β. By doing so and then solving for α it is found that

$$\beta \approx 1/2-(1-\ln 2)\alpha \qquad (2.27)$$

Step 2: As already pointed out, the degree-distribution in this case must follow a power-law whose exponent may be obtained directly plugging the value of β into Equation (2.23) to get

$$\gamma \approx 3+4(1-\ln 2)\alpha. \qquad (2.28)$$

Thus, the introduction of aging of nodes, changes the exponent of the power-law of the degree distribution, and can lead to very large exponents.

(B) The master equation method

This method was introduced by Dorogovtsev *et al*. [DMS00]. In contrast with the continuous method, here no assumption is made about the distribution $\mathbb{P}_t(s,k)$. Furthermore, the calculations are done mostly in the discrete domain and thus the results are more accurate. The method can be outlined as follows:

Step 1: First, determine the degree distribution $P_t(s,k)$ of an arbitrary node s, *i.e.*, the probability that the degree of node s in the graph G_t is k.

Step 2: Then, the degree distribution $\mathbb{P}_t(k)$ of the graph G_t is determined as follows:

$$\mathbb{P}_t(k)=\frac{1}{t}\sum_{s=1}^{t}\mathbb{P}_t(s,k) \qquad (2.29)$$

The derivations using the master equation method are quite lengthy, and therefore we show the application of this method only for the case of DM-1 model.

(i) The degree distribution in the DM-1 model: The following derivation is taken from [DMS00].

Step 1: The normalization constant in Equation (2.3) can be found as follows

$$\sum_{j=1}^{t}\big(\lambda + d_t^-(j)\big) = t\lambda + \sum_{j=1}^{t} d_t^-(j) = t\lambda + \varepsilon t = (1+a)\varepsilon t, \qquad (2.30)$$

where $a = \lambda/\varepsilon$. The number of arcs that link into node s during each time-step, is a binomial random variable with parameters ε and $\mathbb{P}_t(s) = W_s$. Hence, the probability that the node s will receive exactly l new incoming arcs out of the total ε new arcs added during each step, is

$$B_s(\varepsilon, l) = \binom{\varepsilon}{l}(W_s)^l(1-W_s)^{\varepsilon-l}. \qquad (2.31)$$

Now, $\mathbb{P}_{t+1}(s,k)$ may be found by conditioning on the in-degree of s at time t:

$$\begin{aligned}
\mathbb{P}_{t+1}(s,k) &= \sum_{l=0}^{m} B_s(\varepsilon,l)\mathbb{P}_t(s,k-l) \\
&= \sum_{l=0}^{\varepsilon}\binom{\varepsilon}{l}(W_s)^l(1-W_s)^{\varepsilon-l}\mathbb{P}_t(s,k-l) \qquad (2.32)\\
&= \sum_{l=0}^{\varepsilon}\binom{\varepsilon}{l}\left[\frac{k-l+a\varepsilon}{(1+a)\varepsilon t}\right]^l\left[1-\frac{k-l+a\varepsilon}{(1+a)\varepsilon t}\right]^{\varepsilon-l}\mathbb{P}_t(s,k-l).
\end{aligned}$$

Note that $\mathbb{P}_s(s,k) = \delta_{k0}$ since nodes are born with in-degree zero.

Step 2: We sum both sides of Equation (2.32) for $s = 1$ to t. Denoting the left hand side summation by LHS and the right hand side by RHS we have

$$\begin{aligned}
LHS &= \sum_{s=1}^{t}\mathbb{P}_{t+1}(s,k) \\
&= \sum_{s=1}^{t+1}\mathbb{P}_{t+1}(s,k) - \mathbb{P}_{t+1}(t+1,k) \\
&= (t+1)\mathbb{P}_{t+1}(k) - \mathbb{P}_{t+1}(t+1,k) \\
&= (t+1)P_{t+1}(k) - \delta_{k0}
\end{aligned}$$

The summation RHS is a little more complicated. By using the expansion

$$(1-x)^{\varepsilon} = \sum_{k=0}^{\varepsilon}(-1)^k \binom{\varepsilon}{k} x^k \qquad (2.33)$$

and after some algebra (see [DMS00]) we get the following expression for RHS:

$$\left[t - \frac{k+a\varepsilon}{(1+a)}\right]\mathbb{P}_t(k) + \frac{k-1+a\varepsilon}{1+a}\mathbb{P}_t(k-1) + O(\mathbb{P}/t)$$

Now, equating LHS and RHS we obtain the following equation:

$$(t+1)\mathbb{P}_{t+1}(k) - t\mathbb{P}_t(k) + \frac{k+a\varepsilon}{(1+a)}\mathbb{P}_t(k) - \frac{k-1+a\varepsilon}{1+a}\mathbb{P}_t(k-1) = \delta_{k0} + O(\mathbb{P}/t)$$

or,

$$\Delta\big(t\mathbb{P}_t(k)\big) + \frac{k+a\varepsilon}{(1+a)}\mathbb{P}_t(k) - \frac{k-1+a\varepsilon}{1+a}\mathbb{P}_t(k-1) = \delta_{k0} + O(\mathbb{P}/t)$$

Switching to the continuous domain yields:

$$\frac{\partial}{\partial t}[t\mathbb{P}_k(t)] + \frac{k+a\varepsilon}{(1+a)}\mathbb{P}_t(k) - \frac{k-1+a\varepsilon}{1+a}\mathbb{P}_t(k-1) = \delta_{k0} + O(\mathbb{P}/t)$$

or

$$t\frac{\partial\mathbb{P}_t(k)}{\partial t} + \mathbb{P}_t(k) + \frac{k+a\varepsilon}{(1+a)}\mathbb{P}_t(k) - \frac{k-1+a\varepsilon}{1+a}\mathbb{P}_t(k-1) = \delta_{k0} + O(\mathbb{P}/t)$$

Finally, assuming that the stationary probabilities $\mathbb{P}(k) = \mathbb{P}_{t\to\infty}(k)$ exist, we obtain the difference equation

$$(1+a+k+a\varepsilon)\mathbb{P}(k) - (k-1+a\varepsilon)\mathbb{P}(k-1) = (1+a)\delta_{k0} \quad \text{for} \quad k \geq 0. \quad (2.34)$$

Dorogovtsev *et al*. [DMS00] have used the method of *generating functions* to solve Equation (2.34). Here, we apply the formula given in Appendix B; indeed, the solution of Equation (2.34) can be obtained directly in terms of gamma functions as:

$$\mathbb{P}(k) = (1+a)\frac{\Gamma[(\varepsilon+1)a+1]}{\Gamma(\varepsilon a)}\frac{\Gamma(k+\varepsilon a)}{\Gamma[k+2+(\varepsilon+1)a]}$$

In the limit $k \to \infty$ we get

$$\mathbb{P}(k) \sim k^{-(2+a)} \qquad (2.35)$$

i.e., the degree distribution follows a power-law with exponent $\gamma = 2 + a = 2 + \lambda/\varepsilon$.

If $a = 1$, which implies $\lambda = \varepsilon$ (the BA model), we get:

$$\mathbb{P}(k) = \frac{\Gamma[\varepsilon + 2]}{\Gamma(\varepsilon)} \frac{\Gamma(k + \varepsilon)}{\Gamma[k + \varepsilon + 3]} = \frac{\varepsilon(\varepsilon + 1)}{(k + \varepsilon)(k + \varepsilon + 1)(k + \varepsilon + 2)}.$$

(C) The rate equation method

This method was introduced by Krapivsky *et al.* [KRR01, KR03]. We begin with an overview of how this technique may be used to obtain degree distribution.

Let $N_{t,k}$ be the number of nodes of degree k in the graph G_t. The method proceeds in steps:

Step 1: Derive a differential equation that relates $\mathbb{E}\left(N_{t,k}\right)$ to its rate of change. This is called the *rate equation*; this equation is actually a family of differential equations, one for each $k \geq 0$.

Step 2: Solve the first few cases of the rate equation *e.g.,* for $k = 1,2$. Often, it turns out that the rate equation is a first-order linear differential equation and thus a closed form solution may be easily obtained. Furthermore, for every fixed k, the solution has the form $\mathbb{E}\left(N_{t,k}\right) = \mathbb{P}(k)t$ where $\mathbb{P}(k) = \mathbb{P}_{t\to\infty}(k)$ denotes the asymptotic degree distribution.

Step 3: Substitute $\mathbb{P}(k)t$ for $\mathbb{E}\left(N_{t,k}\right)$ in the rate equation. This substitution yields a first-order linear difference equation on $\mathbb{P}(k)$. Such equations can also be easily solved (Appendix B); thus a closed-form expression for the degree distribution $\mathbb{P}(k)$ is obtained.

As an example, we show next the derivation of the degree distribution in the KR-1 model.

(i) Degree distribution in the KR-1 model: The following derivations are taken from [KR01].

Step 1: To get the rate equation, consider how $\mathbb{E}\left(N_{t,k}\right)$ changes between time-steps t and $(t+1)$. We can write:

$$\Delta[\mathbb{E}\left(N_{t,k}\right)] = \mathbb{E}\left(N_{t,(k-1)\to k}\right) - \mathbb{E}\left(N_{t,k\to(k+1)}\right)$$

where $N_{t,(k-1)\to k}$ is the number of nodes whose degree changes from $(k-1)$ to k, and $N_{t,k\to(k+1)}$ is the number of nodes whose degree changes from k to $k+1$, during the $(t+1)^{th}$ step. Note that $N_{t,(k-1)\to k}$ and $N_{t,k\to(k+1)}$ are *Bernoulli* random variables with parameters

$$\frac{\mathbb{E}\left(N_{t,k-1}\right)(k-1)^{\alpha}}{\sum_j j^{\alpha}\mathbb{E}\left(N_{t,j}\right)} \quad \text{and} \quad \frac{\mathbb{E}\left(N_{t,k}\right)k^{\alpha}}{\sum_j j^{\alpha}\mathbb{E}\left(N_{t,j}\right)}$$

respectively. Therefore denoting the normalization constant by $M_{t,\alpha} = \sum_j j^{\alpha}\mathbb{E}\left(N_{t,j}\right)$ we get

$$\mathbb{E}\left(N_{t,(k-1)\to k}\right) = \frac{(k-1)^{\alpha}}{M_{t,\alpha}}\mathbb{E}\left(N_{t,k-1}\right), \quad \mathbb{E}\left(N_{t,k\to(k+1)}\right) = \frac{k^{\alpha}}{M_{t,\alpha}}\mathbb{E}\left(N_{t,k}\right)$$

Hence, the rate equation can be written as

$$\frac{\partial\mathbb{E}\left(N_{t,k}\right)}{\partial t} = \frac{1}{M_{t,\alpha}}\left[(k-1)^{\alpha}\,\mathbb{E}\left(N_{t,k-1}\right) - k^{\alpha}\mathbb{E}\left(N_{t,k}\right)\right] + \delta_{k1}\,.$$

The last term accounts for the addition of a new node with degree one, in each step.

Step 2: Let us solve the rate equation for $k = 1,2$. First, note that $M_{t,0} = \sum_j \mathbb{E}\left(N_{t,j}\right)$ is the expected total number of nodes of the graph G_t. Hence, the rate of change of $M_{t,0}$ is 1 (because one new node is added in each step). It follows that: $M_{t,0} = t + M_{0,0} = t$. Similarly,

we have that $M_{t,1} = \sum_j j\mathbb{E}\left(N_{t,j}\right) = \sum_v d_t(v) = 2m_t$. The rate of change of $M_{t,1}$ is 2 (one edge is added in each step), and as a result $M_{t,1} = 2t + M_{0,1} = 2t$. Next, we consider three separate cases: (a) $\alpha = 1$ (the *linear* case); (b) $\alpha < 1$ (the *sub-linear* case), and (c) $\alpha > 1$ (the *super-linear* case).

(a) The linear case $\alpha = 1$: Replacing for $M_{t,1}$ in the rate equation we get:

- For $k = 1$ the rate equation becomes

$$\frac{\partial \mathbb{E}\left(N_{t,1}\right)}{\partial t} + \frac{\mathbb{E}\left(N_{t,1}\right)}{2t} = 1.$$

  Its solution is $\mathbb{E}\left(N_{t,1}\right) = 2t/3 + C/\sqrt{t} \to 2t/3$, *i.e.*, on average, two-thirds of the nodes will have degree 1 as $t \to \infty$.

- For $k = 2$ the rate equation becomes

$$\frac{\partial \mathbb{E}\left(N_{t,2}\right)}{\partial t} + \frac{\mathbb{E}\left(N_{t,2}\right)}{t} = \frac{1}{3}$$

  Its solution is asymptotically $\mathbb{E}\left(N_{t,2}\right) \to t/6$. Based on the solutions for $k = 1,2$, it is assumed that $\mathbb{E}\left(N_{t,k}\right)$ is linear in t: $\mathbb{E}\left(N_{t,k}\right) = t\mathbb{P}(k)$.

Step 3: Substituting for $\mathbb{E}\left(N_{t,k}\right)$ in the rate equation we get the homogeneous first-order linear difference equation

$$\mathbb{P}(k+1) = \frac{k}{k+3}\mathbb{P}(k), \quad \text{for} \quad k \geq 1.$$

with initial condition $\mathbb{P}(1) = 2/3$, whose solution is

$$\mathbb{P}(k) = \frac{4}{k(k+1)(k+2)}.$$

Note that this is essentially the same result as the one obtained for the BA model using either the *continuous* or *master equation* method.

(b) The sub-linear case $\alpha < 1$: First, note that

$$M_{t,0} = \sum_j \mathbb{E}\left(N_{t,j}\right) \le \sum_j j^\alpha \mathbb{E}\left(N_{t,j}\right) = M_{t,\alpha}$$
$$M_{t,1} = \sum_j j\mathbb{E}\left(N_{t,j}\right) \ge \sum_j j^\alpha \mathbb{E}\left(N_{t,j}\right) = M_{t,\alpha}$$

Therefore for any $\alpha < 1$:

$$t = M_{t,0} \le M_{t,\alpha} \le M_{t,1} = 2t$$

In the limit $t \to \infty$ we can write $M_{t,\alpha} = \mu(\alpha)t$ where the function $\mu(\alpha)$ has yet to be determined (we only know that $1 \le \mu(\alpha) \le 2$ and $\mu(0) = 1$, $\mu(1) = 2$). Now, repeat the steps followed in the linear case. Replace $M_{t,\alpha} = \mu t$ in the rate equation and solve the first few cases:

- For $k = 1$ the rate equation becomes:

$$\frac{\partial \mathbb{E}\left(N_{t,1}\right)}{\partial t} + \frac{\mathbb{E}\left(N_{t,1}\right)}{\mu t} = 1.$$

  The asymptotical solution of the differential equation above is $N_{t,1} = \frac{\mu}{\mu+1} t$.

- For $k = 2$ the rate equation is:

$$\frac{\partial \mathbb{E}\left(N_{t,2}\right)}{\partial t} + \frac{2^\alpha \mathbb{E}\left(N_{t,2}\right)}{\mu t} = \frac{1}{\mu + 1}.$$

  and its solution is $\mathbb{E}\left(N_{t,2}\right) = \frac{\mu}{(\mu+1)(\mu+2^\alpha)} t$.

Step 3: Generalizing, we let $\mathbb{E}\left(N_{t,k}\right) = t\mathbb{P}(k)$. After substituting in the rate equation, the following homogeneous first-order linear difference equation is obtained:

$$\frac{\mu + k^{\alpha}}{\mu}\mathbb{P}(k) - \frac{(k-1)^{\alpha}}{\mu}\mathbb{P}(k-1) = 0.$$

Its solution is immediately obtained as:

$$\mathbb{P}(k) = \frac{\mu}{k^{\alpha}} \frac{1}{\left(1+\frac{\mu}{1^{\alpha}}\right)\left(1+\frac{\mu}{2^{\alpha}}\right)\cdots\left(1+\frac{\mu}{k^{\alpha}}\right)}.$$

In [KR01] it is argued that the asymptotic behavior of $\mathbb{P}(k)$ in this case does not follow a power-law but rather a stretched exponential. Our numerical simulations shown in Figure 2.13 are in good agreement with this conclusion.

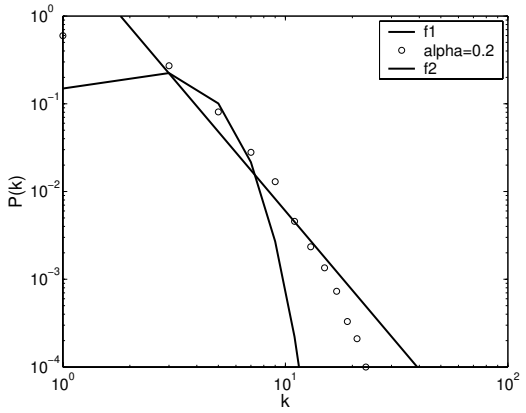


Figure 2.13. Log-log plot of the degree distribution in the KR-1 model with $\alpha = 0.2$.

(c) The super-linear case $\alpha > 1$. In this case, a different approach may be followed to analyze the degree distribution:

The probability that each of the first t nodes will be a neighbor of the initial node is:

$$\frac{t^{\alpha}}{t^{\alpha} + t}$$

Hence, the probability that this pattern continues indefinitely is

$$\mathbb{P} = \prod_{t=1}^{\infty} \frac{t^{\alpha}}{t^{\alpha} + t} = \prod_{t=1}^{\infty} \frac{1}{1 + t^{1-\alpha}}$$

It may be shown that: (a) for $\alpha \leq 2$, $\mathbb{P} = 0$; and (b) for $\alpha > 2$, $\mathbb{P} > 0$. Thus, for $\alpha > 2$, all but a finite number of nodes are connected to a single node. This phenomenon is called "*gelation*" and the central node is called the "*gel*" (see Figure 2.6(b)).

To determine the degree distribution for any $1 < \alpha < 2$, the asymptotic form of $M_{t,\alpha}$ is needed. Skipping the details (see [KR01]), the result is that for $1 < \alpha < 2$, the degree distribution changes an infinite number of times as follows: for $\frac{\varepsilon+1}{\varepsilon} < \alpha < \frac{\varepsilon}{\varepsilon-1}$ the number of nodes with degree larger than ε is finite, while for $k \leq \varepsilon$ $\mathbb{E}\left(N_{t,k}\right) \to t^{k-(k-1)\alpha}$.

(ii) The degree distribution in the KR-2 model: Krapivsky and Redner [KR03] have showed, by applying the rate equation as shown above, that for the KR-2 model the in-degree follows a power-law of the form

$$\mathbb{P}^{-}(k) \to 1/k^{2+p\lambda} \qquad (2.36)$$

while the out-degree follows a power-law of the form

$$\mathbb{P}^{+}(k) \to 1/k^{1+q^{-1}+\mu p q^{-1}} \qquad (2.37)$$

Furthermore, the joint in- and out-degree distribution is asymptotically given by:

$$\mathbb{P}(i,j) = C\,\frac{i^{\lambda-1} j^{\mu}}{(i+j)^{2\lambda+1}} \qquad (2.38)$$

This result provides evidence that in- and out-degree distributions are not independent (there is no factorization of the form $i^{\gamma_1} j^{\gamma_2}$).

3. PROPOSED BIRTH-DEATH DYNAMIC RANDOM GRAPH MODEL

As stated in the previous chapter, dynamic random graph models that combine birth and death (addition and deletion of nodes and edges) have been studied much less than the birth-only models: Dorogovtsev and Mendes [DM00b] studied a model which interleaves the addition of nodes and edges with a *uniform* deletion of edges. Chung and Lu [CL04] and Cooper *et al.* [CFV04], independently, studied a dynamic model that combines the addition of nodes and edges with a *uniform* deletion of both nodes and edges. A similar model with a uniform deletion of nodes appeared in [DC05a]. These birth-death models have been found to generate graphs with power-law degree distribution.

In this chapter, we investigate a dynamic random graph model which interleaves addition of nodes and edges with a *preferential* deletion of nodes that favors deletion of small-degree nodes. The results derived in this chapter appeared first in [DC05c]. This type of node deletion has been chosen in light of the evidence that the small-degree nodes of some web-like networks, such as the Web and the Internet, die much more frequently than the high-degree ones [BBKT04, VPV02].

The birth-death with preferential deletion analyzed in this chapter is defined as follows: Let the graph G_1 consists of a single node with a self-loop. In each discrete time-step $t+1$, $t>0$, either one of the following two processes can take place:

(a) Birth process: with probability p, a new node is added, together with a new edge incident on it. The other end-node u of the new edge is chosen preferentially from among all the existing nodes based on a *linear preferential attachment* [BAJ99] rule:

$$\mathbb{P}_{t+1}[u] = \frac{d_t(u)}{\sum_{w \in V_t} d_t(w)} = \frac{d_t(u)}{2m_t}. \qquad (3.1)$$

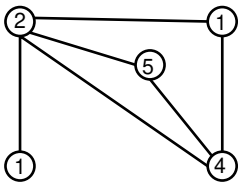
(b) Death process: with probability $q = 1 - p$, a node u is chosen for deletion along with all the edges incident on this node in G_t. To make small-degree nodes more likely candidates for deletion than the higher-degree ones, node u is chosen according to the following probability distribution:

$$\mathbb{P}_{t+1}[u] = \frac{n_t - d_t(u)}{n_t^2 - 2m_t}. \qquad (3.2)$$

Note that the numerator of the ratio in the right-hand side of Equation (3.2) subtracts the degree of node u from the number of nodes in the graph G_t. Therefore, the value assigned by equation (3.2) will be larger for small-degree nodes than for higher-degree ones. The division by $n_t^2 - 2m_t$ ensures that the values lie between 0 and 1. Naturally, there exist other alternative probability distributions that may achieve the same effect, such as:

$$\mathbb{P}_{t+1}[u] = \frac{2m_t - d_t(u)}{2m_t(n_t - 1)}.$$

The distribution in Equation (3.2) was chosen primarily because it was more convenient to work with.



$\mathbb{P}(1) = 4/13$

$\mathbb{P}(3) = \mathbb{P}(5) = 3/13$

$\mathbb{P}(4) = 2/13$

$\mathbb{P}(2) = 1/13$

Figure 3.1. A small graph illustrating the probability distribution used in the preferential deletion model.

As an illustration, a graph with deletion probabilities assigned to its nodes according to Equation (3.2) is given in Figure 3.1.

It is assumed that p is greater than q so that the graph continues to grow. There is a caveat to the two rules (a) and (b): If during some step $t > 0$ the number of nodes in G_t becomes zero, then the process rewinds, *i.e.,* the graph G_{t+1} will consist of a single node with a self-loop. However, as shown in the next section, this case occurs extremely rarely, and thus it may be ignored. In analogy with preferential attachment [BAJ99], the death process defined above is called *preferential deletion*.

3.1. Number of Nodes

First, let us look at $\mathbb{P}[n_t = 0]$—the probability that the number of nodes becomes zero after some step $t > 0$. This event could occur only if t is even and exactly $t/2$ death processes have taken place during steps $1,\ldots,t$ (note that the starting graph G_1 may be seen as the result of a birth process). Thus:

$$\mathbb{P}\big[n_t = 0\big] \le \binom{t}{t/2} q^{t/2} p^{t/2} = O\Big[\big(2\sqrt{pq}\big)^t\Big] = O\left[\frac{1}{(1+\varepsilon)^t}\right],$$

for some $\varepsilon > 0$. Since the probability of reaching an empty graph decreases exponentially with the number of steps, it is assumed that $n_t > 0$ for all $t > 0$. Hence, for all $t > 0$, $n_{t+1} = n_t + X$, where X is a discrete random variable equal to 1 with probability p and equal to -1 with probability q. As a result, the conditional expectation of n_{t+1} with G_t fixed is

$$\mathbb{E}\big[n_{t+1} \mid G_t\big] = n_t + \mathbb{E}[X]. \qquad (3.3)$$

By taking the expectations of both sides in Equation (3.3) we obtain

$$\mathbb{E}[n_{t+1}] = \mathbb{E}[n_t] + (p - q), \quad \text{for} \quad t > 1.$$

Solving this first-order linear difference equation with initial condition $\mathbb{E}[n_1] = 1$, yields:

$$\mathbb{E}[n_t] = (p - q)t + 2q\,, \qquad (3.4)$$

which implies that $\mathbb{E}[n_t] = \Theta[(p - q)t]$. Figure 3.2 shows a comparison of the values of n_t predicted by Equation (3.4) with those obtained by simulating the preferential deletion model.

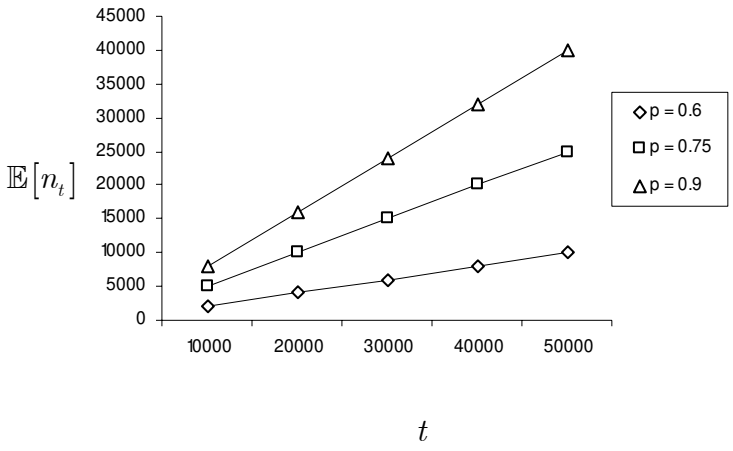


Figure 3.2. Growth in the number of nodes of graph G_t with the number of steps t, for three different values of the birth probability p.

In this figure, the solid lines correspond to the analytical prediction of Equation (3.4) while the data points correspond to the simulation result. To obtain these data points, for each value of p and t shown in Figure 3.2 the number of nodes was computed by averaging over 30 realizations of the model. The analytical prediction and simulation results agree very well, as seen in Figure 3.2.

3.2. Number of Edges

The approach followed in this section is similar to that of Section 3.1. With G_t fixed, the number of edges after the $(t+1)^{\text{th}}$ step may be expressed as $m_{t+1} = m_t + Y_{t+1}$, where Y_{t+1} is a random variable specified by

$$Y_{t+1} = \begin{cases} 1 & \text{with probability} \quad p \\ -k & \text{with probability} \quad \dfrac{q(n_t - k)N_{t,k}}{n_t^2 - 2m_t}, k \geq 0. \end{cases}$$

Thus,

$$\mathbb{E}[Y_{t+1} \mid G_t] = p - q\sum_{k\geq 0} kN_{t,k}\,\frac{n_t - k}{n_t^2 - 2m_t} = p - q\sum_{k\geq 0}\frac{kN_{t,k}}{n_t - \overline{d}_t} + q\sum_{k\geq 0}\frac{k^2 N_{t,k}}{n_t^2 - 2m_t},$$

which implies that

$$\mathbb{E}[m_{t+1}] = \mathbb{E}[m_t] + p - q\mathbb{E}\left[\sum_{k\geq 0}\frac{kN_{t,k}}{n_t - \overline{d}_t}\right] + q\mathbb{E}\left[\sum_{k\geq 0}\frac{k^2 N_{t,k}}{n_t^2 - 2m_t}\right]. \tag{3.5}$$

We continue by evaluating the two expectations multiplied by q in Equation (3.5). First, we have

$$\mathbb{E}\left[\sum_{k\geq 0}\frac{kN_{t,k}}{n_t - \overline{d}_t}\right] = \mathbb{E}\left[\frac{n_t}{n_t - \overline{d}_t}\sum_{k\geq 0}\frac{kN_{t,k}}{n_t}\right] = \mathbb{E}\left[\frac{2m_t}{n_t - \overline{d}_t}\right]$$

Second,

$$\mathbb{E}\left[\sum_{k\geq 0}\frac{k^2 N_{t,k}}{n_t(n_t - \overline{d}_t)}\right] = \mathbb{E}\left[\frac{1}{n_t - \overline{d}_t}\sum_{k\geq 0}\frac{k^2 N_{t,k}}{n_t}\right].$$

Now, using the approximation

$$\sum_{k\geq 0}\frac{k^2 N_{t,k}}{n_t} \approx 2\left(\overline{d}_t\right)^2 \approx \frac{8m_t^2}{n_t^2},$$

and then substituting it into Equation (3.5) we get

$$\mathbb{E}[m_{t+1}] = \mathbb{E}[m_t] + p - q\left(\mathbb{E}\left[\frac{2m_t}{n_t - \bar{d}_t}\right] - \mathbb{E}\left[\frac{8m_t^2}{n_t^2(n_t - \bar{d}_t)}\right]\right),$$

or, equivalently

$$\mathbb{E}[m_{t+1}] - \left(1 - \frac{2q}{\mathbb{E}[n_t - \bar{d}_t]}\right)\mathbb{E}[m_t] - \frac{4q}{\mathbb{E}[n_t^2(n_t - \bar{d}_t)]}\mathbb{E}[m_t]^2 = p, \qquad (3.6)$$

which is a non-linear difference equation. Methods for solving such equations are known only for a few special cases (these methods are usually based on transformations that convert non-linear equations into linear ones). To the best of our knowledge Equation (3.6) does not fall into any of the special cases. Therefore, we search for a solution of the form $\mathbb{E}[m_t] = \varepsilon t$, where ε is a constant that does not depend on t. Substituting into Equation (3.6) and taking the limits as $t \to \infty$ we get:

$$(1 + a)\varepsilon = p \qquad (3.7)$$

where $a = 2q/(p - q)$. Hence:

$$\varepsilon = p(p - q), \qquad (3.8)$$

i.e., $\mathbb{E}[m_t] = \Theta[p(p - q)t]$. To verify the result of Equation (3.8), we computed the number of edges by simulating the model. The simulation results are shown in Figure 3.3. The solid lines in this figure correspond to the analytical prediction for the number of edges m_t while the data points correspond to simulation results. Again, for each value of p and t the number of edges of G_t was computed by averaging over 30 realizations of the model. The two data sets are in very good agreement.

A direct consequence of Equations (3.4) and (3.8) is that the average degree of G_t tends to $2p$ as $t \to \infty$.

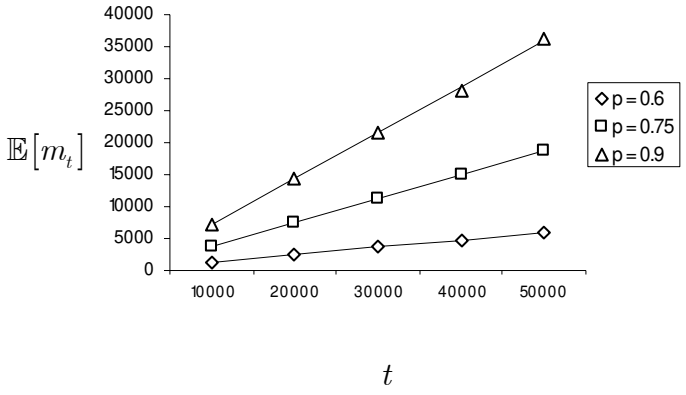


Figure 3.3. Growth in the number of edges of graph G_t with the number of steps t, for three different values of the birth probability p.

3.3. Degree Distribution in the First Neighborhood of the Deleted Node

Before turning our attention to the degree distribution in G_t, we need to evaluate one more quantity, namely the expectation of $N_{t,k}^{(1)}$—the number of degree k nodes adjacent to the node chosen for deletion during step t.

This expectation is computed by conditioning on the node chosen for deletion. Indeed, with G_t fixed, one may write

$$\mathbb{E}\left[N_{k,t}^{(1)} \mid G_t\right] = \sum_{u \in V_t} N_{t,k}^{(1)}(u) \frac{n_t - d_t(u)}{n_t^2 - 2m_t} = \frac{1}{n_t - \bar{d}_t} \sum_{u \in V_t} N_{t,k}^{(1)}(u) - \frac{1}{n_t(n_t - \bar{d}_t)} \sum_{u \in V_t} N_{t,k}^{(1)}(u) d_t(u) \quad (3.9)$$

Next, note that

$$\sum_{u \in V_t} N_{t,k}^{(1)}(u) = k N_{t,k} \quad (3.10)$$

and

$$\sum_{u \in V_t} N_{t,k}^{(1)}(u) d_t(u) = \sum_{i=1}^{N_{t,k}} \sum_{j=1}^{k} d_{k,i,j,t} \qquad (3.11)$$

Here $d_{k,i,j,t}$ denotes the degree of the j^{th} neighbor of the i^{th} node of degree k after step t. It may be approximated by the average degree $\overline{d}_t^{(1)}$ of a random neighbor of a random node. This quantity is related to $\overline{d}_t$ by the identity $\overline{d}_t^{(1)} \approx 2\overline{d}_t$. Substituting into Equation (3.9) we get

$$\mathbb{E}\left[N_{t,k}^{(1)} \mid G_t\right] \approx \frac{kN_{t,k}}{n_t - \overline{d}_t} - \frac{2kN_{t,k}\overline{d}_t}{n_t(n_t - \overline{d}_t)} = \frac{kN_{t,k}}{n_t - \overline{d}_t}\left[1 - \frac{2\overline{d}_t}{n_t}\right],$$

and finally, by taking the expectations of both sides in the last equation we obtain

$$\mathbb{E}\left[N_{t,k}^{(1)}\right] \approx k\mathbb{E}\left[N_{t,k}\right]\mathbb{E}\left[\frac{1 - 2\overline{d}_t/n_t}{n_t - \overline{d}_t}\right] \qquad (3.12)$$

Equation (3.12) was also verified numerically. The results are shown in Figure 3.4 where the solid line corresponds to the prediction of Equation (3.12) while the data points were computed by averaging over 1000 realizations of our model with $t = 40,000$ and $p = 0.8$. The values of $\mathbb{E}[N_{t,k}^{(1)}]$ in Figure 3.4 are shown in a normalized form after having been divided by the degree of the node chosen for deletion.

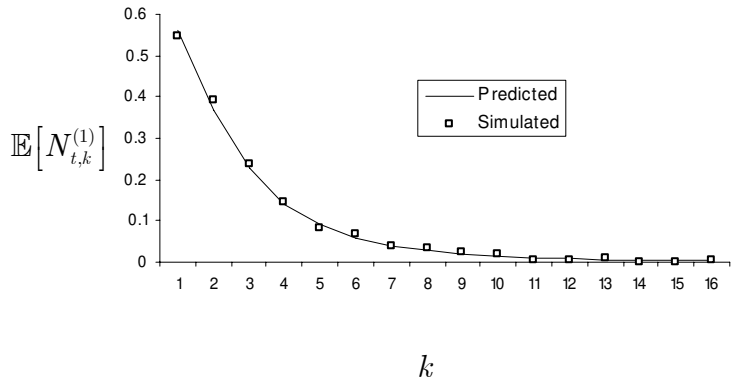


Figure 3.4. The expected number of neighbors of degree k of a node chosen for deletion.

3.4. Degree Distribution

Next, we turn to the degree distribution of the graph G_t. By analyzing the change in $N_{t,k}$ between the t^{th} and $(t+1)^{th}$ steps we get

$$\mathbb{E}[N_{t+1,k}] = \mathbb{E}[N_{t,k}] + pA + qB + p\delta_{k1}, \qquad (3.13)$$

where

$$A = \frac{1}{2\mathbb{E}[m_t]}\big[(k-1)\mathbb{E}[N_{t,k-1}] - k\mathbb{E}[N_{t,k}]\big],$$
$$B = \frac{q}{\mathbb{E}[n_t(n_t - \overline{d}_t)]}\{(k+1)\mathbb{E}[N_{t,k+1}]\mathbb{E}[n_t - 2\overline{d}_t] - [(k+1)n_t - k(1-2\overline{d}_t)]\mathbb{E}[N_{t,k}]\}.$$

Term A in Equation (3.13) reflects the expected change in $N_{t,k}$ due to the birth process. The expression for term A was derived using standard techniques (*e.g.*, [KR03]), and hence the details have been omitted.

Term B reflects the expected change in $N_{t,k}$ due to deletion. Its derivation takes into account the result of Section 3.3. Let us examine, for instance, the derivation of the term $[(k+1)n_t - k(1-2\overline{d}_t)]\mathbb{E}[N_{t,k}]\big/\mathbb{E}[n_t(n_t - \overline{d}_t)]$, which reflects the expected drop in $N_{t,k}$ due to a deletion. The deletion of a node can cause $N_{t,k}$ to decrease in two different ways: (i) if a node of degree k is deleted; or (ii) if the deleted node is adjacent to one or more nodes of degree k. The expected drop due to deletion of a node of degree k is $\mathbb{E}[N_{t,k}](n_t - k)\big/\mathbb{E}[n_t^2 - 2m_t]$. Furthermore, the result of Equation (3.12) implies that the expected drop due to deletion of a node which has one or more neighbors of degree k is $k\mathbb{E}[N_{t,k}](n_t - 2\overline{d}_t)\big/\mathbb{E}[(n_t^2 - 2m_t)]$. Adding these two drop terms yields the expected overall drop due to deletion in the number of nodes of

degree k. In a similar fashion, one may also derive the expected *increase* in $N_{t,k}$ due to a deletion.

The last term in Equation (3.13) comes from the fact that the degree of a newly-born node is always one.

We search for a solution to Equation (3.13) of the form $\mathbb{E}[N_{t,k}] = a_k t$. Substituting for $\mathbb{E}[N_{t,k}]$ and taking the limits as $t \to \infty$, we get the following second-order, linear difference equation with non-constant coefficients:

$$-2q(k+2)a_{k+2} + [2p + (k+1)(2q+1)]a_{k+1} - ka_k = 2p(p-q)\delta_{k1}, \quad k \geq 0. \quad (3.14)$$

To solve Equation (3.14) we have used the method of Laplace as described in [Jor65]. Consider first the homogeneous equation which has the form $\sum_{i=0}^{2} (\alpha_i k + \beta_i) a_{k+i} = 0$, with

$$\alpha_0 = -1,\ \beta_0 = 0,\ \alpha_1 = 2q+1,\ \beta_1 = 3,\ \alpha_2 = -2q,\ \beta_2 = -4q.$$

Following Laplace's method, it is assumed that the solution of the homogenous equation is of the form:

$$a_k = \int_a^b t^{k-1} v(t)\, dt, \qquad (3.15)$$

where the function $v(t)$ and the limits of integration a, b are yet to be determined. As shown in [Jor65], the relation

$$\frac{dv}{v} = \frac{\sum t^i (\beta_i - i\alpha_i)}{\sum \alpha_i t^{i+1}} \qquad (3.16)$$

must hold for any difference equation of the type under consideration. Furthermore, the limits of integration a, b are to be chosen among the roots of the function $t^k v(t)\sum \alpha_i t^i$. In the present case, Equation (3.16) becomes

$$\frac{dv}{v} = \frac{2p - 4pt}{-1 + (1+2q)t - 2qt^2}.$$

By integrating both sides of the preceding equation we get:

$$v(t) = (t-1)^{2p/2p-1} (2tq-1)^{2p^2/q(p-q)}.$$

The roots of $t^k v(t)\sum \alpha_i t^i$ are 0, 1, and $1/2q$. It follows that the two independent solutions of Equation (3.13) are

$$a_k^{(1)} = \int_0^1 t^{k-1} (t-1)^{2p/2p-1} (2tq-1)^{2p^2/q(p-q)}\, dt,$$

and

$$a_k^{(2)} = \int_0^{1/2q} t^{k-1} (t-1)^{2p/2p-1} (2tq-1)^{2p^2/q(p-q)}\, dt.$$

By carrying out the first integration we get:

$$a_k^{(1)} = \Theta\left[k^{-1-\left(2p/2p-1\right)}\right]. \qquad (3.17)$$

Note that as p increases from 0.5 to 1, the ratio $2p/2p-1$ decreases from ∞ to 2. Thus, asymptotically the degree distribution of G_t follows a power-law with exponent that varies between 3 (for $p=1$) and ∞ (for $p=0.5$). In the case when $p=1$ this result agrees with previous well-known results [BAJ99, BRST01]. On the other hand, for values of p significantly smaller than 1 the exponent of the power-law becomes too big compared to the exponents

observed for many web-like networks (which usually lie in the range between 2 and 3 [New03b]).

The second integral $a_k^{(2)}$ may be shown to diverge as $k \to \infty$, and is thus irrelevant.

The plot in Figure 3.5 shows a comparison between the analytical prediction given by Equation (3.17) and the data obtained by simulating our model, with $p = 0.8$.

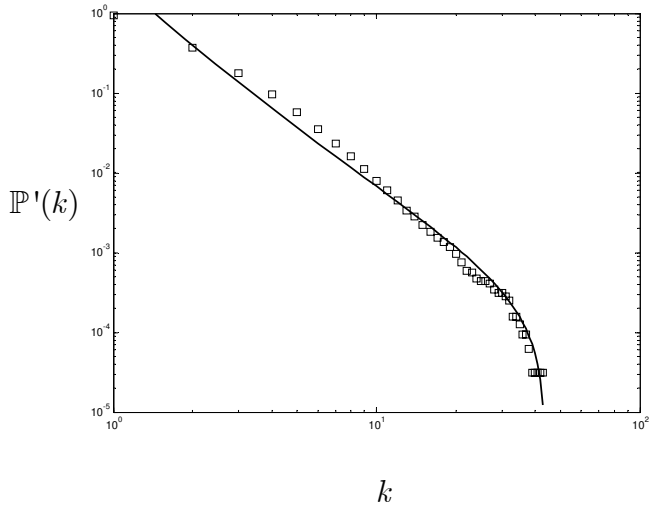


Figure 3.5. Log-log plot of the cumulative degree distribution of the graph generated by the preferential deletion model.

The cumulative distribution $\mathbb{P}'(k) = \sum_{i \geq k} \mathbb{P}(i)$ has been plotted instead of the distribution $\mathbb{P}(k)$ itself in order to reduce the statistical noise in the tail of the distribution. As seen, there is a good agreement between the data obtained from the simulation results and the analytical prediction.

The result derived in this chapter reinforces our view that dynamic models of web-like networks are robust in the sense that a power-law degree distribution is obtained for a wide range of stochastic rules that control such models.

4. THE NOTION OF COMMUNITY

This chapter aims to formalize the meaning of *community*. Before giving formal definitions of community, we review some classical combinatorial optimization problems that have the same flavor as community mining.

4.1. Some Graph-Theoretic Problems Related to Community-Mining

Minimum cuts and graph partitioning. Let A, B form a partition of the set of nodes of a graph $G = (V, E)$. An *edge-cut* (A, B) in G is the set of all edges with one end in A and the other end in B. The *min-cut* problem refers to finding the edge-cut with minimum cardinality. This problem is NP-hard [GJ79] an the two main heuristic algorithms for solving this problem are: the *spectral bisection* method [Fie73], which is based on the eigenvectors of the graph Laplacian, and the *Kernighan-Lin* algorithm [KL70], which improves on an initial division of the graph using a greedy strategy. It should be mentioned that the restricted version of the min-cut problem known as the s-t min-cut—where s and t are two fixed nodes and each of the two partitions must contain exactly one of them—can be solved in polynomial time (*e.g.,* using the max-flow/min-cut algorithm of Ford and Fulkerson). *Graph partitioning* is a generalization of the min-cut problem which refers to partitioning the set of nodes of a graph G into two or more partitions such that the number of edges having their ends in different partitions is minimal. This problem is usually solved by a repeated application of the bisection method.

Maximum clique. Any complete subgraph of a graph G is called a *clique* of G. A clique is said to be *maximal* if it is not properly contained in any other clique. A *maximum* clique is a clique of maximum size. The problem of finding the clique number of a graph (the size of a

maximum clique) as well as the problem of finding a maximum clique are NP-hard [GJ79]. In addition, some theoretical results indicate that it is difficult to find approximation algorithms that guarantee to find cliques of size within a factor of the maximum clique size [FGLS91]. Therefore the problem of finding a maximum clique is usually attacked by approximation or heuristic algorithms; a comparative survey of such algorithms is given in [Pel01].

It is interesting to note that some clique-related problems that are difficult to solve for arbitrary graphs become easy in the case of random graphs. For instance, it is known that *a.a.s* the clique number of a binomial random graph $G_{n,\frac{1}{2}}$ is either $\lfloor f(n) \rfloor$ or $\lceil f(n) \rceil$ where $f(n) = \Theta(2\log n)$ [AS92]. Several polynomial-time algorithms are known to find *a.a.s* a clique of size $\Theta(\log n)$, that is a clique of roughly half the size of the largest one. On the other hand, no polynomial-time algorithms are known to find *a.a.s* a clique of size $(1+\varepsilon)\log n$ for a fixed $\varepsilon > 0$. The situation improves in a related random graph model, namely $G_{n,\frac{1}{2},k}$, which is obtained by first generating a random graph $G_{n,\frac{1}{2}}$, then selecting k random nodes from this graph and forcing them to be a clique by adding edges as needed. Among other results, it has been shown that for every $\varepsilon > 0$ there is a polynomial-time algorithm to find a hidden clique on k nodes in $G_{n,\frac{1}{2},k}$, provided that $k \geq \varepsilon\sqrt{n}$ [AKS98].

The study of the performance of combinatorial algorithms in random graphs is known as the *algorithmic random graph theory*; the major results in this area have been surveyed in a paper by Frieze and McDiarmid [FM97].

Alliances. The concept of alliance has been introduced to graph theory by Kristiansen *et al.* [KHH04]. To date, many types of alliances in graphs have been defined. Some of them are reviewed next.

Let $G = (V, E)$ be a graph and v a node in this graph. The *open neighborhood* of v is defined as the set $N(v) = \{u \mid (u, v) \in E\}$, while the *closed neighborhood* of v is the set $N[v] = N(v) \cup \{v\}$. For a subset S of V the open and closed neighborhoods are defined as $N(S) = \cup_{v \in S} N(v)$ and $N[S] = N(S) \cup S$, respectively. The *boundary* of a set of nodes S is defined as the set $\partial S = N[S] - S$.

A set of nodes S is called a *defensive alliance* if $| N[v] \cap S | \geq | N[v] - S |$ for every node $v \in S$ and an *offensive alliance* if $| N[v] \cap S | \geq | N[v] - S |$ for every $v \in N[S] - S$. As it turns out the idea of defensive alliance has appeared in some earlier publications prior to being given the name defensive alliance in [KHH04]: Flake *et al*. [FLG00] defined a Web community as a set of web sites C in which every member at least as many neighbors inside C as outside it; the book by Wasserman and Faust [WF94] which deals with the analysis of social networks also studies groups of nodes with similar properties as a defensive alliance.

A set of nodes S is called a *powerful alliance* if it is both a defensive and an offensive alliance. A defensive (offensive, powerful) alliance S is said to be *global* if it is also a dominating set. Several papers [HHH03, HHK04, BDH04] have initiated the study of mathematical properties of alliances.

4.2. Graph-theoretic Definitions of Community

This section gives two definitions of community: (i) $p\text{-}alliance$ (Definition 4.1); and (ii) $\alpha\text{-}near\text{-}clique$ (Definition 4.2). The first definition was proposed Flake *et al.* [FTT04] and is a generalization of the concept of defensive alliance.

The second definition of community—$\alpha\text{-}near\text{-}clique$—is a new concept proposed in this dissertation.

Definition 4.1. Let $p \in [0,1]$. A $p\text{-}alliance$ in a graph $G = (V, E)$ is defined as a subset of nodes C_p satisfying the property $\left|N(u) \cap C_p\right| \geq p\left|N(u) - C_p\right|$ for every node $u \in C_p$. A $p\text{-}alliance$ C_p is called *minimal* if no proper subset of C_p forms a $p\text{-}alliance$, *minimum* if C_p has the smallest cardinality among all $p\text{-}alliances$, and *global* if it dominates the graph G.

The parameter p in the definition of community as a $p\text{-}alliance$ quantifies the strength of a community: If $p = 0$, then any set of nodes would be a $p\text{-}alliance$. At the other extreme, if $p = 1$, then a $p\text{-}alliance$ is the same as a defensive alliance.

It turns out that several community-mining problems are NP-hard under the definition of community as a $p\text{-}alliance$ (Section 4.3). Therefore it becomes necessary to investigate the existence of alternative definitions of community which render community-mining amenable to polynomial-time algorithms. The Definition 4.2 shown next aims to achieve exactly that.

Definition 4.2. Let $\alpha \in [0,1]$. An $\alpha\text{-}near\text{-}clique$ is defined as a subset of nodes C_α such that the clustering coefficient of each node in the induced subgraph $G[C_\alpha]$ is greater than or equal to α. An $\alpha\text{-}near\text{-}clique$ C_α is called *maximal* if no proper superset of C_α is an $\alpha\text{-}near\text{-}clique$. A *maximum* $\alpha\text{-}near\text{-}clique$ is one that has maximum cardinality.

The parameter α in the definition of community as an $\alpha\text{-}near\text{-}clique$ quantifies again the strength of relationship among the nodes of a community: If $\alpha = 0$, then every subset of nodes forms an $\alpha\text{-}near\text{-}clique$ and only V forms a maximal $\alpha\text{-}near\text{-}clique$; If $\alpha = 1$, then only the nodes of a clique would satisfy the definition of an $\alpha\text{-}near\text{-}clique$ (Proposition 4.1).

Note that implicit in Definitions 4.1 and 4.2 is the requirement that the subgraph induced by the nodes of a $p\text{-}alliance$ or an $\alpha\text{-}near\text{-}clique$ be connected.

Example 4.1: The following examples show the largest value of α for which some well-known graphs form an $\alpha\text{-}near\text{-}clique$.

- The complete graph K_n is an $\alpha\text{-}near\text{-}clique$ for $\alpha = 1$.
- The k-nearest neighbor lattice, $k = 2$. This graph is an $\alpha\text{-}near\text{-}clique$ for $\alpha = \frac{1}{2}$.
- The complete bipartite graph $K_{m,n}$ is not an $\alpha\text{-}near\text{-}clique$ for any $\alpha > 0$. ■

Proposition 4.1 gives some basic properties of $\alpha\text{-}near\text{-}cliques$.

Proposition 4.1. The following properties hold:

a) If a set S forms an $\alpha\text{-}near\text{-}clique$ then it would form $\alpha'\text{-}near\text{-}clique$ for all $\alpha' < \alpha$.

b) If a subset of nodes S of a graph G is an $1\text{-}near\text{-}clique$ then the induced subgraph $G[S]$ is a clique.

c) If the set of nodes of a graph G forms an $\alpha\text{-}near\text{-}clique$ then $C(G) \geq \alpha$.

Proof. Trivial.

Note that the two definitions of community given above are quite different from each other. First, while the definition of a community C as an $\alpha\text{-}near\text{-}clique$ involves only the nodes of C and the edges with both ends in C, the definition as a $p\text{-}alliance$ involves the nodes of C, the edges with both ends in C *and* the edges with only one end in C. Second,

under the definition of community as an α-$near$-$clique$ only the problems about maximal communities are interesting (any triangle would be a 1-$near$-$clique$), while under the definition of a community as a p-$alliance$ only the problems about minimal communities are interesting (the set of nodes of the graph itself is a 1-$alliance$). This asymmetry between the two definitions of community implies differences in the ranges of applications that suits each of them; we will return to this point in Section 4.3.

Having defined the concept of community, we return to the definition of *community mining*. This problem may be posed in at least two different versions, as shown in Definition 4.3 and 4.4:

Definition 4.3. Let $G = (V, E)$ be a graph and S a subset of V. The community-mining problem $\mathcal{P}_1$ is defined as the problem of finding a maximal (minimal) community in G which contains (is contained in) the set of nodes S.

Definition 4.4. Let $G = (V, E)$ be a graph. The community-mining problem $\mathcal{P}_2$ is defined as the problem of finding a partition of the set of nodes V into two or more subsets such that each subset is a community in G.

Thus in the problem $\mathcal{P}_1$ the goal is to find a *single* community which satisfies certain requirements, while in the problem $\mathcal{P}_2$ the goal is to find *all* communities.

4.3. Computational Complexity of Community Mining

In view of the definitions of community in Section 4.2, it is natural to ask whether there exist polynomial-time algorithms to solve the community-mining problems $\mathcal{P}_1$ and $\mathcal{P}_2$. The answer to this question makes up the topic of the present section.

In order to highlight the applications that better suit the various definitions of community and community-mining, next we present some Web-application scenarios each of which boils down to a specific version of community mining. For each of the subsequent scenarios we discuss the computational complexity of the community-mining problem arising in that scenario.

Scenario 1: Clustering the responses to a user query by a search engine

Consider the following procedure for clustering the responses by a search engine to a user query:

- Send a query on a pre-defined topic to a search engine, say Google.
- Let the set of many of the top responses returned by the search engine, say the first $10,000$, be denoted by R.
- Construct the subgraph induced by R in the Web graph; call this the *context graph* G_1 of the given query.
- Solve the community-mining problem $\mathcal{P}_2$ on the graph G_1. The solution to this problem is expected to partition the set of nodes of the context graph into two or more partitions—each representing a subtopic of the query's topic.

Assuming the definition of community as a $p\text{-}alliance$ with $p \geq \frac{1}{2}$, the problem described in *Scenario 1* is NP-hard. This conclusion follows directly from the fact that the following decision problem:

Given: An undirected, weighted graph $G = (V, E, w)$, a real number $p \geq 1/2$, and positive integer k.

Question: Can the nodes of G be partitioned into k disjoint sets $V_1, \ldots, V_k$, such that for all V_i and $u \in V_i$, $\sum_{v \in V_i} w(u,v) \geq p \sum_{v \in V} w(u,v)$?

is NP-complete—a result obtained by Flake *et al.* [FTT04] by using a transformation from BALANCED PARTITION [GJ79].

The computational complexity of the problem described in *Scenario 1* is open under the definition of community as an $\alpha\text{-}near\text{-}clique$.

Scenario 2: Dominating community

The procedure below shows a method for extracting a small "strong" community from a larger one by selecting a subset of nodes of the large community that satisfies the definition of community:

- Let the set of web pages that belong to a broad topic be denoted by S. This set may be obtained for example using a topic directory such as Yahoo! or Open Directory.
- Construct the subgraph of the Web graph induced by S. Call this graph the *context graph* G_2.
- Solve the minimal community-mining problem $\mathcal{P}_1$ on the context graph G_2.

The solution to this problem is expected to find a subset of nodes that is of high-quality, or central to the broad topic at hand.

Assuming a definition of community as a *global* p-*alliance* with $p = 1$, the computational complexity of this problem has been analyzed in [CBDD04] where it was proved—using transformations from the DOMINATING SET (DS) problem [GJ79]—that the problems of finding a minimum global defensive (offensive, powerful) alliance are NP-hard.

To explain the idea of the transformations used in [CBDD04], we show the details of the proof for the case of global defensive alliance; the complete proofs for the case of offensive and powerful alliance may be found in [CBDD04].

The decision version of the problem of finding a minimum global defensive alliance is:

GLOBAL DEFENSIVE ALLIANCE (GDA):

Given: A graph $G(V, E)$ and a positive integer $K \leq | V |$.

Question: Is there a global defensive alliance in G of size K or less?

Theorem 4.1. GDA is NP-Complete.

Proof.

The GDA problem is clearly in the set NP. Let $I = [G(V, E), K]$ be any instance of DS. We need to construct an instance $I' = [G'(V', E'), K']$ of GDA such that G has a dominating set of size K or less if and only if G' has a global defensive alliance of size K' or less.

First, let us describe a procedure to construct the graph G': Initially let $G' = G$. Then, for each non-isolated node $v_i \in V$, add $d_G(v_i) - 1$ components $C_{i,1}, \ldots, C_{i,d_G(v_i)-1}$ to G'. Each

component $C_{i,j}$ consists of two nodes and two edges as follows:

$$C_{i,j} = (\{a_{i,j}, b_{i,j}\}, \{(v_i, a_{i,j}), (a_{i,j}, b_{i,j})\}).$$

In other words the node $a_{i,j}$ of the component $C_{i,j}$ is connected to the root v_i as well as to the other node $b_{i,j}$ of this component. We say that the components $C_{i,j}, 1 \le j \le d_G(v_i) - 1$ are *rooted* at v_i. Letting

$$A_{v_i} = \{a_{i,j} \mid 1 \le j \le d(v_i) - 1\}, \quad B_{v_i} = \{b_{i,j} \mid 1 \le j \le d(v_i) - 1\},$$

$$A_S = \bigcup_{v_i \in S} A_{v_i}, \quad B_S = \bigcup_{v_i \in S} B_{v_i},$$

and

$$A = A_V, \quad B = B_V,$$

the graph G' is completely specified by

$$V' = V \cup A \cup B,$$

$$E' = E \cup \left(\bigcup_{v_i \in V} \{(v_i, a_{i,j}), (a_{i,j}, b_{i,j}) \mid a_{i,j} \in A_{v_i}, b_{i,j} \in B_{v_i}\} \right).$$

In the remainder of the proof, we shall refer to the nodes (edges) of components $C_{i,j}$ as *component* nodes (edges), to distinguish them from the nodes (edges) of V. Let Q be the total number of components $C_{i,j}$. To complete the construction of the instance I' we let $K' = K + Q$. Figure 4.1 shows an example of the application of this procedure. Both graphs G (on the left) and G' (on the right) are shown in Figure 4.1. The *component* nodes are represented by empty circles and *component* edges are represented by dotted lines. The total number of components in this example is $Q = 7$.

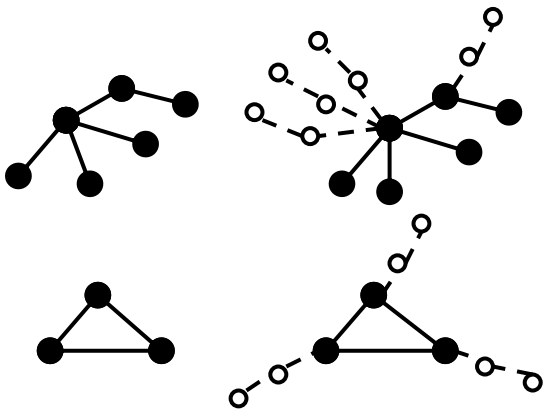


Figure 4.1. Construction of an instance of GDA from an instance of DS.

Note that $Q = \sum_{v_i \in V} (d_G(v_i) - 1) = 2 \mid E \mid - \mid V \mid$. Therefore, the construction of G' can be accomplished in linear time.

To complete the proof of Theorem 4.1, it remains to show that G has a dominating set of size K or less if and only if G' has a global defensive alliance of size K' or less.

First, suppose that $S \subseteq V$ is a dominating set in G with $\mid S \mid \leq K$. Let

$$S' = S \cup A_S \cup B_{V-S}$$

Note that S is a subset of S'. Furthermore, for each node $v_i \in S$, S' contains all the nodes $a_{i,1},\ldots,a_{i,d_G(v_i)-1}$. Finally, for each node $v_j \notin S$, S' contains all the nodes $b_{j,1},\ldots,b_{j,d_G(v_j)-1}$. These observations together with Lemma 1 imply that I' is a YES instance of GDA problem.

Lemma 4.1. S' is a global defensive alliance in G' with size K' or less.

Proof.

S' contains all nodes of S as well as one node from each component $C_{i,j}$. Therefore,

$$| S' | = | S | + Q \leq K + Q = K'.$$

Furthermore, S' dominates V (since S is a dominating set in G and $S \subseteq S'$), and, it also dominates all the components $C_{i,j}$ (because S' contains exactly one node from every such component). Thus, S' is a dominating set in G'.

Finally, S' is a defensive alliance in G'. To see this, first note that every node $v_i \in S' \cap V$, has exactly $d_G(v_i) - 1$ neighbors $a_{i,1}, \ldots, a_{i,d_G(v_i)-1}$ in the set S'. Since v_i can have at most $d_G(v_i)$ neighbors outside S' (which happens only if all the neighbors of v_i in V are outside S'), the defensive alliance property is satisfied for v_i. Furthermore, each node $a_{i,j} \in S'$ has exactly one neighbor inside S' (the "root" node v_i) and exactly one neighbor outside (the node $b_{i,j}$), thus it satisfies the defensive alliance property. Finally, each node $b_{i,j} \in S'$ has degree one in G', therefore it satisfies the defensive alliance property.

Conversely, suppose that S' is a global defensive alliance in G' with K' or less nodes. We need to find a set $S \subseteq V$ of size K or less that forms a dominating set in G. Let us begin with the following simple observation:

Observation 4.1. S' contains at least Q *component* nodes.

Proof.

S' is a dominating set in G', hence it contains at least one node from each component $C_{i,j}$.

An immediate corollary of Observation 4.1 is that

$$| S' \cap V | \leq K' - Q = K.$$

Since the set $S' \cap V$ has size at most K, this set may be considered as a first candidate for a dominating set in G. However, $S' \cap V$ does not necessarily form a dominating set in G, because there might exist a node $v_i \in (V' - S') \cap V$ which has no neighbor in $S' \cap V$ (see Figure 4.2).

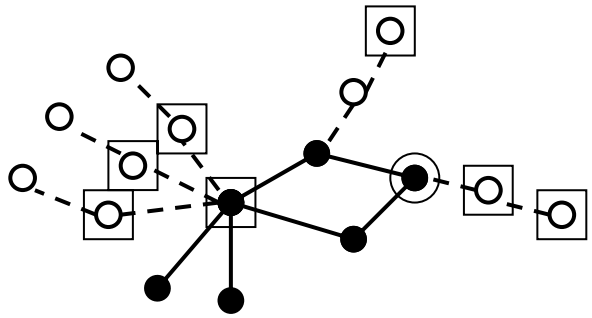


Figure 4.2. A graph G', a global defensive alliance S' in G' (nodes surrounded by squares) and a *non-component* node (surrounded by a circle) that has only one neighbor in S' which is a *component* node.

Such a node, v_i wouldn't be dominated by $S' \cap V$. We say that v_i is a *component-dominated* node. Now, let D' be the set of *component-dominated* nodes *i.e.*,

$$D' = \{v_i \in (V' - S') \cap V \mid v_i \text{ has no neighbor in } V \cap S'\}.$$

Note that the nodes of D' are the only ones among the nodes of V that are *not* dominated by $S' \cap V$. Hence, the set $(S' \cup D') \cap V$ must form a dominating set in G. The next lemma, which is a strengthened version of Observation 1, implies that $| S' | \leq K$.

Lemma 4.2. S' contains at least $Q+ \mid D' \mid$ *component* nodes.

Proof.

Consider an arbitrary node $v_i \in D'$. There must be a node $a_{i,j}$ such that $a_{i,j} \in S'$ (because S' is a dominating set and v_i does not have any neighbor in $S' \cap V$). Now, the node $b_{i,j}$ must also be in S', because otherwise the defensive alliance property would be violated for $a_{i,j}$. Hence, for every node $v_i \in D'$ there exists at least one component $C_{i,j}$ with both nodes in S'. This implies that, in total, there are $\mid D' \mid$ components with both nodes contained in S'. The remaining $Q- \mid D' \mid$ components, must each have at least one node in S' because S' is a dominating set. Therefore, the number of *component* nodes in S' is at least $2 \mid D' \mid + Q - \mid D' \mid$, that is $Q+ \mid D' \mid$.

Now, let

$$S = (S' \cup D') \cap V.$$

From Lemma 4.1 it follows that $\mid S \mid \leq K$. Since, as argued earlier, S is also a dominating set in G, it follows that, I is a YES instance of DS.

Scenario 3: Focused crawling

As mentioned in Chapter 1, focused aims to discover web pages related to a pre-defined topic. The search for such pages is *selective* in the sense that only some search paths that are deemed relevant are followed. Focused crawling may be performed *on-line* or *off-line*. In the on-line version of focused crawling, the graph structure of the relevant portion of the Web is not know beforehand but is obtained during the search; in the off-line version, the graph structure of

the relevant portion of the Web is obtained first and the subset of nodes relevant to the given topic—*i.e.,* the community—is found subsequently.

The following procedure gives a method to perform off-line focused crawling:

- Send a query on a pre-defined topic to a search engine, say Google.
- Let the set of a few of the top responses returned by the search engine, say the first 200, be denoted by R.
- Construct the graph that consist of the nodes of R as well as all the neighborhoods of R up to a certain depth k (this graph is constructed by the following the *forward links* contained in the visited pages as well as the *backward links* that may be obtained using tools such as Connectivity Server [BBHK98]). Call this the *context graph* G_2 of the given query.
- Select manually a few nodes from the set R, say 10 or 20, that form a community. Denote the set of the seed-nodes by S.
- Solve the maximal community-mining problem $\mathcal{P}_1$ with the graph G_2 and the set S as inputs.

The solution to this problem is expected to produce a set of web pages related to the given topic. This procedure may be used, for example, to refresh the indices of a topic directory.

The computational complexity of this problem has not been determined assuming the definition of community as an $\alpha\text{-}near\text{-}clique$. Several polynomial time approximation algorithms for this problem are given in Chapter 7.

5. COUNTING COMMUNITIES IN WEB-LIKE NETWORKS

In order to evaluate the correctness of different definitions of community, it is helpful to analyze the density (or, concentration) of communities in random graph models and in real web-like networks. Random graph models may be analyzed using techniques of the random graph theory, while real networks may be investigated through sampling.

In the following sections we discuss some recent techniques for determining analytically the expected number of simple subgraphs in the dynamic random graph models as well as a technique for estimating the concentration of various subgraphs in large networks. Then we present the results of our extensive sampling experiments to estimate the concentration of *alliances* or *near-cliques* in real-life web-like networks.

5.1. Subgraph Counting in Dynamic Random Graph Models

LCD model

Assume that S is a fixed graph that can be a subgraph of G_t in the single-edge version of the LCD model (Section 2.2). In other words, we assume that (i) S has set of nodes $V(S) \subseteq [n]$; (ii) S does not have any self-loops; and (iii) for every node $i \in V(S)$, there is at most one edge ij with $j < i$.

What is the probability that S is a subgraph of G_t? The answer to this question was first obtained by Bollobas and Riordan [BR04]; the outline of the proof in [BR04] is shown next.

First, let us introduce some notation. Since the graph G_t in the single-edge version of LCD is a tree, we use the notation T_t instead of G_t. For each node $i \le t$ of S let $R_t(i)$ be the

number of nodes $j > t$ such that $ij \in E(S)$. In other words $R_t(i)$ is the number of edges of S coming into node i after time t. Let S_t be the subgraph of S induced by edges ij with $i, j \leq t$. Next, let $C_S(t+1)$ be the number of edges $ij \in E(S)$ with $i \leq t, j > t$. Finally, let

$$X_t = \prod_{ij \in E(S_t)} \mathbb{I}_{\{ij \in E(T_t)\}} \prod_{i \in V(S), i \leq t} [d_t(i)]_{R_t(i)}$$

Here $\mathbb{I}_A$ is the indicator function of event A and $[n]_r = n(n+1)\cdots(n-r+1)$ denotes the *rising* factorials. Let's analyze the definition of X_t: First note that $X_0 = 1$. The first product will be 1 if and only if $E(S_t) \subset E(T_t)$. The second product will be one if $S_t = S$, that is, if there are no nodes or edges of S coming after time t. Therefore, for t sufficiently large (at least as large as the largest node in S), X_t is the indicator variable of the event $\{S \subset T_t\}$ and hence $\lambda_t = \mathbb{E}(X_t)$ is the quantity we wish to calculate. The following lemma establishes a recurrence relation for λ_t:

Lemma 5.1. Let $t \geq 0$. If there exists an edge $\{k, t+1\} \in E(S)$ with $k \leq t$, then

$$\lambda_{t+1} = R_{t+1}(t+1)! \frac{1}{2t-1} \lambda_t$$

Otherwise,

$$\lambda_{t+1} = R_{t+1}(t+1)! \left(1 + \frac{C_S(t+1)}{2t-1}\right) \lambda_t$$

Proof:

Letting

$$Y = \prod_{ij \in E(S_{t+1})} \mathbb{I}_{\{ij \in E(T_{t+1})\}} \prod_{i \in V(S), i \leq t} [d_{t+1}(i)]_{R_t(i)}$$

and noting that $[d_{t+1}(t+1)]_{R_{t+1}(t+1)} = R_{t+1}(t+1)!$, we can write $X_{t+1} = R_{t+1}(t+1)!\, Y$.

First, consider the case when there is no edge $\{k, t+1\} \in E(S)$. In this case, it can be verified

that: (1) $S_{t+1} = S_t$; (2) For each $f_{t+1} \notin V(S)$ with $i \le t$, $R_{t+1}(i) = R_t(i)$; and (3) $\mathbb{I}_{\{ij \in E(T_{t+1})\}} = \mathbb{I}_{\{ij \in E(T_t)\}}$. As a result

$$Y = \prod_{ij \in E(S_t)} \mathbb{I}_{\{ij \in E(T_t)\}} \prod_{i \in V(S), i \le t} [d_{t+1}(i)]_{R_t(i)}$$

i.e., Y is the same as X_t with $d_t(i)$ replaced by $d_{t+1}(i)$.

Fix T_t (and hence X_t). Let f_{t+1} be the random variable denoting the parent of $t+1$. There are two possibilities: If $f_{t+1} \notin V(S)$, then $Y = X_t$. Otherwise, $d_{t+1}(j) = d_t(j) + 1$, which implies

$$[d_{t+1}(j)]_{R_t(j)} = \frac{d_t(j) + R_t(j)}{d_t(j)} [d_t(j)]_{R_t(j)}$$

and thus for a fixed j

$$Y - X_t = X_t \frac{R_t(j)}{d_t(j)}$$

It follows that

$$\begin{aligned}
\mathbb{E}(Y - X_t \mid T_t) &= \sum_{j \in V(S), j \le t} X_t \frac{R_t(j)}{d_t(j)} \mathbb{P}[f_{t+1} = j \mid T_t] \\
&= \sum_{j \in V(S), j \le t} X_t \frac{R_t(j)}{d_t(j)} \frac{d_t(j)}{2t-1} \\
&= \frac{X_t}{2t-1} \sum_{j \in V(S), j \le t} R_t(j) \\
&= X_t \frac{C_S(t+1)}{2t-1}
\end{aligned}$$

Taking expectations once more, we get the second result of the Lemma 5.1.

Now, turn to the first case, *i.e.*, when there exists an edge $\{k, t+1\} \in E(S)$. In this case $S_{t+1} = S_t \cup \{k, t+1\}$. Since $\{k, t+1\} \in T_{t+1}$, it follows that

$$\prod_{ij \in E(S_{t+1})} \mathbb{I}_{\{ij \in E(T_{t+1})\}} = \prod_{ij \in E(S_t)} \mathbb{I}_{\{ij \in E(T_t)\}}. \qquad (5.1)$$

Furthermore, for $i \leq t,\ i \neq k$ we can write $d_{t+1}(i) = d_t(i),\ R_{t+1}(i) = R_t(i)$, while $d_{t+1}(k) = d_t(k) + 1,\ R_{t+1}(k) = R_t(k) - 1$. From these equalities we derive that

$$\prod_{i \in V(S), i \leq t} [d_{t+1}(i)]_{R_{t+1}(i)} = \frac{1}{d_t(k)} \prod_{i \in V(S), i \leq t} [d_t(i)]_{R_t(i)} \qquad (5.2)$$

Observing that $f_{t+1} = k$ with probability $d_t(k)/2t - 1$, and using Equations (5.1), (5.2) we get that $Y = X_t / d_t(k)$ with probability $d_k(t)/2t - 1$. Taking conditional expectations we get

$$\mathbb{E}[Y \mid T_t] = \frac{X_t}{d_t(k)} \frac{d_t(k)}{2t-1} = \frac{X_t}{2t-1}$$

and taking expectations in the last equation we get the first result of Lemma 5.1.

Theorem 5.1 gives closed-form expressions for $\mathbb{P}(S \subset T_n)$, derived by using the recurrence relation of Lemma 5.1. In this theorem, $V^+(S)$ denotes the set of nodes of S that have outgoing arcs, and $V^-(S)$ the set of nodes of S that have incoming arcs.

Theorem 5.1. Let S be a possible subgraph of T_n. Then

$$\mathbb{P}(S \subset T_n) = \prod_{i \in V^-(S)} d_S^-(i)! \prod_{i \in V^+(S)} \frac{1}{2i-1} \prod_{i \notin V^+(S)} \left(1 + \frac{C_S(i)}{2i-1}\right)$$

$$\mathbb{P}(S \subset T_n) = \prod_{i \in V^-(S)} d_S^-(i)! \prod_{ij \in E(S)} \frac{1}{2\sqrt{ij}} \exp\left(O\left(\sum_{i \in V(S)} \frac{C_S^2(i)}{i}\right)\right)$$

Triangles and Transitivity

By applying Theorem 5.1, it is relatively straightforward to derive expressions for the expected number of triangles and paths of length two in G_t for the case of LCD model. Denoting

by N_{t,C_3} the number of triangles in G_t, the following theorem holds:

Theorem 5.2. Let $\varepsilon \geq 1$ be fixed. Then

$$\mathbb{E}\left(N_{t,C_3}\right) \sim \frac{\varepsilon(\varepsilon-1)(\varepsilon+1)}{48}(\log t)^3$$

Similarly, let N_{t,P_2} denote the number of paths of length two in G_t. The following theorem holds:

Theorem 5.3. Fix $\varepsilon \geq 1$, $\delta > 0$. Then:

$$(1-\delta)\frac{\varepsilon(\varepsilon+1)}{2} t\log t \leq N_{t,P_2} \leq (1+\delta)\frac{\varepsilon(\varepsilon+1)}{2} t\log t$$

The proofs for the last two theorems appear in [BR04]. An immediate corollary of the previous two theorems is that the expected value of the transitivity of graph G_t is given by

$$\mathbb{E}[T(G_t)] \sim \frac{\varepsilon-1}{8}\frac{(\log t)^2}{t}$$

Paths and Cycles

Theorem 5.4 and 5.5 appear in [BR04]. Let N_{t,C_l} denote the number of cycles of length l in G_t. Then

Theorem 5.4. Let $l \geq 3$. Then

$$\mathbb{E}(N_{t,C_l}) = \Theta(\varepsilon^l)(\log t)^l$$

A similar result was independently derived by Bianconi *et al.* [BC03, Bia04]. Now, let N_{t,P_k} denote the number of paths of length k in G_t.

Theorem 5.5. Suppose $k = k(t)$ satisfies $k(t)/\log t \to \alpha$ where $0 < \alpha < e$. Then

$$\mathbb{E}\left(N_{t,P_k}\right) = \Theta\left(\frac{t^{1+\alpha\log(e/\alpha)}}{\sqrt{\log t}}\right)$$

Furthermore, if $k(t) = \log t + x\sqrt{\log t}$ where $x = x(t) = o(\log t)$, then

$$E(N_{t,P_k}) = \frac{t^2}{2}\frac{1}{\sqrt{2\pi\log t}}e^{-x^2/2}$$

as $t \to \infty$.

Note that the second statement of the theorem means that the distribution of the path lengths is asymptotically normal with mean and variance $\log t$.

COPY model

Consider now the COPY model of Kumar *et al.* [KKRT00] which was described in Section 2.2. This model was partly motivated by a desire to account for the large number of complete bipartite subgraphs found empirically in the Web graph (see Section 5.2). Kumar *et al.* [KKRT00] showed that the number of complete bipartite subgraphs in the COPY model is also large. This result in stated in Theorem 5.6.

Theorem 5.6. Let $N_{t,K_{i,d}}$ denote the number of complete bipartite subgraphs $K_{i,d}$ at time t. Then, for $i \leq \log t$

$$N_{t,K_{i,d}} = \Omega(te^{-i})$$

Proof.

Consider the node v_τ born at time $\tau \le t$. Call this node a *leader* if at least one of its neighbors is chosen uniformly at random and a *duplicator* if all of its neighbors are copied from some other node. It is easy to notice that $\mathbb{P}\left(\left\{v_\tau \text{ is a leader}\right\}\right) = 1-(1-\alpha)^d$ and $\mathbb{P}\left(\left\{v_\tau \text{ is a duplicator}\right\}\right) = (1-\alpha)^d$. Assume that v_τ is a leader and consider the sequence of epochs $(\tau,2\tau],(2\tau,4\tau],\ldots,(\frac{t}{2},t]$. Let $A_{(\tau,2\tau]}$ be the event that at least one node born during the epoch $(\tau,2\tau]$ chooses v_τ as prototype. Then:

$$\mathbb{P}(A_{(\tau,2\tau]}) \ge 1-\prod_{\tau'=1}^{\tau}\left(1-\frac{1}{\tau+\tau'}\right) \approx 1/2$$

The same is true for all other epochs $(2\tau,4\tau],\ldots,(t/2,t]$. Hence, the expected number of duplicators of v_τ up to time t is $\Omega(\ln(t/\tau))$. Note, that a $K_{d,\Omega(\ln\{t/\tau\})}$ forms between the duplicators of v_t and its neighbors. Now, suppose that $i \le \log t$ and let $\tau = te^{-i}$. The preceding arguments imply that the expected number of duplicators of v_τ is i. Hence, for each of the nodes $v_1,\ldots,v_{te^{-i}}$ there is at least one $K_{d,i}$ and thus $N^t_{K_{i,d}} = \Omega(te^{-i})$.

In the same paper, Kumar *et al.* [KKRT00] proved that the expected number of complete bipartite cliques in some other models (including a growing uniform random graph model and a random graph with given degree distribution) is very small.

5.2. Counting Communities by Trawling

In this section, the Web is considered as a directed graph and the notation $C_{i,j}$ is used to denote a *bipartite core—i.e.*, a directed graph on $i + j$ nodes that contains at least one complete bipartite graph $K_{i,j}$ as a subgraph.

Kumar *et al.* [KRRT99] used a number of empirical observations to devise an efficient procedure for extracting bipartite cores from a subgraph of the Web with approximately 200 million web pages. The problem of enumerating the small subgraphs of a large, web-like graph is now commonly referred to as *trawling*—a term first used in [KKRT99]. Note that a trawling algorithm must take into account the fact that the data that represents the web-like network generally would not fit in main memory. The trawling methodology devised by Kumar *et al.* is called *elimination-generation*. The input to the elimination-generation trawling algorithm is a subgraph of the Web obtained via crawling and stored in disk as an edge-list. The algorithm performs several passes over the data. During each pass, it writes a modified version of the dataset to disk for the next pass. It also collects some metadata which resides in main memory and serves as state in the next pass. Passes over the data are interleaved with sort operations, which constitute the bulk of the processing cost. Two processes, *elimination* and *generation*, are interleaved between passes.

Elimination: There are many necessary conditions for a node to be in a bipartite core. Take as an example $C_{4,4}$: Any node with in-degree three or less can not participate on the right of a $C_{4,4}$. Likewise, nodes with out-degree three or smaller cannot participate on the left side of a $C_{4,4}$. Thus edges that are directed into such nodes can be pruned from the graph. These necessary conditions are called *elimination filters*.

Generation: Generation is counterpoint to elimination. Nodes that barely qualify for potential membership in a $C_{i,j}$ can be easily verified to either belong in such a core or not. Consider again $C_{4,4}$: Let u be a node of in-degree exactly four. Then, u belongs to a $C_{4,4}$ if and only if the four nodes that point to it have a neighborhood intersection of size at least 4. This can be verified cheaply. A *generation filter* is a procedure that identifies barely-qualifying nodes, and for all such nodes, either outputs a core or proves that such a core does not exist. Regardless of the outcome, the node can be pruned since all potential interesting cores containing it have already been enumerated.

The *sorting* of edges by the first (or the second) node, is essential so that filtering can be applied in a single scan. Details of how this can be implemented may be found in [KKRT99]. After an elimination/generation pass, the remaining nodes have fewer neighbors than before in the residual graph, which may present new opportunities during next pass. Depending on the filters, one of two things will eventually happen: (1) all the nodes will be removed until nothing is left; and (2) after several passes, the benefits of "elimination/generation" tail off as fewer and fewer nodes are eliminated at each phase. In the experiment by Kumar *et al.* [KKRT99] the second phenomenon dominates. Running the trawling algorithm on a crawl of the Web with 200 million Web pages, Kumar *et al.* found over $100,000$ bipartite cores, some being as large as $C_{6,9}$. Interestingly, even the smallest identified cores ($C_{3,3}$ and $C_{3,5}$) were topically focused on an identifiable theme in 96% of the sampled examples. Hence, the identified cores were usually topically focused and so specific that they were often not part of any preexisting portal hierarchy. This last point is important because it means that cores are “natural” in the sense that they are self-organized, and not an artifact of a single individual entity.

5.3. Estimating the Density of Communities by Sampling

This section begins by discussing a sampling algorithm devised by Kashtan *et al.* [KIMA04]. Then it describes a proposed improvement to this algorithm and finally presents several experimental results obtained by applying the improved sampling algorithm to a large web-like network.

Sampling Algorithm

The sampling procedure proposed by Kashtan *et al.* [KIMA04] is shown in Algorithm 5.1 below:

Algorithm 5.1: **SUBGRAPH-DENSITY**

Input: $G(V, E)$: an undirected graph
$SampleSize$: an integer
$NumberOfSamples$: an integer
$Type$: the type of subgraph being sampled

Output: An estimate for the density of subgraphs of type $Type$ in G

1. **real**: $SubgraphWeight$, $TotalWeight$, P
2. **graph:** $G_S(V_S, E_S)$
3. $Weight = 0$
4. **FOR** $i = 1$ **TO** $NumberOfSamples$ **DO**
5. GENERATE-RANDOM-SAMPLE ($G, SampleSize, G_S$)
6. $P =$ GET-PROBABILITY-SAMPLE($G, SampleSize, G_S$)
7. $TotalWeight = TotalWeight + 1/P$
8. **IF** (IS-OF-TYPE($G_S, Type$)) **THEN**
9. $SubgraphWeight = SubgraphWeight + 1/P$
10. **END IF**
11. **END FOR**
12. **RETURN** $SubgraphWeight \,/\, TotalWeight$

As seen, the procedure SUBGRAPH-DENSITY generates a user-specified number of random samples (subgraphs) and for each generated sample checks if it is of a given type. The output of this procedure is an estimate for the density of subgraphs of type $Type$ on $SampleSize$ nodes. Here, the *density* of a subgraph of type T and size S is defined as the ratio of the number of subsets of nodes of cardinality S that induce subgraphs of type T with the number of all connected subgraphs on S nodes. Key to the procedure SUBGRAPH-DENSITY are the functions GENERATE-RANDOM-SAMPLE which is called to generate a random subgraph $G_S(V_S, E_S)$ and GET-PROBABILITY-SAMPLE which determines the probability that the sampling procedure generates a specific subgraph G_S.

Algorithm 5.2 shows the pseudo-code for the procedure GENERATE-RANDOM-SAMPLE. This procedure starts out by selecting an edge e uniformly at random from the graph G and then constructs a *tree* with $SampleSize$ nodes and $SampleSize - 1$ edges. The first edge of this tree is the edge e and the rest of the tree is constructed by selecting at each step $i = 2,\ldots,SampleSize$ an edge uniformly at random from the neighborhood of the tree constructed up to the step $i-1$. Example 5.1 illustrates the idea of the procedure GENERATE-RANDOM-SAMPLE.

Example 5.1. Consider the graph in Figure 5.1 and assume hat the first edge

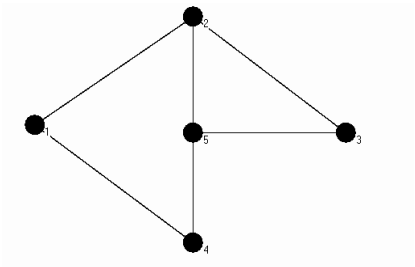


Figure 5.1. A graph on five nodes.

is chosen uniformly at random to be the edge $(2,5)$.Then the second edge will be selected uniformly at random from the set $\{(1,2),(2,3),(3,5),(4,5)\}$. Suppose that $(2,3)$ is chosen as the second edge. Then the third edge will be selected uniformly at random fro the set $\{(1,2),(4,5)\}$, *etc*.

Algorithm 5.2: **GENERATE-RANDOM-SAMPLE**

Input: $G(V,E)$: an undirected graph
$SampleSize$: an integer
Output: $G_S(V_S,E_S)$: a random sample

1. **edge:** e
2. $e=(u,v) \leftarrow$ a randomly chosen edge from G
3. $V_S=\{u,v\},\ E_S=\{e\}$
4. $L=(\{\text{edges incident on } u\}\cup\{\text{edges incident on v}\})-\{e\}$
5. $i=1$
6. **WHILE** $i<SampleSize$ **DO**
7. $\quad e=(u,v) \leftarrow$ a randomly chosen edge from L; assume $u\in V_S, v\notin V_S$
8. $\quad V_S \leftarrow V_S\cup\{v\}$
9. $\quad E_S=E\cup\{e\}$
10. $\quad L=L\cup\{\text{edges incident on } v\}-\{\text{edges with both ends in } V_S\}$
11. $\quad$ **IF** $L=\varnothing$ **THEN**
12. $\quad\quad$ **GO TO** Step 2
13. $\quad$ **ELSE**
14. $\quad\quad i=i+1$
15. $\quad$ **END IF**
16. **END WHILE**
17. **RETURN** G_S

Of course, not every tree on $SampleSize$ nodes in the graph G has the same probability of being generated. As a result it is necessary to compute for each generated tree the probability that the sampling procedure would generate that specific tree. Algorithm 5.3 shows the pseudo-

code for a procedure that computes the probability that the sampling procedure GENERATE-RANDOM-SAMPLE will generate a specific fixed tree.

Algorithm 5.3: **GET-PROBABILITY-SAMPLE**

Input: $G(V, E)$: an undirected graph
$SampleSize$: an integer
$G_S(V_S, E_S)$: a subgraph of G that may be generated by the sampling procedure GENERATE-RANDOM-SAMPLE

Output: P : probability that the sampling procedure generates G_S

1. **array**: $l(1, SampleSize)$
2. $P = 0$
3. **FOR EACH** permutation σ of the set E_S **DO**
4. **FOR** $i = 2$ **TO** $SampleSize$ **DO**
5. $l[i]$= size of set L before selecting the i^{th} edge during a sampling of subgraph G_S in the sequence specified by σ
6. **END FOR**
7. $p = 1 / |E|$
8. **FOR** $i = 2$ **TO** $SampleSize$ **DO**
9. $p = p * 1 / l[i]$
10. **END FOR**
11. $P = P + p$
12. **END FOR**
13. **RETURN** P

The computation in the procedure GET-PROBABILITY-SAMPLE is illustrated in Example 5.2, below:

Example 5.2: Consider the graph G show in Figure 5.2 and the subgraph G' induced by edges $(3,5),(5,6)$ and $(6,8)$. What is the probability that the sampling procedure GENERATE-RANDOM-SAMPLE will generate G' as a sample? To compute this probability, first we compute for each fixed permutation of the set of edges $\{(3,5),(5,6),(6,8)\}$ the probability that

the sampling procedure will generate these three edges in the sequence specified by the fixed permutation and then we add these individual probabilities together.

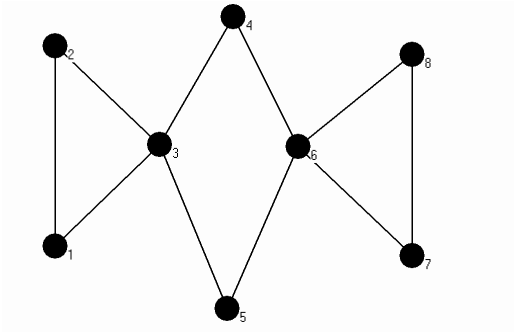


Figure 5.2. A graph with eight nodes and ten edges.

It is easy to see that for two of the six permutations—that is, for the permutations $(3,5),(6,8),(5,6)$ and $(6,8),(3,5),(5,6)$—this probability is zero because it is not possible to generate this triple of edges by using our sampling procedure in the sequence specified by any of these two permutations. On the other hand, the probabilities for the remaining four permutations are as follows:

- for $(3,5),(5,6),(6,8)$, $p = 1/10 \times 1/4 \times 1/6 = 1/240$
- for $(6,8),(5,6),(3,5)$, $p = 1/10 \times 1/4 \times 1/4 = 1/160$
- for $(5,6),(3,5),(6,8)$, $p = 1/10 \times 1/4 \times 1/6 = 1/240$
- for $(5,6),(6,8),(3,5)$, $p = 1/10 \times 1/4 \times 1/4 = 1/160$

By adding these four probabilities together we find that the probability that our sampling procedure will generate the subgraph G' is $5/240$.

Proposed Improvement of the Sampling Procedure

As indicated in Example 5.2, for a given a tree $G_S(V_S, E_S)$ which is a subgraph of a graph G, only some of the permutations of the set of edges E_S will specify sequences in which it is possible to generate G_S. We say that a particular permutation σ of the edges of set E_S is *feasible* if it is possible that the sampling procedure generates the edges of G_S in the sequence specified by σ. Consider, for instance, the subgraph induced by the edges $\{e_1 = (3,5), e_2 = (5,6), e_3 = (6,8)\}$ in the graph shown in Figure 5.2. Among the six possible permutations of these three edges, two are not feasible (permutations $\{e_1, e_3, e_2\}$ and $\{e_3, e_1, e_2\}$) while the remaining four are feasible.

It may be seen that a permutation σ of the set $E_S = \{e_1, \ldots, e_{SampleSize-1}\}$ is feasible if and only if the subgraph induced by the edges $e_{\sigma(1)}, \ldots, e_{\sigma(j)}$ is *connected* for all $j = 1, \ldots, SampleSize - 1$.

This observation indicates that the for-loop in Line 3 of the procedure GET-PROBABILITY-SAMPLE does not need to iterate over all permutations of the set E_S but only over *all feasible* permutations.

How much is the performance of the procedure GET-PROBABILITY-SAMPLE improved by applying this change? In order to answer this question we consider first the best and the worst inputs for the improved procedure GET-PROBABILITY-SAMPLE. It may be seen that the worst-case input is a tree of diameter one consisting of a node u to which all other nodes are linked with an edge (*i.e.*, a *star* graph). Every permutation of this tree is feasible and thus the number of feasible permutations is $(SampleSize - 1)!$. On the other hand, the best-case input is a path of length $SampleSize - 1$. In this case it is easy to see that the number of feasible

permutations is $2^{SampleSize-2}$. In order to estimate the average number of feasible permutations for a random sample, we used simulation. Table 5.1 shows the average number of feasible permutations for trees with $5,\ldots,10$ nodes (in the 3rd column). For a fixed number of nodes, the values shown in the 3rd column of Table 5.1 were averaged over 100 runs. For comparison, this table also shows the number of feasible permutations for the best case (2nd column) and worst case (4th column). As seen, the average case is much closer to the best case than it is to the worst case.

Table 5.1. Number of feasible permutations for the best-, average-, and worst-case sample inputs.

$SampleSize$	$2^{SampleSize-2}$	$Average$	$(SampleSize-1)!$
5	8	14	24
6	16	47	120
7	32	185	720
8	64	1041	5040
9	128	5397	40320
10	256	43330	362880

We applied the improved sampling procedure SUBGRAPH-DENSITY to study the density of near-cliques in a large real-life web-like network. The obtained results are presented next.

Case study: *FOLDOC Network*

The real-life data set that was used in our sampling experiments is the network representation of Free OnLine Computing Dictionary (FOLDOC). FOLDOC is a searchable dictionary of terms related to computing such as acronyms, jargon, programming languages, tools, architecture, operating systems, networking, theory, conventions, standards, companies, projects, products, history, *etc*.

The dictionary has been growing since 1985 and now contains over 13000 definitions totaling nearly five megabytes of text. Entries of this dictionary cross-reference each other and related resources elsewhere on the net. The nodes in the network representation of FOLDOC represent *terms*. An arc (u, v) means that the term v is used to describe the meaning of term u.

The graph representation of FOLDOC network was obtained from the web site of the network visualization tool Pajek (see Appendix A). This graph has 13356 nodes and 120238 directed arcs. For convenience, we converted this directed graph into an undirected graph by ignoring the orientation of arcs. We then merged the parallel arcs formed as a result of this process, thereby reducing the number of edges to 91465. All experiments discussed next refer to this undirected version of the FOLDOC network.

First, we computed several global parameters of the FOLDOC graph. The values of these parameters are shown in Table 5.2. Notice that this network satisfies the salient properties of web-like networks: it has a power-law degree distribution (Figure 5.3) with exponent $\gamma = 3$, a small average distance ($\mathrm{dist}(G) = 5.85$) and a large value of clustering coefficient ($C(G) = 0.3379$).

Table 5.2. Some parameters of the FOLDOC network.

Parameter	**Value**
n	13356
m	91465
$\min d(v)$	2
$\max d(v)$	728
$d(G)$	13.697
$\mathrm{dist}(G)$	5.85
γ	3.0
$\min C(v)$	0
$\max C(v)$	1
$C(G)$	0.3379

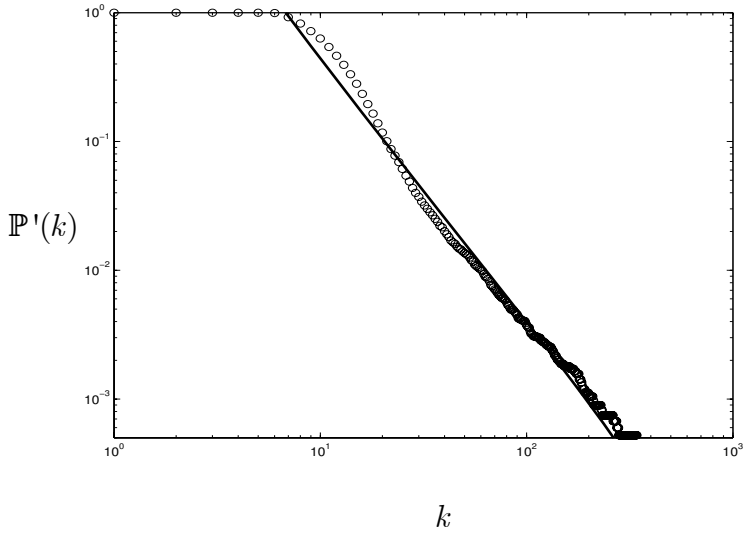


Figure 5.4. Log-log plot of the cumulative degree distribution of the FOLDOC network.

A) Some examples of communities discovered through sampling

First we used the sampling procedure to determine if subgraphs with high values of clustering coefficient consist of nodes with related meaning. Figures 5.4 and 5.5 show two subgraphs with high values of clustering coefficient that were found during sampling.

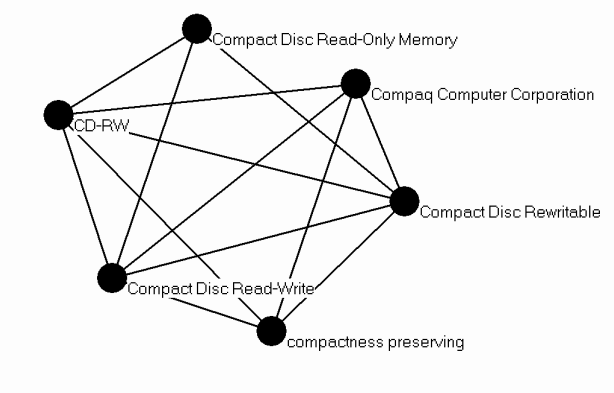


Figure 5.4. A sampled graph on six nodes. Minimum clustering coefficient is 0.5 and average clustering coefficient is 0.8.

In the first example it may be seen that all the terms except one ("compactness preserving") are indeed closely-related terms. The graph in the second example consists of nodes that belong to two different communities. This is an interesting example which highlights the idea that groups of nodes that satisfy the definition of $\alpha\text{-}near\text{-}clique$ for large values of parameter α might be unions of nodes from several communities.

The two examples given in Figures 5.4 and 5.5 are representative of the groups of nodes with high clustering coefficient that we inspected visually by using Pajek visualization tool. It should be mentioned that in some cases the nodes of a sample with high clustering coefficient

did not seem to be related to each-other. However, the reason for this unexpected result was traced back to errors in the input graph FOLDOC.

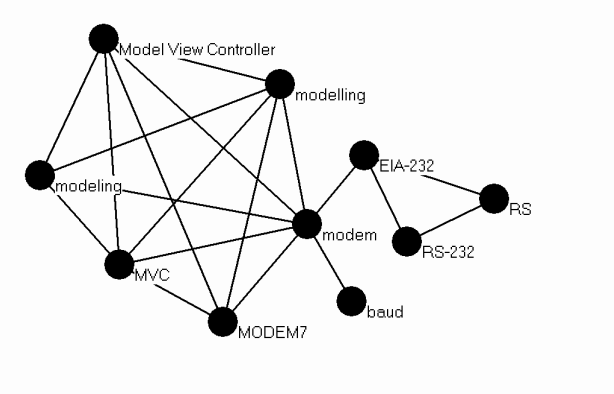


Figure 5.5. A sampled graph on ten nodes. The average clustering coefficient of this graph is 0.7.

B) Density of α-$near$-$cliques$

Second, we used the sampling procedure to determine the density of α-$near$-$cliques$. The results of these sampling experiments are shown in Figure 5.6. This figure shows the density of subgraphs with six nodes and with *minimum* clustering coefficient in the range $[i, i+2)$ for $i = 0,\ldots,8$. The densities were computed after one, two, …, five thousand samples were taken. For clarity the density values for the range $[0, 0.2)$ have been omitted from Figure 5.6 because they were always greater then 0.99, that is much larger than the values corresponding to the other four ranges.

Figure 5.6 indicates that after 30000 samples the density values converge. This observation agrees with the convergence results in [KIMA04] were it was noticed that the densities for many types of small subgraphs will converge after $5000 - 50000$ samples.

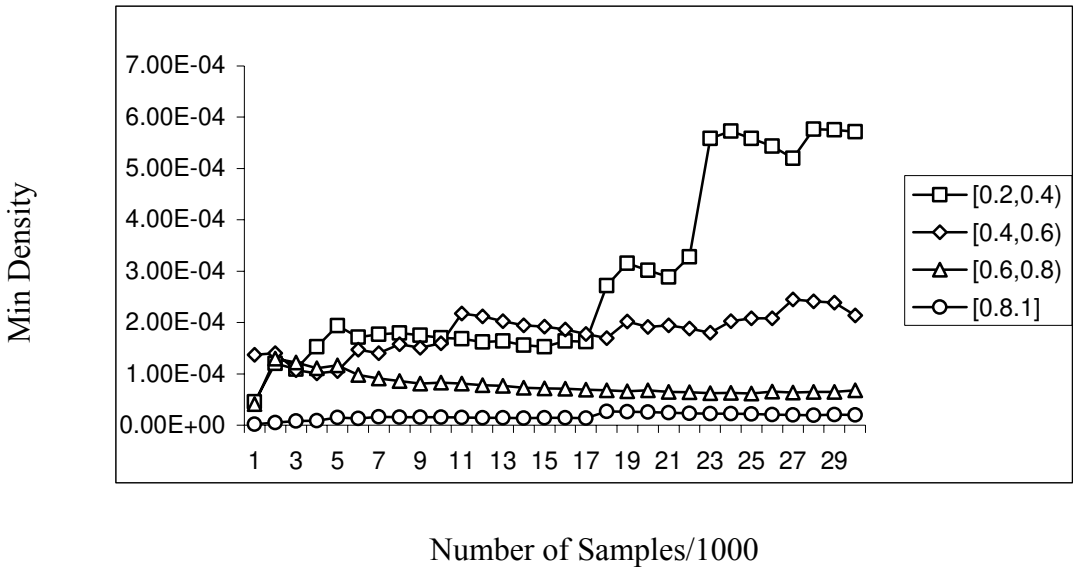


Figure 5.6. The density of subgraphs with six nodes versus the minimum clustering coefficient in these subgraphs.

Figure 5.7 shows the density of subgraphs with six nodes and with *average* clustering coefficient in those ranges. In this case, for clarity, the density values for the bottom range $[0, 0.2)$ have been omitted because they are too small compared to the values for the other three ranges. Again after 30000 samples the density values appear to converge.

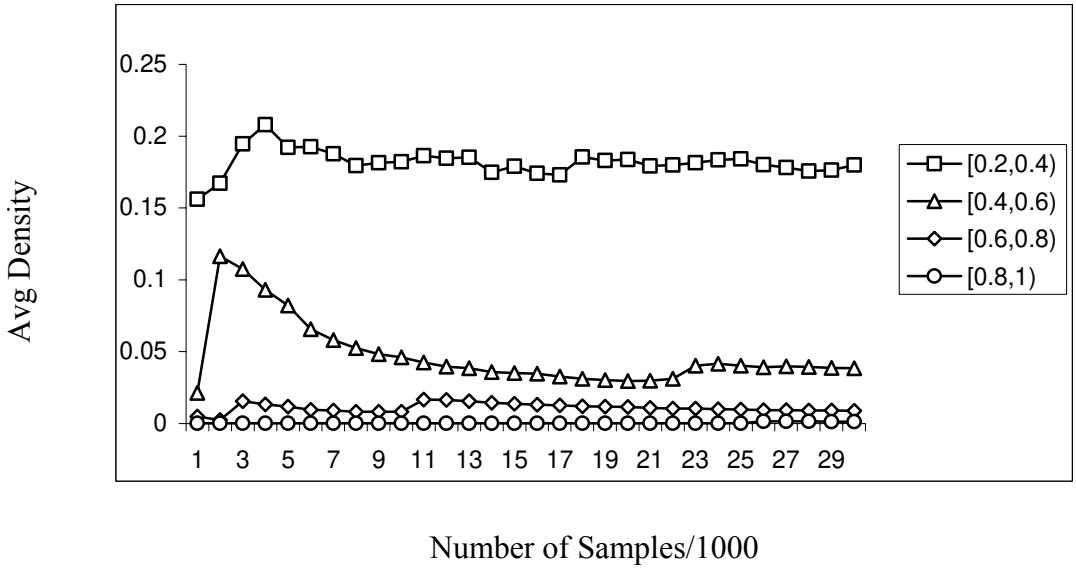


Figure 5.7. The density of subgraphs with six nodes versus the average clustering coefficient in these subgraphs.

The sampling experiments presented in this section show that sampling is a very useful tool for (1) visually inspecting groups of the nodes that satisfy a certain definition of community, and (2) for investigating the density of various types of community.

6. EXISTING ALGORITHMS FOR COMMUNITY MINING

Recently, several community-mining algorithms spanning a wide spectrum of techniques have been proposed. These algorithms are surveyed next.

6.1. Algorithms Based on Hierarchical Clustering

The algorithms in this group employ the technique of *hierarchical clustering*, which is essentially based on the computation of certain measures of "similarity" between distinct nodes and may be performed in either a bottom-up or a top-down fashion. An *agglomerative* hierarchical clustering algorithm begins with each node in a separate cluster, and then iteratively pulls together the two clusters that are the most similar, in a bottom-up fashion,. Two measures of similarity borrowed from the field of bibliometrics—*bibliographic* and *co-citation* coupling—have been used in some community-mining agglomerative clustering algorithms [HKKS04, BD05].

In contrast with agglomerative clustering, a *divisive* clustering algorithm follows a top-down approach to iteratively identify pairs of adjacent nodes that are most "dissimilar", and remove the edge(s) between them. Usually, the iteration ends when the graph breaks into disconnected components, which then represent the desired clusters. A measure of similarity proposed by Girvan and Newman [GN02], called the "*edge betweenness*"—the number of shortest paths passing through an edge—has gained some popularity due do its intuitive appeal and simplicity. This algorithm provides a good illustration of the divisive clustering techniques and therefore it is described in detail next.

Recall that the *load*, or *betweenness*, of a node v in an undirected graph is defined as the

number of shortest paths passing through v. Similarly, the load of an edge e is defined as the number of shortest paths passing through e. The intuition behind the Girwan-Newman algorithm is that in a network with communities, the edges between communities can be thought as forming "bottlenecks" in the sense that most shortest paths will go through them. Therefore removing the edges with the highest load should split the network into natural communities.

The pseudocode of the procedure by Girwan and Newman [GN02] is shown in Algorithm 6.1, below.

Algorithm 6.1: GIRWAN-NEWMAN

Input: $G = (V, E)$: an undirected graph
Output: C : a "dendrogram" of communities

13. **edge**: e
14. **WHILE** $E \neq \varnothing$ **DO**
15. $e \leftarrow$ MAX-LOAD-EDGE (G)
16. remove e from G
17. **END**

The main loop of the algorithm is very simple: in each step, the edge with the highest load is found and removed from the graph. The procedure MAX-LOAD-EDGE is called to find the edge with the highest load. In turn, MAX-LOAD-EDGE calls SINGLE-SOURCE-NODE-LOAD from each node. Algorithm 6.2 shows the pseudocode for SINGLE-SOURCE-NODE-LOAD which is implemented by a simple modification of Breadth-First Search (BFS) proposed in [GN02].

The time complexity of procedure SINGLE-SOURCE-NODE-LOAD is of order $O(m+n)$ because it involves a single run of BFS. Algorithm 6.3 shows the pseudocode for the MAX-LOAD-EDGE procedure. The MAX-LOAD-EDGE calls SINGLE-SOURCE-NODE-

LOAD for each node and therefore its time complexity is of order $O(mn)$. Finally, the GIRWAN-NEWMAN algorithm calls MAX-EDGE-LOAD after the removal of each edge, and thus its time complexity is of order $O(m^2n)$, which is prohibitively slow for analyzing large web-like networks.

Algorithm 6.2: SINGLE-SOURCE-NODE-LOAD

Input: G = (V, E) : an undirected graph
$s \in V$: a node
Output: the BFS tree and the load for each node

1. **array**: d, load
2. $d(s) \leftarrow 0$
3. $load(s) \leftarrow 1$
4. **FOR ALL** nodes i adjacent to s **DO**
5. $d(i) \leftarrow 1$
6. $load(i) \leftarrow 1$
7. **END FOR**
8. **REPEAT**
9. **FOR ALL** nodes j adjacent to one of i **DO**
10. **IF** j has not been assigned a distance **THEN**
11. $d(j) = d(i) + 1$
12. $load(j) = load(i)$
13. **END IF**
14. **IF** j has already been assigned a distance **AND** $d(j) = d(i) + 1$ **THEN**
15. $load(j) \leftarrow load(j) + load(i)$
16. **END IF**
17. **END FOR**
18. **UNTIL** there are no nodes that have been assigned distances but whose neighbors have not been assigned distances
19. **RETURN** d, $load$

Another measure of similarity, called the "*edge clustering coefficient*"—the analog of the node clustering coefficient (see Section 1.1)—was proposed by Radichi *et al.* [RCCL04]. Castellano *et al.* [CCLP04] combined a divisive clustering technique with a formal definition of

a community as a group of nodes where each member has more neighbors inside the group than outside it (*i.e.*, a defensive alliance).

Algorithm 6.3: MAX-EDGE-LOAD

Input: $G = (V, E)$: an undirected graph
Output: the edge with the highest load

1. **array:** $load$
2. **FOR ALL** $e \in E$ **DO**
3. $load(e) \leftarrow 0$
4. **END FOR**
5. **FOR ALL** $s \in V$ **DO**
6. $T \leftarrow$ SINGLE-SOURCE-NODE-LOAD(G, s)
7. $D \leftarrow$ depth(T)
8. **FOR ALL** nodes i at level $D-1$ neighboring leaf t **DO**
9. $load(i,t) \leftarrow load(i)/load(t)$
10. **END FOR**
11. **FOR** $l = D-2$ **DOWNTO** 0 **DO**
12. **FOR ALL** edges (i, j) such that j is at level l **AND**
13. i at an upper level **DO**
14. $load(i,j) \leftarrow [1 + \sum_{k \text{ lower than } i} load(k,i)](load(i)/load(j)$
15. **END FOR**
16. **END FOR**
17. **END FOR**
18. **RETURN** an edge with maximum load

Clustering algorithms produce groups of nodes that are densely linked with each other while being sparsely linked with the rest of the nodes. However, these algorithms have considerable time demand, which limits their application to networks of moderate size.

6.2. Algorithms Based on Spectral Analysis

This section discusses some global methods that essentially consider all edges of a graph

to decide on the similarity between two nodes. First, let us recall some definitions from linear algebra:

Any non-singular $n \times n$ matrix M can be represented as summation of vector outer-products:

$$M = \sum_{i=1}^{k} \lambda_i \mathbf{r}_i \mathbf{l}_i^T$$

where $\mathbf{l}_i$ and $\mathbf{r}_i$ are, respectively, the i^{th} left and right eigenvectors of M and λ_i is the i^{th} eigenvalue of M. The matrix M has all of the following properties:

$$\begin{aligned}
&\lambda_i \mathbf{r}_i = M\mathbf{r}_i, \\
&\lambda_i \mathbf{l}_i = M^T \mathbf{l}_i, \\
&\mathbf{l}_i^T \mathbf{l}_i = \mathbf{r}_i^T \mathbf{r}_i = \mathbf{r}_i^T \mathbf{l}_i, \quad \text{for all } i \\
&\mathbf{l}_i^T \mathbf{r}_j = 0, \quad \text{for } i \neq j \\
&\lambda_i \geq \lambda_{i+1}.
\end{aligned}$$

The eigenvalues and eigenvectors form the *spectrum* of a matrix. If the spectrum of a matrix is *full* (*i.e.,* it contains n distinct eigenvectors), then either the left or the right eigenvectors can be used as a basis to express any n-dimensional vector. If M is *symmetric*, then the left and right eigenvectors of M are identical.

Probably, the most famous algorithm that uses spectral techniques is PageRank [PBMW98]. The main objective of this algorithm was to remedy the “abundance problem" inherent in broad search engine queries. To achieve this objective, PageRank assigns to each web site a measure of *prestige* which is independent of any information need or query. In simple terms, the prestige of a web site is proportional to the sum of prestige scores of pages linking to it.

The earliest applications of spectral techniques for mining communities are Kleinberg's HITS (Hypertext Induced Topic Search) algorithm [Kle99] and its variations [BH98, DH99]. HITS algorithm is described next. The intuition behind this algorithm comes from the observation that, as in the academic literature where some publications initiate new ideas, while others consolidate and survey significant research, the Web includes two flavors of prominent web pages: *authorities*, which contain definitive high-quality information, and *hubs*, which are comprehensive lists of links to authorities. Every page is to some extent both a hub and an authority, but these properties are graded. Thus, every page v has two distinct measures of merit, its *hub score* $h[u]$ and its *authority score* $a[v]$.

HITS operates in two phases. In the first phase a subgraph of the Web that is specific to a query q is constructed as follows: The query q is sent to a search engine and the web pages that constitute the top, say 200, responses from the search engine are said to form the *root set* R. The *base set* V_q is constructed by adding to the root set R all the web pages v, such that for some $u \in R$, at least one of the two arcs uv and vu is an arc of the Web. Arcs that connect web pages from the same web site are eliminated because they are considered "nepotistic". The set of the remaining arcs is denoted by E_q. This process constructs a query-specific subgraph $G_q = (V_q, E_q)$.

In the second phase, the hub and authority scores for all nodes in V_q are computed. Collectively, the scores of all the nodes are written as vectors $\mathbf{a}$ and $\mathbf{h}$. The authority score of a page is proportional to the sum of hub scores linking to it, and conversely, its hub score is proportional to the authority scores of the pages to which it links. Assuming that A is the adjacency matrix of G_q, this translates to the following pair of equations:

$$\mathbf{a} = A^T\mathbf{h},$$
$$\mathbf{h} = A\mathbf{a}.$$

The method of *power iteration* may be used to solve this system of equations, as shown in pseudocode in Algorithm 6.4. It is a well-known fact of linear algebra that using the power iteration shown in Algorithm 6.4, the vector $\mathbf{h}$ will converge to the principal eigenvector of AA^T while the vector $\mathbf{a}$ will converge to the principal eigenvector of A^TA. The error after i iterations, is proportional to $O(|\lambda_2/\lambda_1|^i)$. This procedure tends to be fast for *power-law* graphs which often have the property $\lambda_1 \gg \lambda_2$ [CL03]. Typically, runs with several thousand nodes and links "converge" in 20 to 30 iterations, in the sense that the rankings of hubs and authorities stabilize.

Algorithm 6.4: HITS-SCORES

Input: $G = (V, E)$: an undirected graph
$\mathbf{a}, \mathbf{h}$: vectors
Output: "authority" and "hub" scores of all nodes

1. $\mathbf{a} \leftarrow (1,\ldots,1)$
2. $\mathbf{h} \leftarrow (1,\ldots,1)^T$
3. **WHILE** $\mathbf{h}$ and $\mathbf{a}$ change significantly **DO**
4. $\quad \mathbf{h} \leftarrow A\mathbf{a}$
5. $\quad \mathbf{l}_h \leftarrow || h ||_1 = \sum_v h[v]$
6. $\quad \mathbf{h} \leftarrow \mathbf{h}/\mathbf{l}_h$
7. $\quad \mathbf{a} \leftarrow A^T\mathbf{h}$
8. $\quad \mathbf{l}_a \leftarrow || a ||_1 = \sum_v a[v]$
9. $\quad \mathbf{a} \leftarrow \mathbf{a}/\mathbf{l}_a$
10. **END WHILE**

HITS communities

If the query q that serves as input to HITS is ambiguous (*e.g.*, "jaguar") or polarized

(*e.g.,* “abortion"), the set $V_q - R$ will contain a few almost disconnected communities. In each community there may be dense bipartite subgraphs. In such cases, a few of the highest-order eigenvectors found by HITS will reveal authorities and hubs in the largest near-bipartite component. The highest-order eigenvectors can also be found using an iterative method as follows: Given an $n \times n$ matrix M (say, $M = A^T A$) for which we wish to find k top eigenvectors, we initialize an $n \times k$ matrix X with positive entries. Let $X(i)$ be the i^{th} column of X. The iteration steps are shown in Algorithm 6.5, below.

Algorithm 6.5: HIGHER-ORDER-EIGENVECTORS

Input: $M : n \times n$ matrix
$X : k \times n$ matrix
Output: k top-ranked eigenvalues

1. **WHILE** X does not converge **DO**
2. $X \leftarrow MX$
3. **FOR** $i = 1,\ldots,k$ **DO**
4. **FOR** $j = 1,\ldots,i-1$ **DO**
5. $X(i) \leftarrow X(i) - [X(i)X(j)]X(i)$
6. **END FOR**
7. normalize $X(i)$ to unit L_2 norm
8. **END FOR**
9. **END WHILE**

By computing the k top-ranked eigenvalues, each node will be assigned k hub scores and k authority scores. These scores can be used to discover densely linked communities on the Web. Indeed, by plotting each node as a point in a k dimensional space using its hub or authority scores one can discover points that are close to each other, say by visualization. For example, for $k = 2$, it was found [Kle99] that the pages of the base set belonging to the query “abortion", split into two communities along pro-choice and pro-life camps.

Capocci *et al.* [CSCC04] and Donetti and Muñoz [DM04] have also proposed community-mining algorithms based on spectral techniques.

The advantage of spectral methods is that they are elegant and often produce good results. However, these methods, too, are not applicable to very large networks due to their time complexity (at least *quadratic* in the order of the graph).

6.3. Algorithms based on Flows

The well-known *max-flow/min-cut* algorithm by Ford and Fulkerson lies at the heart of some recent methods for mining Web communities proposed by Flake *et al.* [FLG00, FLGC02]. The basic algorithm proposed by these authors, aims to discover the community to which a given set of web pages belongs. This problem is cast into an *s-t* network flow problem by first constructing a graph G that contains all the neighborhoods of the seed pages up to a certain depth, and then adding two artificial nodes: a *source* node that links to each seed page with an edge of infinite capacity, and a *sink* node which links to every node of the graph with an edge of capacity α—a parameter of the algorithm. The community containing the seed pages is then obtained by running a modified version of the max-flow/min-cut algorithm.

This subsection begins by describing the *s-t* maximum flows and minimum cuts and the Ford-Fulkerson algorithm for solving the *s-t* maximum flow problem. Then, it continues with a description of the algorithms by Flake *et al*. mentioned earlier.

Maximum Flows and Minimum Cuts

While flows and cuts are well-defined for both directed and undirected graphs, we restrict the attention to undirected graphs to simplify the presentation. Note that any directed graph can

be converted into an undirected graph by ignoring the arc orientations. Let $G = (V, E)$ be an undirected graph, and let c, f be two non-negative, real-valued functions, where $c(u, v)$ denotes the *capacity* of the edge (u, v) and $f(u, v)$ denotes the *flow* along the edge (u, v). By convention, if the edge (u, v) is not present, it is assumed that $c(u, v) = 0$. Given two nodes, s and t, the s-t *maximum flow problem* is to find the maximum flow that can be routed from s to t while obeying the constraint $f(e) \leq c(e)$ for every edge e. Ford and Fulkerson's "max-flow/min-cut" theorem, proves that the value of *maximum flow* of a graph is identical to the value of a *minimum cut* that separates s and t. This result can be stated as follows: Let the maximum flow value between s and t be represented as $f(s, t)$. Denote the edge cut that separates s and t by $C(s, t) \subseteq E$. Removing the cut set $C(s, t)$ from E will leave at least two connected components: one that contains s and the other that contains t. Then the maximum flow has the following relationship to the cut set:

$$f(s, t) = \sum_{(u,v) \in C(s,t)} c(u, v).$$

The meaning of functions $c(\cdot)$ and $f(\cdot)$ may be generalized so that their arguments range over sets of nodes. In this case $C(S, T)$ will be the edge-cut set of minimum capacity separating the nodes of S from the nodes of T, and $f(S, T)$ is the maximum flow or minimum cut value between the two sets.

Many polynomial-time algorithms exist for solving the s-t maximum flow problem; the authoritative book on this topic is [AMO93]. Algorithm 6.6 shows the pseudocode for the *augmenting path* algorithm—the simplest known s-t maximum flow algorithm. The procedure operates on a *residual* network, which is a data structure used to keep track of edge capacities, both used and available. The residual network $R = (V, E')$ of graph G has two directed edges

for every undirected edge in E, *i.e.,* for every $(u,v) \in E$, the set E' will contain both (u,v) and (v,u).

Algorithm 6.6: MAX-FLOW

Input: $G = (V,E)$: a weighted graph
s,t: nodes
Output: the residual network of G

```
1.  R ← residual network of G
2.  WHILE R contains a directed path from s to t DO
3.      Identify the shortest augmenting path P, from s to t
4.      δ ← min{r(u,v) | (u,v) ∈ P}
5.      FOR ALL (u,v) ∈ P DO
6.          r(u,v) ← r(u,v) − δ
7.          r(v,u) ← r(u,v) + δ
8.      END FOR
9.  END WHILE
10. RETURN R
```

The residual capacities in R are initialized by $r(u,v) = r(v,u) = c(u,v)$ for all $(u,v) \in E$. The residual network R is said to have an *augmenting path*, from s to t, if there exists a path connecting these two nodes such that each directed edge along the path has a non-zero residual capacity. Line 4 of Algorithm 6.6, identifies the smallest capacity value along the path P. Lines $5-8$ remove the available capacity from the residual network along the path; if $r(u,v)$ becomes zero, the edge (u,v) is treated as no longer being available. This way, the procedure simply forces flow from s to t, until no more flow can be passed. Finally, when there are no more paths from s to t, the residual network R is returned, at line 10. The network R contains sufficient information to easily find the s-t minimum cut or maximum flow of G. The residual network can also be used to find a connected component that contains s; this fact will be used in the following algorithms.

Algorithm 6.7 shows the pseudocode of the algorithm by Flake *et al.* [FLG00, FLGC02]

aimed at finding the community of a given set of web sites. Its input is a graph G, a set of "seed" web sites S, and a single parameter α, explained below.

Algorithm 6.7: FLAKE-et-al-1

Input: $G = (V, E)$: weighted graph
S: set of nodes
α: real number
Output: the community that contains S

1. $V_\alpha \leftarrow V \cup \{s, t\}$
2. $E_\alpha \leftarrow E \cup \{(v, t) \mid v \in V\} \cup \{(s, u) \mid u \in S\}$
3. **FOR ALL** $v \in V$ **DO**
4. $\quad c(v, t) \leftarrow \alpha$
5. **END FOR**
6. **FOR ALL** $u \in S$ **DO**
7. $\quad c(s, u) \leftarrow \infty$
8. **END FOR**
9. $G_\alpha \leftarrow (V_\alpha, E_\alpha)$
10. $R \leftarrow \text{MAX-FLOW}(G_\alpha, s, t)$
11. $X \leftarrow$ Nodes of the smallest component containing the source s in R
12. **RETURN** $X - \{s\}$

This algorithm creates a new graph G_α with two artificial nodes s and t. The source node s is connected with infinite capacity to all pages in the seed set S. The sink node t is connected to all original nodes with capacity α. After constructing the graph G_α, the procedure calls MAX-FLOW as a subroutine and returns the portion of the resulting residual graph R that remains connected to s. This connected component is guaranteed to be a *defensive alliance*, provided that the algorithm has not terminated with the trivial cut that separates the nodes of S from the rest of the graph. The main theoretical result about this algorithm is connected with the parameter α and is given in Theorem 6.1.

Theorem 6.1. Let X be a community found by Algorithm 6.7. For any pair of node-sets P and Q such that $P \cup Q = X$ and $P \cap Q = \varnothing$ the following bounds hold:

$$\frac{f(X, V - X)}{| V - X |} \leq \alpha \leq \frac{f(P, Q)}{min(| P |, | Q |)}.$$

The proof of Theorem 6.1 may be found in [FTT04]. This theorem shows that the parameter α serves as an upper-bound for the *inter-community* edge capacity, and a lower-bound for the *intra-community* edge capacity. Thus, the algorithm simultaneously guarantees that community nodes are relatively densely linked to one another but relatively sparsely connected to non-community nodes. The bounds given in Theorem 6.1 show how to use α to tune the size and the number of identified communities. A small choice of α, say close to zero, can yield just one community that comprises the entire graph. A large value for α, say $\alpha = 1 + \sum_{(u,v) \in E} c(u, v)$ will yield n singleton communities. The main disadvantage of Algorithm 6.7 is that it will fail to find an existing community that does not obey the bounds given in Theorem 6.1. Algorithm 6.8 and 6.9 use Algorithm 6.7 as a subroutine.

Algorithm 6.8: FLAKE-et-al-2

Input: S : set of nodes
k : integer
Output: an approximate community containing the set S

1. **WHILE** number of iterations is less than desired **DO**
2. $G \leftarrow$ a crawl from S of depth k
3. $\alpha \leftarrow | S |$
4. $X \leftarrow$ FLAKE-et-al-1(G, S, α)
5. rank the nodes of X by the number of neighbors they have inside X
6. add the highest ranked non-seed nodes of X to S
7. **END WHILE**
8. **RETURN** X

Algorithm 6.8 uses a fixed-depth crawl to calculate an approximate community and then uses the "strongest" members in the community to serve as the seeds for the next iteration. This algorithm is appropriate when only a small portion of the graph can be contained in memory.

Algorithm 6.9 aims to find all the communities in a graph. It is only appropriate when the whole graph fits in main memory.

Algorithm 6.9: FLAKE-et-al-3

Input: G : graph
α : real
Output: a clustering of G into communities

1. **array**: $ClusterLabel$
2. $S \leftarrow V$
3. **WHILE** there is a node $s \in S$ **DO**
4. $X \leftarrow$ FLAKE-et-al-1($G, \{s\}, \alpha$)
5. **FOR ALL** $v \in X$ **DO**
6. $ClusterLabel\ (v) \leftarrow s$
7. **END FOR**
8. $S \leftarrow S - X$
9. **END WHILE**
10. **RETURN** $ClusterLabel$

6.4. Other community-mining algorithms

A few additional algorithms that do not fall under any of the preceding categories have also been proposed. For example, Newman [New04] proposed a greedy algorithm that optimizes "*modularity*"—a measure of the quality of a partition into communities. Alternative strategies for optimizing the same measure were proposed in [CNM04]. Greco *et al.* [GGZ04] modeled web communities as bipartite graphs (where *hubs* links to *authorities*) and then analyzed the expected

growth of such communities by using tools from random graph theory. Based on the results of this analytical work, these authors developed an algorithm for mining such communities. Finally, Bagrow and Bollt [BB05] proposed a local, greedy, community-mining algorithm based on *degree*.

7. PROPOSED ALGORITHMS FOR COMMUNITY MINING

In Chapter 4, two versions of community-mining—(i) *partitioning into communities* and (ii) *seed growth*—were defined. In Chapter 6 we saw that many algorithms—employing techniques ranging from hierarchical clustering to spectral partitioning and network flows—have been proposed for the former version. On the other hand, relatively little attention has been devoted to the latter version of the problem.

This chapter proposes some greedy, best-first algorithms for the seed-growth version of the community-mining problem.

7.1. Description of the Algorithms

The community-mining algorithms described in this section are designed with several considerations in mind. First, the main target application for this algorithm is focused crawling [CBD99, DCLG00, MPS04]. In order to be suitable for such an application, the algorithm has to begin with a small set of seed-nodes and then expand by searching their neighborhood. Second, the objective is to discover a group of nodes that are densely linked among them while being sparsely linked with the rest of the network. Under these considerations, the clustering coefficient is a reasonable parameter to guide the search because: (i) a group of nodes having a large clustering coefficient must necessarily have a high density of links; and (ii) there is evidence that regions of several web-like networks that have high clustering coefficient consist of nodes that share some common theme [EM02]. It remains to explore the extent to which it is possible to discover densely-linked groups of nodes via a greedy strategy that favors the nodes

which have a high clustering coefficient with respect to the community nodes. This idea is analyzed in the remainder of the chapter.

Algorithm 7.1 shows the pseudo-code for the FIND-COMMUNITY procedure. This procedure takes as inputs a graph G, a set S of seed-nodes and a threshold value α. Although many web-like networks are directed, here for simplicity we assume that G is an undirected graph. Beginning with $C = S$, the algorithm FIND-COMMUNITY grows the community C by repeatedly searching for "valuable" nodes in the neighborhood of C.

Algorithm 7.1: FIND-COMMUNITY

Input: $G = (V, E)$: an undirected graph
S : the set of "seed" nodes
α : real (threshold)
Output: C : a set of nodes representing a community that contains S

1. $C = S$
2. N_1 = First neighborhood of C
3. N_2 = Second neighborhood of C
4. **REPEAT**
5. I, L = FILTER-NEIGHBORHOOD (G, C, N_1, N_2)
6. $C = C \cup I$
7. N_1 = {First neighborhood of I } $\cap$ N_2
8. N_2 = {First neighborhood of N_1 } $-$ C
9. **UNTIL** N_1 or N_2 becomes very small
10. **RETURN** C

These "valuable" nodes are found by calling the procedure FILTER-NEIGHBORHOOD (Algorithm 7.2). This procedure partitions the nodes of the first neighborhood N_1 of C into two subsets: the set I, which consists of the nodes that will be included in the community (called internal nodes), and the set L, which consists of the non-community (or, leaf) nodes.

Algorithm 7.2: FILTER-NEIGHBORHOOD

Input: $G = (V, E)$: a graph
C : community obtained so far
N_1 : the first neighborhood of C
N_2 : the second neighborhood of C
α : threshold

Output: I : a subset of N_1 ; internal nodes to be included in C
L : complement of I in N_1 ; leaf nodes, not to be expanded

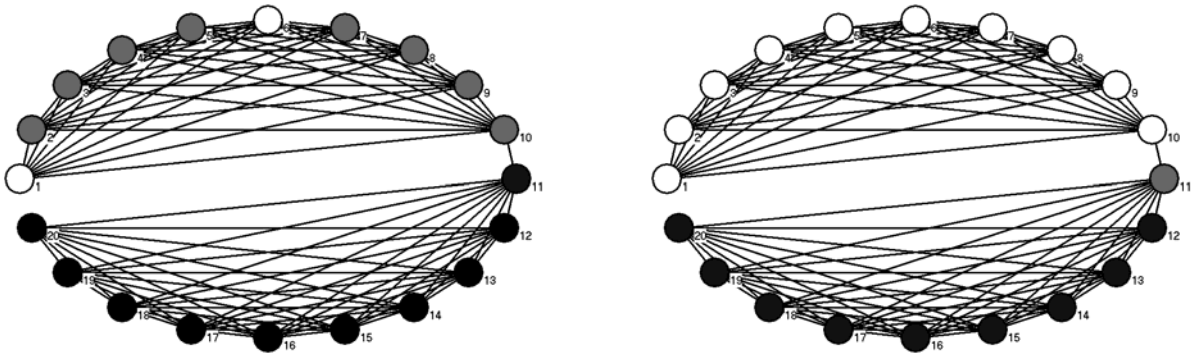
1. $I = \varnothing,\ L = \varnothing$
2. **FOR ALL** nodes v in N_1 **DO**
3. $[C_0(v), C_1(v), C_2(v)]$ = COMPUTE-CC(G, v, C, N_1, N_2) ;
4. **IF** $C_0(v) - C_2(v) \geq \alpha$ **OR** $C_1(v) - C_2(v) \geq \alpha$ **THEN**
5. $I = I \cup \{v\}$
6. **ELSE**
7. $L = L \cup \{v\}$
8. **END FOR**
9. **RETURN** I, L

The most critical and time-consuming computation of the procedure FILTER-NEIGHBORHOOD lies in Line 3, namely, in the call to the procedure COMPUTE-CC. This procedure is called for each node v in the first neighborhood N_1 and returns three scores: (i) $C_0(v)$—the clustering coefficient of node v with respect to the subgraph induced by $C + v$; (ii) $C_1(v)$—the clustering coefficient of node v with respect to the subgraph induced by $N_1 + v$; and (iii) $C_2(v)$—the clustering coefficient of node v with respect to the subgraph induced by $N_2 + v$.

Here, it is assumed that the clustering coefficient of a node v is computed by a brute-force method, *i.e.,* by counting the number of edges between the neighbors of v (other options, such as randomized approximations are considered later in this chapter).

The decision whether v will be an internal or a leaf node is made in Line 4 of procedure FILTER-NEIGHBORHOOD. The criterion is simple: if node v is more tightly clustered to the nodes of C or N_1 than it is to the nodes of N_2 by a value greater than or equal to the threshold α, then v is added to the set I, otherwise it is added to the set L. In all the experimental results presented later, the parameter α was fixed at 0.05—a value which was empirically found to yield good results. To keep the pseudo-code simple we have omitted the treatment of nodes with degree less than two (for which the clustering coefficient is undefined). These nodes are handled following the same reasoning as in Line 4 of FILTER-NEIGHBORHOOD. For example, if a node w of C has a single neighbor in N_1 and no neighbor in N_2 then it is added to I; the remaining cases are handled similarly. FILTER-NEIGHBORHOOD algorithm runs until the first or second neighborhoods become very small. The stopping criterion used in our experiments was $|N_1| \leq 1$ or $|N_2| \leq 2$.

As a first illustration, Figure 7.1 shows how the FIND-COMMUNITY algorithm performs in a trivial case: a graph consisting of two cliques K_{10} joined by an edge. Two nodes (nodes 1 and 6, shown in white in Figure 7.1(a)) were selected as seeds uniformly at random from the first K_{10}. The first neighborhood of the seed nodes (nodes shown in grey in Figure 7.1(a)) is $N_1 = \{2, 3, 4, 5, 7, 8, 9, 10\}$ while the second neighborhood N_2 consists of all the nodes of the second K_{10} (shown in black in Figure 7.1(a)). It one step (*i.e.*, one execution of Lines 3-7) of FIND-COMMUNITY, all the nodes of the first K_{10} are classified as community nodes (shown in white in Figure 7.1(b)).



(a) (b)

Figure 7.1. A graph consisting of two complete graphs K_{10} plus an edge that links them together: (a) initial configuration (b) after one step of the algorithm FIND-COMMUNITY.

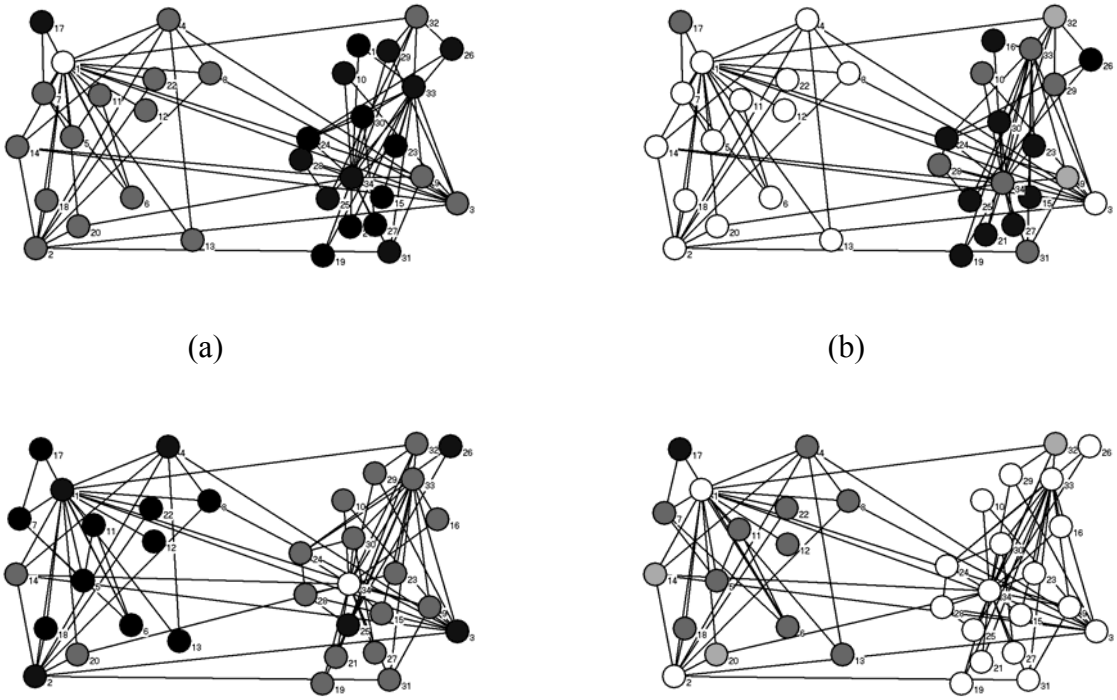
7.2. Experimental Results

We implemented the FIND-COMMUNITY algorithm and tested its performance on several real and computer-generated networks. The implementation was done in C++ using the software library LEDA[2] and the testing was carried out on a Linux box. The results of our experiments are shown next.

a) Zachary's karate club network

The first network we used to test the FIND-COMMUNITY algorithm, is the Zachary's karate club network [Zac77]—a real-life network with 34 nodes and two communities, which has become a frequently-used benchmark for community-mining algorithms. The nodes of this

network represent the members of a karate club at an American university, while the edges represent their social interactions. Zachary's network is known to consist of two different communities of nodes, each corresponding to the club members that where sided with one of the two club leaders during a dispute (see *e.g.,* [GN03] for a more detailed description of this network). Figure 7.2 shows the performance of our algorithm in discovering each of the two communities of this network, beginning in each case from a single seed-node (the highest-degree node).



(a) (b)

(c) (d)

Figure 7.2. Discovering the two communities of Zachary's karate club network: (a) Node 1 (white) is chosen as the seed for the first community; (b) The white nodes represent the

[2] Available from http://www.algorithmic-solutions.com/enleda.htm

community found by the algorithm; (c) Node 34 (white) is chosen as the seed for the second community; (d) The white nodes represent the community found by the algorithm.

In the case of discovering the first community with node 1 as seed, all except two nodes were classified correctly (Figure 7.2(a, b)). In the case of discovering the second community with node 34 as seed all but three nodes were classified correctly (Figure 7.2(c, d)).

b) Random graphs with known community structure

Next, we tested the algorithm extensively on a family of random graphs with known community structure. This family of random graphs has also been used frequently to test community-mining algorithms, *e.g.,* [NG03, BB05]. A random graph from this family is characterized by the following parameters: (1) n—the number of nodes; (2) c—the number of communities; (3) d_{in}—the expected number of neighbors of a node within its community; (3) d_{out}—the expected number of neighbors of a node outside its community. The following algorithm generates an instance of such a random graph $G_{n,c,d_{in},d_{out}}$:

1. Let $CommunitySize = n/c$
2. Let $p_{in} = \dfrac{d_{in}}{CommunitySize}$ and $p_{out} = \dfrac{d_{out}}{n - CommunitySize}$
3. Partition the set of nodes into c communities of equal size (it is assumed that n is evenly divided by c)
4. For each pair of nodes (u, v), independently, do the following:
   a) if u and v belong to the same community, join them with an edge with probability p_{in}

b) otherwise, join them with an edge with probability p_{out}

While testing the FIND-COMMUNITY algorithm, we focused on three main questions: (1) How does the accuracy of the algorithm—measured as the fraction of nodes classified correctly—change while the ratio d_{in}/d_{out} increases? (2) What is the impact of the size of the set of seed nodes on the accuracy of the algorithm? (3) How robust is the algorithm to different sets of seed nodes?

To investigate these questions we carried out a number of experiments. The following scenario was common to all them: While keeping the rest of the parameters fixed, the parameter of interest was changed in small increments and for each case a random graph with known community structure was generated; A fraction of the nodes in the *first* community of this random graph was randomly chosen as the set of seed-nodes; The FIND-COMMUNITY algorithm was run with the generated random graph and seed-nodes as input; The fraction of nodes classified correctly by the algorithm was computed.

The performance of the FIND-COMMUNITY algorithm with respect to the questions above is described next.

Accuracy versus d_{in}/d_{out}

Figure 7.3, shows how the accuracy of the algorithm changes with the ratio d_{in}/d_{out} for two random graphs with 16,384 nodes and with 4 and 32 communities, respectively. In this experiment, d_{in} is kept fixed at 32 while d_{out} is changed from 4 to 28 in increments of 4 (*i.e.*, the

ratio d_{in}/d_{out} changes from 0.125 to 0.875). In both cases, the set of *seed* nodes, consisted of 5% of the nodes of the first community, selected uniformly at random. As seen in Figure 7.3, the fraction of correctly classified nodes changed from nearly 0.9 (when $d_{in} = 4$) to nearly 0.7 (when $d_{in} = 28$).

Accuracy versus the size of the set of seed-nodes

Figure 7.4, shows the impact of the relative size of the set of seed nodes on the accuracy of the algorithm. In this case, two random graphs with 16,438 nodes and with 8 and 32 communities, respectively, were generated. In both cases $d_{in} = 32$ while $d_{out} = 8$.

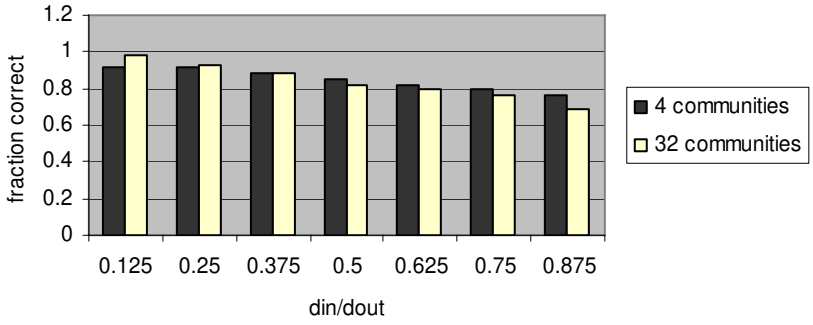


Figure 7.3. Fraction of correctly classified nodes versus the ratio d_{in}/d_{out}.

The size of the set of seed nodes was changed from 2% of community size to 20% of the community size. In the case of the graph with 32 communities, the fraction of correctly classified nodes changed from nearly 90% to almost 99%. In the other case, when only 8 communities were present, this fraction was smaller than in the first case, but always greater than 83%.

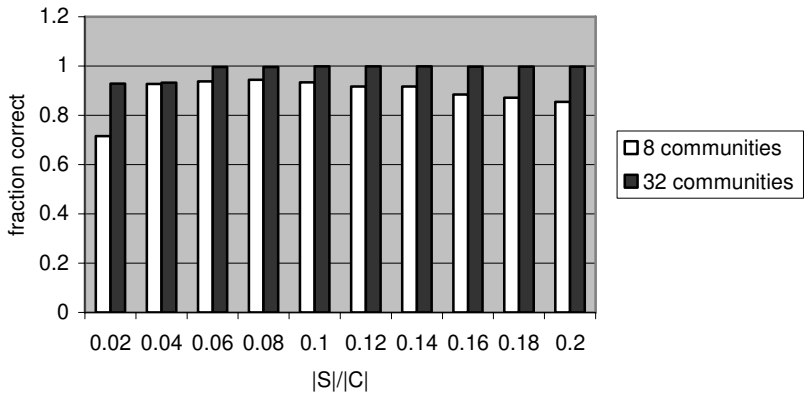


Figure 7.4. Fraction of correctly classified nodes versus the relative size of the set of seed nodes.

Accuracy versus the set of seed-nodes

Figure 7.5, illustrates the robustness of the algorithm to different sets of seeds. In this experiment, we generated a random graph with 16,384 nodes, with 8 communities, and with d_{in} = 32, d_{out} = 8. The FIND-COMMUNITY algorithm was executed ten times with this graph as input. For each run, a new set S was generated by selecting 5% of the nodes of the *first* community uniformly at random. As seen in Figure 7.5, the fraction of nodes classified correctly changed very little—it was always between 95% and 96.5%.

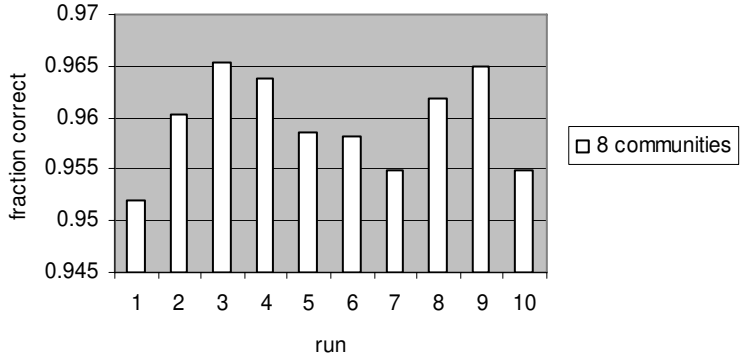


Figure 7.5. Robustness of the algorithm to different sets of seed nodes.

In summary, the fraction of nodes classified correctly by the FIND-COMMUNITY algorithm was generally above 80% and often above 90% when tested on random graphs with known community structure. Furthermore, the algorithm achieves good accuracy with only a small fraction (about 1%) of community nodes as seeds and the performance varies little with the set of seed nodes.

However, several issues need to be addressed before this algorithm can be usefully applied in practice. First, more extensive testing needs to be done especially with data from real networks such as Web crawls. Second, a rigorous method needs to be developed to evaluate the quality of the communities produced by this algorithm. Third, in order to mine large communities (in the order of tens of thousands of nodes), the speed of the algorithm needs to be improved (without compromising its accuracy). One could consider randomizing the COMPUTE-CC procedure, perhaps in combination with using a *generalized clustering coefficient*, where not only the first neighborhood is taken into account but also the second, the third, *etc.* Another option would be to parallelize the algorithm. Investigating these questions as well as applying this algorithm to mine the communities of some web-like networks, remain as topics of our current and future research.

Time complexity

It is difficult to obtain an exact expression for the time complexity of FIND-COMMUNITY in terms of the *order* and *size* of the graph *G*. However, assuming that the number of nodes visited by this algorithm (internal plus leaf nodes) is n and that the maximum

degree encountered is $d_{\max}$, it is easy to see that the time complexity is of order $O(nd_{\max}^2)$, because the procedure COMPUTE-CC takes $O(d_{\max}^2)$ time for each node.

Thus, FIND-COMMUNITY algorithm would have practical value only if $d_{\max}$ is small, or if the degree of most encountered nodes is small. It is reasonable to expect that this algorithm would run in near-linear time in the number of visited nodes, if the input is a web-like network which is known to follow a power-law degree distribution.

8. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this dissertation we investigated the community structure in web-like networks. Motivated by simple processes existent in web-like networks, we proposed two birth-death dynamic random graph models of such networks. Both models were found to posses a power-law degree distribution, in agreement with many real web-like networks. Our modeling studies suggest that *preferential deletion* of nodes is likely to be a key mechanism in the evolution of web-like networks.

Due to a wide array of potential applications, community mining in web-like networks has attracted the attention of researchers from many fields. Several graph theoretic definitions, generally motivated by empirical observations, have appeared in literature. However, a formal evaluation of the appropriateness of various definitions of community has been lacking. To address this issue, we developed a framework for evaluating the suitability of a particular definition of community. This framework consists in estimating through sampling techniques the concentration in web-like networks of a subgraph proposed as definition of community and then deducing the statistical significance (z-score) of the concentration by contrasting with appropriately defined random graphs. We applied this methodology to evaluate two graph concepts—alliance and near-clique. Essentially, an alliance is a group of nodes with high minimum alliance coefficient (ratio of the number of neighbors of a node inside the group to the number of neighbors outside), while near-clique is a group of nodes with high minimum clustering coefficient (fraction of pairs of neighbors of a node that are neighbors themselves). We found that the concentration was generally higher for near-cliques than for alliances. Furthermore, the occurrence of near-cliques was statistically more significant (as indicated by higher z-scores) than the occurrence of

alliances. These results suggest that near-clique is a better characterization of community than alliance. More importantly, the proposed framework may be applied to discover additional graph parameters that are essential in characterizing community.

Assuming the definition of community as alliance or near-clique, we analyzed the computational complexity of various community-mining problems. The results we derived together with other results that have recently appeared in literature show that several community-mining problems are hard to solve in general graphs. In particular, the problem of partitioning a given graph into subsets of nodes, each forming a community, is NP-hard. Due to these hardness results, we concentrated on the easier problem of finding the maximal community that contains a given set of seed nodes. This version of community mining is suitable for applications such as focused crawling—the selective search for web pages on a given topic. We devised several fast, greedy community-mining algorithms based on clustering coefficient and generalized versions of this parameter. The performance of the proposed algorithms was evaluated experimentally in several benchmark networks and it was found that they are very effective in mining alliances and near-cliques.

The future research agenda spans all three lines of investigation discussed above. First, despite the significant progress toward designing an accurate model of web-like networks, we are still far from having a comprehensive understanding of the basic processes responsible for the evolution of complex networks. We intend to deepen the research into the role of preferential deletion of nodes in the evolution of networks. Of particular interest is to understand the impact of this process on degree-correlation and clustering coefficient. Further, it would be desirable to tune the parameters that control the relative rates of birth and death, for instance by deriving the critical probability for the emergence of the giant

component in a birth-death model with a preferential deletion of nodes. These problems may be attacked using essentially the same tools of random graph theory that where employed in the dissertation.

Second, the proposed framework for evaluating community definitions will be applied to discover other parameters that are essential in characterizing community. Additional insight into these parameters may be achieved by investigating the evolution of communities experimentally (looking at network data over time) and analytically (in dynamic random graph models). Some natural parameters that we intend to investigate are average distance, degree correlation and the entropy of degree distribution. Complementary to the problem of finding graph theoretic characterizations of community is that of devising techniques to tie the statistical significance of concentration (high z-scores) with function (topic, theme) on specific networks, such as the Web. These techniques would necessarily be ad-hoc and dictated by the nature of the network. For instance, in the Web one can use existing text-based techniques to evaluate the degree to which groups that occur in statistically significant concentration are topically related.

Finally, many algorithmic questions remain open as well. We will investigate the performance of various fast greedy algorithms that attempt to find optimal communities satisfying the parameter constrains identified using the techniques discussed in the previous paragraph. A related problem to which we intend to devote efforts and which has immense theoretical and practical interest is to determine whether it is easier to solve hard community-mining instances in the dynamic random graph models.

APPENDIX

WEB RESOURCES AND SOFTWARE TOOLS

The following list contains some pointers to recently offered courses on data mining and complex networks which have overlapping content with this dissertation.

1. The Structure of Information Networks (Cornell): Kleinberg.
   http://www.cs.cornell.edu/Courses/cs685/2002fa/

2. Algorithmic Aspects of Computer Networks (Boston University): Byers.
   http://www.cs.bu.edu/fac/byers/courses/591/S02/cs591.html

3. Internet Algorithmics (Brown): Goodrich.
   http://www.cs.brown.edu/courses/cs195-3/

4. Algorithms for Indexing and Search (Carnegie Mellon): Blelloch, Lafferty, Miller.
   http://www-2.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15850-s99/www/readings

5. Networks and Complexity in Social Systems (Columbia): Watts.
   http://www.columbia.edu/itc/sociology/watts/w3233/

6. Scaling in Networks (Columbia): Lazar.
   http://comet.columbia.edu/courses/elen_e9701/2001/overview.html

7. Algorithms at the End of the Wire (Harvard): Mitzenmacher.
   http://www.eecs.harvard.edu/~michaelm/CS222/class.html

8. Hypertext retrieval and mining (IIT Bombay): Chakrabarti.
   http://www.cse.iitb.ac.in/~soumen/teach/cs610s2001/

9. Complex Human Networks Reading Group (MIT): Pentland, Clarkson, Choudhury.
   http://web.media.mit.edu/~tanzeem/cohn/CoHN.htm

10. Advanced Algorithms in Data Mining (Penn State): Zha.
    http://www.cse.psu.edu/~zha/CSE597/administria.html

11. Web Protocols, Principles, and Applications (Polytechnic): Suel.
    http://cis.poly.edu/cs912/

12. Information Retrieval, Discovery, and Delivery (Princeton): LaPaugh.
    http://www.cs.princeton.edu/courses/archive/spring02/cs435/

13. Data Mining (Stanford): Ullman.
    http://www-db.stanford.edu/~ullman/mining/mining.html

14. Information Retrieval and Distributed Databases (Stanford): Raghavan.
    http://www-db.stanford.edu/cs347.2001.spring/

15. Seminar in Data Mining and Search (Tel Aviv): Fiat.
    http://www.cs.tau.ac.il/~fiat/datamine/dm.htm

16. Recommender Systems (Virginia Tech): Ramakrishnan.
    http://people.cs.vt.edu/~ramakris/Courses/CS6604-RS/

17. Advanced Topics in Data Mining (UC Irvine): Smyth.
    http://www.ics.uci.edu/~smyth/courses/ics280/

18. Networks and Complexity (UC Irvine): White.
    http://eclectic.ss.uci.edu/~drwhite/Anthro179a/SocialDynamics02.html

19. Advanced algorithms in data mining (U. Helsinki): Mannila.
    http://www.cs.helsinki.fi/u/mannila/aadm

20. Graph Mining and Link Analysis Reading Group (U. Maryland): Getoor, Lu.
    http://www.cs.umd.edu/~qinglu/summer02-reading.htm

21. Scaling, Power Laws, and Small World Phenomena in Networks (U. Mass.): Towsley.
    http://www-net.cs.umass.edu/cs691s/

22. Peer-to-Peer and Application-Level Networking (U. Mass.): Kurose, Levine, Towsley.
    http://www-net.cs.umass.edu/cs791n/

23. Practicum in Data Mining (U. Texas): Ghosh.
    http://www.lans.ece.utexas.edu/course/ee380l/2002sp/index_prac.shtml

24. Machine Learning for Text Analysis (U. Wisconsin): Craven.
    http://www.cs.wisc.edu/~craven/cs838-f00.html

The following web sites provide data sets for various web-like networks:

1. http://webgraph-data.dsi.unimi.it/

This web site provides several data sets obtained by crawling the Web. These data sets are very large; for instance the most recent data set provided is obtained by a 2004 crawl of the *.it* domain performed by *UbiCrawler*. The graph contains 41.3 Mpages and 1.15 Glinks. The data sets are stored in compressed format; several tools in Java are provided to handle the data.

2. http://vlado.fmf.uni-lj.si/pub/networks/data/

This web site provides numerous data sets for real networks. The data for some of the networks used in our experiments was obtained from this web site.

Some Software Tools

LEDA

LEDA is a C/C++ library that implements various advanced data structures, including graphs. This library is commercial software distributed by Algorithmic Solution Software GmbH and is available at http://algorithmic-solutions.com/. LEDA has been used heavily to implement most algorithms discussed in this dissertation.

PAJEK

PAJEK is a tool for visualizing networks. We found this tool useful in our experiments with random graph models and during the testing of community-mining algorithms discussed in the dissertation. The software may be freely downloaded at http://vlado.fmf.uni-lj.si/pub/networks/pajek/

UCFBOT

UcfBot is a high-performance general-purpose Web crawler developed in the Center for Parallel Computation of the Computer Science Department at UCF. The details of the architecture, implementation and capabilities of this crawler are discusses in [BCD00].

MERSENNE TWISTER Pseudo-Random Number Generator

Mersenne Twister (MT) is a recent pseudo-random number generator developed by Matsumoto and Nishimura [MN98]. This generator has been shown to generate sequences of high quality and has been found to be up to four times faster than the standard *rand*() function C/C++. The version MT19937 of this generator—which has been used in all our experiments—has a period of $2^{19937} - 1$. Implementations of MT in various languages may be found freely at http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html.

LIST OF REFERENCES

[AAS03] G. Adami, P. Avesani and D. Sona. Clustering documents in a web directory. In *Proceedings of the 5th ACM International Workshop on Web Information and Data Management (WIDM '03)*, pp. 66-73, 2003.

[AH01] L. A. Adamic and B. A. Huberman. The Web's hidden order. *Commun. ACM*, vol. 44(9), pp. 55-60, 2001.

[AM01] M. Adler and M. Mitzenmacher. Towards compressing Web graphs. In *Proceedings of the Data Compression Conference (DCC 2001)*, pp. 203-212, 2001.

[AMO93] R. K. Ahuja, T. L. Magnanti and J. B. Orlin. *Network flows: Theory, algorithms and applications*, Englewood Cliffs, NJ: Prentice Hall, 1993.

[ACL01] W. Aiello, F. Chung and L. Lu. A random graph model for power law graphs. *Experiment. Math.*, vol. 10(1), pp. 53-66, 2001.

[ACL02] W. Aiello, F. Chung and L. Lu. Random evolution in massive graphs. In *Handbook of Massive Data Sets*, vol. 4, *Massive Comput.*, J. Abello, P. M. Pardalos, and M. G. Resende, (Eds.). Dordrecht: Kluwer Acad. Publ., pp. 97-122, 2002.

[AJB99] R. Albert, H. Jeong and A.-L. Barabási. Diameter of the World Wide Web. *Nature*, vol. 401, pp. 130-131, 1999.

[AB00] R. Albert and A. L. Barabási. Topology of evolving networks: local events and universality. *Physical Review Letters*, vol. 85(24), pp. 5234-5237, 2000.

[AHB00] R. Albert, J. Hawoong and A. L. Barabási. Error and attack tolerance of complex networks. *Nature*, vol. 406(6794), pp. 378-382, 2000.

[AB02] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, vol. 74(1), pp. 47-97, 2002.

[Ald04] D. J. Aldous. A tractable complex network model based on the stochastic mean-field model of distance. *Complex networks (Lecture Notes in Phys. Vol.650)*, pp. 51-87, 2004.

[AKS02] E. Almaas, R. V. Kulkarni and D. Stroud. Characterizing the structure of small-world networks. *Physical Review Letters*, vol. 88(9), art. no. 098101, 2002.

[AS92] N. Alon and J. Spencer. *The probabilistic method*. New York, NY: Wiley, 1992.

[AKS98] N. Alon, M. Krivelevich and B. Sudakov. Finding a large hidden clique in a random graph. *Random Structures & Algorithms*, vol. 13(3-4), pp. 457-466, 1998.

[ACDL03] E. Amitay, D. Carmel, A. Darlow, R. Lempel and A. Soffer. The connectivity sonar: detecting site functionality by structural patterns. In *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia (HYPERTEXT '03)*, pp. 38-47, 2003.

[AHS96] P. Arabie, L. J. Hubert and G. de Soete. *Clustering and classification*. Singapore; NJ: World Scientific, 1996.

[ADDG04] A. Arenas, L. Danon, A. Diaz-Guilera, P. M. Gleiser and R. Guimera. Community analysis in social networks. *European Physical Journal B*, vol. 38(2), pp. 373-380, 2004.

[BB05] J. Bagrow and E. Bollt. A Local Method for Detecting Communities. Online Article, 2005.

[BCD05] H. Balakrishnan, A. Cami and N. Deo. UcfBot - A High-Performance Web Crawler. University of Central Florida, School of Computer Science, Technical Report CS-TR-05-08, 2005.

[BFS03] P. Baldi, P. Frasconi and P. Smyth. *Modeling the Internet and the Web: Probabilistic Methods and Algorithms*: John Wiley & Sons, 2003.

[BBKT04] Z. Bar-Yossef, A. Z. Broder, R. Kumar and A. Tomkins. Sic transit gloria telae: towards an understanding of the web's decay. In *Proceedings of the 13th International Conference on World Wide Web (WWW'04)*, pp. 328-337, 2004.

[BA99a] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, vol. 286(5439), pp. 509-512, 1999.

[BAJ99] A.-L. Barabási, R. Albert and H. Jeong. Mean-field theory for scale-free random networks. *Physica A: Statistical Mechanics and its Applications*, vol. 272(1-2), pp. 173-187, 1999.

[BAJ00] A.-L. Barabási, 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*, vol. 281(1-4), pp. 69-77, 2000.

[BRV01] A.-L. Barabási, E. Ravasz and T. Vicsek. Deterministic scale-free networks. *Physica A: Statistical Mechanics and its Applications*, vol. 299(3-4), pp. 559-564, 2001.

[Bar02] A.-L. Barabási. *Linked: the New Science of Networks*. Cambridge, MA: Perseus Pub., 2002.

[BMBB04] A. L. Barabási, M. A. de Menezes, S. Balensiefer and J. Brockman. Hot spots and universality in network dynamics. *European Physical Journal B*, vol. 38(2), pp. 169-175, 2004.

[BW00] A. Barrat and M. Weigt. On the properties of small-world network models. *European Physical Journal B*, vol. 13(3), pp. 547-560, 2000.

[BBPV04a] A. Barrat, M. Barthélemy, R. Pastor-Satorras and A. Vespignani. The architecture of complex weighted networks. *Proceedings Of The National Academy Of Sciences Of The United States Of America*, vol. 101(11), pp. 3747-3752, 2004.

[BA99b] M. Barthélemy and L. A. N. Amaral. Small-World networks: evidence for a crossover picture. *Physical Review Letters*, vol. 82(15), pp. 3180-3183, 1999.

[BBPV04b] M. Barthélemy, A. Barrat, R. Pastor-Satorras and A. Vespignani. Velocity and hierarchical spread of epidemic outbreaks in scale-free networks. *Physical Review Letters*, vol. 17, art. no. 178701, 2004.

[BBPV05] M. Barthélemy, A. Barrat, R. Pastor-Satorras and A. Vespignani. Characterization and modeling of weighted networks. *Physica A: Statistical Mechanics and its Applications*, vol. 346(1-2), pp. 34-43, 2005.

[BC78] E. A. Bender and E. R. Canfield. The asymptotic number of labeled graphs with given degree sequences. *J. Combinatorial Theory Ser. A*, vol. 24(3), pp. 296-307, 1978.

[BBBC03] N. Berger, B. Bollobás, C. Borgs, J. Chayes and O. Riordan. Degree distribution of the FKP network model. In *Automata, Languages and Programming*, vol. 2719, *Lecture Notes in Computer Science*, J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, (Eds.). Berlin: Springer-Verlag, pp. 725-738, 2003.

[BH98] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*, pp. 104-111, 1998.

[BCHR01] K. Bharat, B.-W. Chang, M. R. Henzinger and M. Ruhl. Who Links to Whom: Mining Linkage between Web Sites. In *2001 IEEE International Conference on Data Mining (ICDM '01)*: IEEE Computer Society, pp. 51-58, 2001.

[BGS03] M. Bianchini, M. Gori and F. Scarselli. PageRank and Web communities. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI 2003)*, pp. 365-371, 2003.

[BB01] G. Bianconi and A. L. Barabási. Competition and multiscaling in evolving networks. *Europhysics Letters*, vol. 54(4), pp. 436-442, 2001.

[BC03] G. Bianconi and A. Capocci. Number of loops of size h in growing scale-free networks. *Physical Review Letters*, vol. 7, art. no. 078701, 2003.

[Bia04] G. Bianconi. Number of cycles in off-equilibrium scale-free networks and in the Internet at the autonomous system level. *European Physical Journal B*, vol. 38(2), pp. 223-230, 2004.

[BMV05] G. Bianconi, M. Marsili and F. Vega-Redondo. On the non-trivial dynamics of complex networks. *Physica A: Statistical Mechanics and its Applications*, vol. 346(1-2), pp. 116-122, 2005.

[BF01] T. Bohman and A. Frieze. Avoiding a giant component. *Random Structures Algorithms*, vol. 19(1), pp. 75-85, 2001.

[BPSR05] T. Bohman, O. Pikhurkho, B. Sudakov, A. Rucinski, N. Wormald, M. Molloy, D. Achlioptas and A. Frieze. Eight lectures on Random Graphs: Talks given at a joint MAA-AMS meeting in Atlanta, Georgia, January 3-4, 2005. Online Article, 2005.

[Bol79] B. Bollobás. *Graph theory: an introductory course*. New York: Springer Verlag, 1979.

[Bol85] B. Bollobás. *Random graphs*. London: Academic Press, 1985.

[Bol90] B. Bollobás. The diameter of random graphs. *IEEE Trans. Inform. Theory*, vol. 36(2), pp. 285-288, 1990.

[BRST01] B. Bollobás, O. Riordan, J. Spencer and G. Tusn□?dy. The degree sequence of a scale-free random graph process. *Random Structures Algorithms*, vol. 18(3), pp. 279-290, 2001.

[BBCR03] B. Bollobás, C. Borgs, J. Chayes and O. Riordan. Directed scale-free graphs. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 132-139, 2003.

[BR03a] B. Bollobás and O. Riordan. Robustness and vulnerability of scale-free random graphs. *Internet Math.*, vol. 1(1), pp. 1-35, 2003.

[BR03b] B. Bollobás and O. M. Riordan. Mathematical results on scale-free random graphs. In *Handbook of Graphs and Networks: From the Genome to the Internet*, S. Bornholdt and H. G. Schuster, (Eds.). Weinheim: Wiley-VCH, pp. 1-34, 2003.

[BR04] B. Bollobás and O. Riordan. The diameter of a scale-free random graph. *Combinatorica*, vol. 24(1), pp. 5-34, 2004.

[BR04] B. Bollobás and O. Riordan. Coupling scale-free and classical random graphs. *Internet Math.*, vol. 1(2), pp. 215-225, 2004.

[BS91] R. A. Botafogo and B. Shneiderman. Identifying aggregates in hypertext structures.

In *Proceedings of the 3rd annual ACM conference on Hypertext (HYPERTEXT '91)*, pp. 63-74, 1991.

[BDH04] R. C. Brigham, R. D. Dutton and S. T. Hedetniemi. A sharp lower bound on the powerfull alliance number of Cm \sq Cn. *Congressum Numerantium*, vol. 167, pp. 57-63, 2004.

[BKMR00] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins and J. Wiener. Graph structure in the Web. *Computer Networks*, vol. 33(1-6), pp. 309-320, 2000.

[Buc02] M. Buchanan. *Nexus: small worlds and the groundbreaking science of networks*. New York: W.W. Norton, 2002.

[BO04] P. G. Buckley and D. Osthus. Popularity based random graph models leading to a scale-free degree sequence. *Discrete Mathematics*, vol. 282(1-3), pp. 53-68, 2004.

[Bur74] J. D. Burtin. Extremal metric characteristics of a random graph I. *Theo. Verojatnost. i Primenen.*, vol. 19, pp. 740-754, 1974.

[BFT01] B. Bush, C. Files and D. Thompson. Empirical Characterization of Infrastructure Networks. Los Alamos National Laboratory, Technical Report LA-UR-01-5784, 2001.

[CRZM03] P. Calado, B. Ribeiro-Neto, N. Ziviani, E. Moura and I. Silva. Local versus global link information in the Web. *ACM Trans. Inf. Syst.*, vol. 21(1), pp. 42-63, 2003.

[CPV04] G. Caldarelli, R. Pastor-Satorras and A. Vespignani. Cycles Structure and Local Ordering in Complex Networks. *European Physical Journal B*, vol. 38, pp. 183-186, 2004.

[CNSW00] D. S. Callaway, M. E. J. Newman, S. H. Strogatz and D. J. Watts. Network robustness and fragility: percolation on random graphs. *Physical Review Letters*, vol. 25, art. no. 5468, 2000.

[CHKN01] D. S. Callaway, J. E. Hopcroft, J. M. Kleinberg, M. E. Newman and S. H. Strogatz. Are randomly grown graphs really random? *Physical Review. E (Statistical, Nonlinear, And Soft Matter Physics)*, vol. 4, part 1, art. no. 041902, 2001.

[CBDD05] A. Cami, C. Lisetti and M. Sierhuis. Towards the Simulation of a Multi-Level Model of Human Emotions. In *Proceedings of the AAAI 2004 Spring Symposium, TR SS-04-02*, pp. 5-9, 2004.

[CBDD05] A. Cami, H. Balakrishnan, N. Deo and R. D. Dutton. On the complexity of some global alliance problems. *Journal of Combinatorial Mathematics and Combinatorial Computing (submitted)*, 2005.

[CSCC05] A. Capocci, V. D. P. Servedio, G. Caldarelli and F. Colaiori. Detecting communities in large networks. *Physica A: Statistical and Theoretical Physics*, vol. 352(2-4), pp. 669-676, 2005.

[CCLP04] C. Castellano, F. Cecconi, V. Loreto, D. Parisi and F. Radicchi. Self-contained algorithms to detect communities in networks. *European Physical Journal B*, vol. 38(2), pp. 311-319, 2004.

[CDAR98] S. Chakrabarti, B. Dom, R. Agrawal and P. Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *The VLDB Journal*, vol. 7(3), pp. 163-178, 1998.

[CDGK99] S. Chakrabarti, B. E. Dom, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins. Topic distillation and spectral filtering. *Artificial Intelligence Review*, vol. 13(5-6), pp. 409-435, 1999.

[CBD99] S. Chakrabarti, M. van den Berg and B. Dom. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks*, vol. 31(11-16), pp. 1623-1640, 1999.

[CJPP02] S. Chakrabarti, M. M. Joshi, K. Punera and D. M. Pennock. The structure of broad topics on the web. In *Proceedings of the 11th International Conference on World Wide Web (WWW'02)*, pp. 251-262, 2002.

[CL01] F. Chung and L. Lu. The diameter of sparse random graphs. *Adv. in Appl. Math.*, vol. 26(4), pp. 257-279, 2001.

[CL02] F. Chung and L. Lu. Connected components in random graphs with given expected degree sequences. *Ann. Comb.*, vol. 6(2), pp. 125-145, 2002.

[CL03] F. Chung and L. Lu. The average distance in a random graph with given expected degrees. *Internet Math.*, vol. 1(1), pp. 91-113, 2003.

[CLV03] F. Chung, L. Lu and V. Vu. Spectra of random graphs with given expected degrees. *Proc. Natl. Acad. Sci. USA*, vol. 100(11), pp. 6313-6318, 2003.

[CL04] G. F. Chung and L. Lu. Coupling online and offline analyses for random power law graphs. *Internet Math.*, vol. 1(4), pp. 409-461, 2004.

[CNM04] A. Clauset, M. E. J. Newman and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, vol. 70, art. no. 066111, 2004.

[CEBH00] R. Cohen, K. Erez, D. Ben-Avraham and S. Havlin. Resilience of the Internet to random breakdowns. *Physical Review Letters*, vol. 85(21), pp. 4626-4628, 2000.

[CEBH01] R. Cohen, K. Erez, D. Ben-Avraham and S. Havlin. Breakdown of the Internet under intentional attack. *Physical Review Letters*, vol. 86(16), pp. 3682-3685, 2001.

[Coo02] C. Cooper. Classifying special interest groups in web graphs. *Randomization and approximation techniques in computer science*, vol. 2483, pp. 263-275, 2002.

[CF03] C. Cooper and A. Frieze. A general model of web graphs. *Random Structures Algorithms*, vol. 22(3), pp. 311-335, 2003.

[CF04] C. Cooper and A. Frieze. The size of the largest strongly connected component of a random digraph with a given degree sequence. *Combin. Probab. Comput.*, vol. 13(3), pp. 319-337, 2004.

[CFV04] C. Cooper, A. Frieze and J. Vera. Random deletions in a scale free random graph process. *Internet Math.*, vol. 1(4), pp. 463-483, 2004.

[DH99] J. Dean and M. R. Henzinger. Finding related pages in the World Wide Web. In *Proceedings of the 8th International Conference on World Wide Web (WWW '99)*, pp. 1467-1479, 1999.

[DVGB03] L. Denoyer, J.-N. Vittaut, P. Gallinari, S. Brunessaux and S. Brunessaux. Structured multimedia document classification. In *Proceedings of the 2003 ACM Symposium on Document Engineering (DocEng '03)*, pp. 153-160, 2003.

[Deo74] N. Deo. *Graph theory with applications to engineering and computer science*. Englewood Cliffs, N.J.: Prentice-Hall Inc., 1974.

[DL98] N. Deo and B. Litow. A structural approach to graph compression. In *Proceedings of the MFCS Workshop on Communication*, pp. 91-101, 1998.

[DLP00] N. Deo, B. Litow and P. Gupta. Modeling the Web: Linking Discrete and Continuous. In *Proceedings of the Summer Computer Simulation Conference 2000*, Vancouver, Canada, pp. 395-400, 2000.

[DG01a] N. Deo and P. Gupta. Graph-theoretic Web algorithms: An overview. *Lecture Notes in Computer Science*, vol. 2026, pp. 91-102, 2001.

[DG01b] N. Deo and P. Gupta. Sampling the Web Graph with Random Walks. *Congressus Numerantium*, vol. 149, pp. 143-154, 2001.

[DZN02] N. Deo, Y. Zhang and Z. Nikoloski. Estimating the Independence Number of a Large Random Graph. *Congressus Numerantium*, vol. 159, pp. 95-111, 2002.

[DG03] N. Deo and P. Gupta. Graph-theoretic analysis of the world wide web: new directions and challenges. *Mat. Contemp.*, vol. 25, pp. 49-69, 2003.

[DN03] N. Deo and Z. Nikoloski. The game of cops and robbers on graphs: a model for quarantining cyber attacks. *Congressus Numerantium*, vol. 162, pp. 193-215, 2003.

[DC05a] N. Deo and A. Cami. A birth-death dynamic model of web-like networks. *In Proceedings of the 43rd Annual ACM Southeast Conference*, vol. 2, art. no. 2-26, 2005.

[DC05b] N. Deo and A. Cami. A greedy community-mining algorithm based on clustering coefficient. In *Proceedings of the 36th Southeastern International Conference on Combinatorics, Graph Theory, and Computing (Congressus Numerantium, vol. 168)*, (accepted), 2005.

[DC05c] N. Deo and A. Cami. Preferential deletion in dynamic models of web-like networks. *Information Processing Letters (submitted)*, 2005.

[DC05d] N. Deo and A. Cami. A survey of dynamic models of web-like networks. *Networks (submitted)*, 2005.

[DB02] Z. Dezso and A.-L. Barabási. Halting viruses in scale-free networks. *Physical Review. E, Statistical, Nonlinear, And Soft Matter Physics*, vol. 5, part 2, art. no. 055103, 2002.

[Die00] R. Diestel. *Graph theory*, 2nd ed. New York: Springer, 2000.

[DCLG00] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles and M. Gori. Focused Crawling Using Context Graphs. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB '00)*, pp. 527-534, 2000.

[DKMR01] S. Dill, R. Kumar, K. S. McCurley, S. Rajagopalan, D. Sivakumar and A. Tomkins. Self-similarity in the Web. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)*, pp. 69-78, 2001.

[DM00a] S. N. Dorogovtsev and J. F. F. Mendes. Evolution of networks with aging of sites. *Physical Review E (Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics)*, vol. 62(2), pp. 1842-1845, 2000.

[DM00b] S. N. Dorogovtsev and J. F. F. Mendes. Scaling behaviour of developing and decaying networks. *Europhysics Letters*, vol. 52(1), pp. 33-39, 2000.

[DMS00] S. N. Dorogovtsev, J. F. F. Mendes and A. N. Samukhin. Structure of growing networks with preferential linking. *Physical Review Letters*, vol. 85(21), pp. 4633-4636, 2000.

[DM01a] S. N. Dorogovtsev and J. F. F. Mendes. Effect of the accelerating growth of communications networks on their structure. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 2, part 1-2, art. no. 025101, 2001.

[DM01b] S. N. Dorogovtsev and J. F. F. Mendes. Scaling properties of scale-free evolving networks: Continuous approach. *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 5, part 1-2, art. no. 056125/1, 2001.

[DMS01b] S. N. Dorogovtsev, J. F. F. Mendes and A. N. Samukhin. Giant strongly connected component of directed networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 2, art. no. 025101, 2001.

[DMS01a] S. N. Dorogovtsev, J. F. F. Mendes and A. N. Samukhin. Anomalous percolation properties of growing networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 6, art. no. 066110, 2001.

[DGM02] S. N. Dorogovtsev, A. V. Goltsev and J. F. F. Mendes. Ising model on networks with an arbitrary distribution of connections. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 1, art. no. 016104/1, 2002.

[DM02] S. N. Dorogovtsev and J. F. F. Mendes. Evolution of networks. *Advances in Physics*, vol. 51(4), pp. 1079-1187, 2002.

[DGMS03] S. N. Dorogovtsev, A. V. Goltsev, J. F. F. Mendes and A. N. Samukhin. Spectra of complex networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 4, art. no. 46109, 2003.

[DM03] S. N. Dorogovtsev and J. F. F. Mendes. *Evolution of networks: from biological nets to the Internet and WWW*. Oxford: Oxford University Press, 2003.

[DMS03] S. N. Dorogovtsev, J. F. F. Mendes and A. N. Samukhin. Metric structure of random networks. *Nuclear Physics B*, vol. 653(3), pp. 307-338, 2003.

[DGMS04] S. N. Dorogovtsev, A. V. Goltsev, J. F. F. Mendes and A. N. Samukhin. Random networks: eigenvalue spectra. *Physica A: Statistical Mechanics and its Applications*, vol. 338(1-2), pp. 76-83, 2004.

[EK02] V. M. Eguíluz and K. Klemm. Epidemic threshold in structured scale-free networks. *Physical Review Letters*, vol. 89(10), art. no. 108701/1-4, 2002.

[EHPK03] V. M. Eguíluz, E. Hernandez-Garcia, O. Piro and K. Klemm. Effective dimensions and percolation in hierarchically structured scale-free networks. *Physical Review. E (Statistical, Nonlinear, And Soft Matter Physics)*, vol. 5, part 2, art. no. 055102, 2003.

[Ela99] S. Elaydi. *An introduction to difference equations*, 2nd ed. New York: Springer, 1999.

[ER59] P. Erdös and A. Rényi. On random graphs I. *Publ. Math. Debrecen*, vol. 6, pp. 290-

297, 1959.

[EG60] P. Erdös and T. Gallai. Graphs with Prescribed Degrees of Vertices. *Mat. Lapok*, vol. 11, pp. 264-274, 1960.

[ER60] P. Erdös and A. Rényi. On the evolution of random graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl.*, vol. 5, pp. 17-61, 1960.

[ER02] G. Ergün and G. J. Rodgers. Growing random networks with fitness. *Physica A*, vol. 303(1-2), pp. 261-272, 2002.

[FFF99] M. Faloutsos, P. Faloutsos and C. Faloutsos. On power-law relationships of the Internet topology. *Computer Communication Review*, vol. 29(4), pp. 251-262, 1999.

[FDBV01] I. J. Farkas, I. Derenyi, A. L. Barabási and T. Vicsek. Spectra of "real-world" graphs: beyond the semicircle law. *Physical Review. E (Statistical, Nonlinear, And Soft Matter Physics)*, vol. 2, part 2, art. no. 026704, 2001.

[FGLS91] U. Feige, S. Goldwasser, L. Lovász, S. Safra and M. Szegedy. Approximating clique is almost NP-complete. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, pp. 2-12, 1991.

[FMN04] D. Fetterly, M. Manasse and M. Najork. Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In *Proceedings of the 7th International Workshop on the Web and Databases (WebDB '04)*, pp. 1-6, 2004.

[Fie73] M. Fiedler. Algebraic connectivity of graphs. *Czech. Math. J.*, vol. 23, pp. 298-305, 1973.

[FPK89] P. Flajolet, D. E. Knuth and B. Pittel. The first cycles in an evolving graph. *Discrete Math.*, vol. 75(1-3), pp. 167-215, 1989.

[FLG00] G. W. Flake, S. Lawrence and C. L. Giles. Efficient identification of web communities. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00)*, pp. 150-160, 2000.

[FLGC02] G. W. Flake, S. Lawrence, C. L. Giles and F. M. Coetzee. Self-organization and identification of Web communities. *Computer*, vol. 35(3), pp. 66-71, 2002.

[FTT04] G. W. Flake, R. Tarjan and K. Tsioutsiouliklis. Graph Clustering and Minimum Cut Trees. *Internet Math.*, vol. 1(4), pp. 385-408, 2004.

[FFV05] A. Flaxman, A. Frieze and J. Vera. Adversarial Deletion in a Scale Free Random Graph Process. In *Proceedings of the ACM-SIAM Symposium on Doscrete Algorithms (SODA '05)*, pp. 287-292, 2005.

[FFV04] A. D. Flaxman, A. M. Frieze and J. Vera. A geometric preferential attachment model of networks. *Algorithms and Models for the Web-Graph. Third International Workshop, WAW 2004. Proceedings (Lecture Notes in Computer Science Vol. 3243)*, pp. 44-55, 2004.

[FM97] A. Frieze and C. McDiarmid. Algorithmic Random Graph Theory. *Random Structures & Algorithms*, vol. 10, pp. 5-42, 1997.

[FHJS02] A. Fronczak, J. A. Holyst, M. Jedynak and J. Sienkiewicz. Higher Order Clustering Coefficients in Barabasi-Albert Networks. *Physica A*, vol. 316, pp. 688-694, 2002.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. San Francisco: W. H. Freeman, 1979.

[GKR98] D. Gibson, J. Kleinberg and P. Raghavan. Inferring Web communities from link topology. In *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia (HYPERTEXT '98)*, pp. 225-234, 1998.

[Gil59] E. N. Gilbert. Random graphs. *Ann. Math. Statist.*, vol. 30, pp. 1141-1144, 1959.

[GN02] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. In *Proceedings of the National Academy Of Sciences of the USA*, pp. 7821-7826, 2002.

[GFLB01] E. J. Glover, G. W. Flake, S. Lawrence, W. P. Birmingham, A. Kruger, C. L. Giles and D. M. Pennock. Improving category specific Web search by learning query modifications. In *Proceedings of the 2001 Symposium on Applications and the Internet*, pp. 23-32, 2001.

[GLGB01] E. J. Glover, S. Lawrence, M. D. Gordon, W. P. Birmingham and C. L. Giles. Web search-your way. *Communications of the ACM*, vol. 44(12), pp. 97-102, 2001.

[GKK01] K.-I. Goh, B. Kahng and D. Kim. Spectra and eigenvectors of scale-free networks. *Physical Review. E, Statistical, Nonlinear, And Soft Matter Physics*, vol. 5, part1, art. no. 051903, 2001.

[GOJK02] K.-I. Goh, E. Oh, H. Jeong, B. Kahng and D. Kim. Classification of scale-free networks. *Proceedings Of The National Academy Of Sciences Of The United States Of America*, vol. 99(20), pp. 12583-12588, 2002.

[GKK03] K.-I. Goh, B. Kahng and D. Kim. Packet transport and load distribution in scale-free network models. *Physica A: Statistical Mechanics and its Applications*, vol. 318(1-2), pp. 72-79, 2003.

[GT00] R. Govindan and H. Tangmunarunkit. Heuristics for Internet map discovery. In *Proceedings of the IEEE INFOCOM 2000. Conference on Computer*

Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064), pp. 1371-1380, 2000.

[GGZ04] G. Greco, S. Greco and E. Zumpano. Web communities: models and algorithms. *World Wide Web*, vol. 7(1), pp. 59-82, 2004.

[GDDG03] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt and A. Arenas. Self-similar community structure in a network of human interactions. *Physical Review. E: Statistical, Nonlinear, And Soft Matter Physics*, vol. 6, art. no. 065103, 2003.

[GD02] P. Gupta and N. Deo. Analysis of graph-theoretic models for the world wide web. *Proceedings of the Thirty-third Southeastern International Conference on Combinatorics, Graph Theory and Computing (Boca Raton, FL, 2002)*, vol. 158, pp. 99-107, 2002.

[GD03] P. Gupta and N. Deo. Diameter of a random graph and its implications for the web graph. *Congressus Numerantium*, vol. 160, pp. 109-116, 2003.

[Hak62] S. L. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. *J. Soc. Indust. Appl. Math.*, vol. 10, pp. 496-506, 1962.

[HHH03] T. W. Haynes, S. T. Hedetniemi and M. A. Henning. Global defensive alliances in graphs. *Electron. J. Combin.*, vol. 10, pp. Research Paper 47, 13 (electronic), 2003.

[Hen03] M. R. Henzinger. Algorithmic challenges in web search engines. *Internet Math.*, vol. 1(1), pp. 115-123, 2003.

[HKKS04] J. Hopcroft, O. Khan, B. Kulis and B. Selman. Tracking evolving communities in large linked networks. In *Proceedings of the National Academy Of Sciences Of The USA*, pp. 5249-5253, 2004.

[HZ03] J. Hou and Y. Zhang. Utilizing hyperlink transitivity to improve web page clustering. In *Proceedings of the 14th Australasian Database Conference on Database Technologies 2003 (CRPITS'17)*, pp. 49-57, 2003.

[IK04] N. Imafuji and M. Kitsuregawa. Finding Web communities by maximum flow algorithm using well-assigned edge capacities. *IEICE Transactions on Information and Systems*, vol. E87-D(2), pp. 407-415, 2004.

[IMKZ03] S. Itzkovitz, R. Milo, N. Kashtan, G. Ziv and U. Alon. Subgraphs in random networks. *Physical Review. E: Statistical, Nonlinear, And Soft Matter Physics*, vol. 2, art. no. 026127, 2003.

[JKLP93] S. Janson, D. E. Knuth, T. Łuczak and B. Pittel. The birth of the giant component. *Random Structures Algorithms*, vol. 4(3), pp. 231-358, 1993.

[JLR00] S. Janson, T. Łuczak and A. Ruciński. *Random graphs*. New York: John Wiley, 2000.

[Jor65] C. Jordan. *Calculus of finite differences*, 3rd ed. New York, NY: Chelsea Publishing Company, 1965.

[JN03] P. Juyong and M. E. J. Newman. Origin of degree correlations in the Internet and other networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, vol. 2, art. no. 26112, 2003.

[KIMA04] N. Kashtan, S. Itzkovitz, R. Milo and U. Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics (Oxford, England)*, vol. 20(11), pp. 1746-1758, 2004.

[KL70] B. W. Kerrnighan and S. Lin. An efficient heuristic for graphs. *Bell Systems Tech. J.*, vol. 49, pp. 291-307, 1970.

[Kes63] M. Kessler. Bibliographic coupling betweeb scientific papers. *American Documentation*, vol. 14, pp. 10-25, 1963.

[Kle00] J. Kleinberg. The small-world phenomenon: an algorithmic perspective. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pp. 163-170, 2000.

[KKRR99] 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 (Tokyo, 1999)*, vol. 1627, pp. 1-17, 1999.

[KE02a] K. Klemm and V. M. Eguíluz. Growing scale-free networks with small-world behavior. *Physical Review. E, Statistical, Nonlinear, And Soft Matter Physics*, vol. 5, part 2, art. no. 057102, 2002.

[KE02b] K. Klemm and V. M. Eguíluz. Highly clustered scale-free networks. *Physical Review. E, Statistical, Nonlinear, And Soft Matter Physics*, vol. 3, part 2a, art. no. 036123, 2002.

[KR01] P. L. Krapivsky and S. Redner. Organization of growing random networks. *Physical Review. E, Statistical, Nonlinear, And Soft Matter Physics*, vol. 63(6, part 2), art. no. 066123, 2001.

[KRR01] P. L. Krapivsky, G. J. Rodgers and S. Redner. Degree distributions of growing networks. *Physical Review Letters*, vol. 86(23), pp. 5401-5404, 2001.

[KR03] P. L. Krapivsky and S. Redner. Rate equation approach for growing networks. *Statistical Mechanics of Complex Networks. 18th Sitges Conference*, pp. 3-22, 2003.

[BBHK98] Krishna Bharat, Andrei Broder, Monika Henzinger, P. Kumar and S. Venkatasubramanian. The Connectivity Server: fast access to linkage information on the Web. *Computer Networks and ISDN Systems*, vol. 30(1-7), pp. 469-477, 1998.

[KHH04] P. Kristiansen, S. M. Hedetniemi and S. T. Hedetniemi. Alliances in graphs. *J. Combin. Math. Combin. Comput.*, vol. 48, pp. 157-177, 2004.

[KRRT99] R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins. Trawling the Web for emerging cyber-communities. *Computer Networks*, vol. 31(11-16), pp. 1481, 1999.

[KRRS00] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins and E. Upfal. Stochastic models for the Web graph. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pp. 57-65, 2000.

[KMS04] R. Kumar, U. Mahadevan and D. Sivakumar. A graph-theoretic approach to extract storylines from search results. In *KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*: ACM Press, pp. 216-225, 2004.

[LG00] S. Lawrence and C. L. Giles. Accessibility of information on the web. *Intelligence*, vol. 11(1), pp. 32-39, 2000.

[LPFK01] S. Lawrence, D. M. Pennock, G. W. Flake, R. Krovetz, F. M. Coetzee, E. Glover, F. A. Nielsen, A. Kruger and C. L. Giles. Persistence of Web references in scientific research. *Computer*, vol. 34(2), pp. 26-31, 2001.

[LD04] B. Litow and N. Deo. Graph Compression and Zeros of Polynomials. *Information Processing Letters*, vol. 92, pp. 39-44, 2004.

[LDC04] B. Litow, N. Deo and A. Cami. Compression of Vertex Transitive Graphs. In *Proceedings of the 35th Southeastern International Conference on Combinatorics, Graph Theory and Computing (Congressus Numerantium, vol. 167)*, pp. 161-173, 2004.

[Luc98] T. Łuczak. Random trees and random graphs. *Random Structures & Algorithms*, vol. 13, pp. 485-500, 1998.

[MW97] B. D. McKay and N. C. Wormald. The degree sequence of a random graph. I. The models. *Random Structures & Algorithms*, vol. 11(2), pp. 97-117, 1997.

[MPS04] F. Menczer, G. Pant and P. Srinivasan. Topical web crawlers: Evaluating adaptive algorithms. *ACM Trans. Inter. Tech.*, vol. 4(4), pp. 378-419, 2004.

[MSIK02] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii and U. Alon. Network motifs: simple building blocks of complex networks. vol. 298(5594), pp. 824-827, 2002.

[MSIK02] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, vol. 298(5594), pp. 824-827, 2002.

[MIKL04] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer and U. Alon. Superfamilies of evolved and designed networks. *Science*, vol. 303(5663), pp. 1538-1542, 2004.

[Mit04] M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Math.*, vol. 1(2), pp. 226-251, 2004.

[MR95] M. Molloy and B. Reed. A critical point for random graphs with a given degree sequence. *Random Structures Algorithms*, vol. 6(2-3), pp. 161-179, 1995.

[MR98] M. Molloy and B. Reed. The size of the giant component of a random graph with a given degree sequence. *Combin. Probab. Comput.*, vol. 7(3), pp. 295-305, 1998.

[MN00] C. Moore and M. E. Newman. Exact solution of site and bond percolation on small-world networks. *Physical Review. E, Statistical Physics, Plasmas, Fluids, And Related Interdisciplinary Topics*, vol. 62(5, part b), pp. 7059-7064, 2000.

[NMW00] M. E. Newman, C. Moore and D. J. Watts. Mean-field solution of the small-world network model. *Physical Review Letters*, vol. 84(14), pp. 3201-3204, 2000.

[New01] M. E. Newman. Clustering and preferential attachment in growing networks. *Physical Review. E: Statistical, Nonlinear, And Soft Matter Physics*, vol. 2, part 2, art. no. 025102, 2001.

[NSW01] M. E. Newman, S. H. Strogatz and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review. E, Statistical, Nonlinear, And Soft Matter Physics*, vol. 64(2, part 2), art. no. 026118, 2001.

[NW99] M. E. J. Newman and D. J. Watts. Renormalization group analysis of the small-world network model. *Phys. Lett. A*, vol. 263(4-6), pp. 341-346, 1999.

[New03a] M. E. J. Newman. Mixing patterns in networks. *Physical Review. E (Statistical, Nonlinear, And Soft Matter Physics)*, vol. 2, part 2, art. no. 026126, 2003.

[New03b] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, vol. 45(2), pp. 167-256, 2003.

[New04] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review. E, Statistical, Nonlinear, And Soft Matter Physics*, vol. 69(6, part 2), art. no. 066133, 2004.

[PBMW98] L. Page, S. Brin, R. Motwani and T. Winograd. The Pagerank citation ranking: Bringing order to the web. Stanford Digital Library Technologies Project, Technical Report 1998.

[PS00] C. R. Palmer and J. G. Steffan. Generating network topologies that obey power laws. In *Proceedings of the Globecom'00 - IEEE Global Telecommunications Conference. Conference Record (Cat. No.00CH37137)*, pp. 434-438, 2000.

[Pal85] E. M. Palmer. *Graphical evolution*. New York: Wiley, 1985.

[Pel01] M. Pelillo. Heuristics for maximum clique and independent set. In *Encyclopedia of optimization*, vol. 2, C. A. Floudas and P. M. Pardalos, (Eds.): Kluwer Academic Publishers, pp. 411-423, 2001.

[PFLU00] A. Popescul, G. W. Flake, S. Lawrence, L. H. Ungar and C. L. Giles. Clustering and identifying temporal trends in document databases. In *Proceedings of the IEEE Advances in Digital Libraries 2000*, pp. 173-182, 2000.

[RCCL04] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto and D. Parisi. Defining and identifying communities in networks. In *Proceedings of the National Academy Of Sciences Of The USA*, pp. 2658-2663, 2004.

[RG03] S. Raghavan and H. Garcia-Molina. Representing Web graphs. In *Proceedings of the 19th International Conference on Data Engineering (Cat. No.03CH37405)*, pp. 405-416, 2003.

[RC02] A. Ranganathan and R. H. Campbell. Advertising in a pervasive computing environment. In *Proceedings of the 2nd International Workshop on Mobile Commerce (WMC '02)*, pp. 10-14, 2002.

[RC02] A. Ranganathan and R. H. Campbell. Advertising in a pervasive computing environment. In *Proceedings of the 2nd International Workshop on Mobile Commerce (WMC '02)*, pp. 10-14, 2002.

[RK02] P. K. Reddy and M. Kitsuregawa. An approach to relate the Web communities through bipartite graphs. In *Proceedings of the 2nd International Conference on Web Information Systems Engineering*, pp. 301-310, 2002.

[SW04] T. Schank and D. Wagner. Approximating the clustering coefficient and transitivity. University of Karlsruhe, Technical Report 2004-9, 2004.

[SCBB02] N. Schwartz, R. Cohen, D. Ben-Avraham, A. L. Barabási and S. Havlin. Percolation in directed scale-free networks. *Physical Review. E (Statistical, Nonlinear, And Soft Matter Physics)*, vol. 1, part 2, art. no. 015104, 2002.

[Sma73] H. Small. Co-citation in the scientific literature: A new measure of the relationship

between two documents. *J. Am. Soc. for Inf. Sci.*, vol. 24(4), pp. 265-269, 1973.

[SAK02] G. Szabó, M. Alava and J. Kertész. Shortest paths and load scaling in scale-free trees. *Physical Review. E, Statistical, Nonlinear, And Soft Matter Physics*, vol. 66(2, part 2), art. no. 026101, 2002.

[SAK03a] G. Szabó, M. Alava and J. Kertész. Structural transitions in scale-free networks. *Physical Review. E, Statistical, Nonlinear, And Soft Matter Physics*, vol. 5, Part 2, art. no. 056102, 2003.

[SAK03b] G. J. Szabó, M. Alava and J. Kertész. Geometry of minimum spanning trees scale-free networks. *Physica A*, vol. 330(1-2), pp. 31-36, 2003.

[TOSK03] T. Tomiyama, R. Ohgaya, A. Shinmura, T. Kawabata, T. Takagi and M. Nikravesh. Concept-based Web communities for Google TM search engine. In *Proceedings of the 12th IEEE International Conference on Fuzzy Systems (Cat. No.03CH37442)*, pp. 1122-1128, 2003.

[VPV02] A. Vazquez, R. Pastor-Satorras and A. Vespignani. Large-scale topological and dynamical properties of the Internet. *Physical Review. E (Statistical, Nonlinear, And Soft Matter Physics)*, vol. 6, part 2, art. no. 066130, 2002.

[Vaz03] A. Vazquez. Growing Networks with Local Rules: Preferential Attachment, Clustering Hierarchy and Degree Correlations. *Physical Review E (Statistical, Nonlinear, And Soft Matter Physics)*, vol. 67, art. no. 056104, 2003.

[WF94] S. Wasserman and K. Faust. *Social network analysis: methods and applications*. Cambridge; New York: Cambridge University Press, 1994.

[WS98] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, vol. 393(6684), pp. 440-442, 1998.

[Wat03] D. J. Watts. *Six degrees: the science of a connected age*. New York, NY: W.W. Norton, 2003.

[Wil90] H. S. Wilf. *generatingfunctionology*. Boston: Academic Press, 1990.

[WH04] F. Wu and B. Huberman. Finding communities in linear time: a physics approach. *European Physical Journal B*, vol. 38(2), pp. 331-338, 2004.

[WCS04] K.-J. Wu, M.-C. Chen and Y. Sun. Automatic topics discovery from hyperlinked documents. *Inf. Process. Manage.*, vol. 40(2), pp. 239-255, 2004.

[YJB02] S.-H. Yook, H. Jeong and A.-L. Barabási. Modeling the Internet's large-scale topology. *Proceedings Of The National Academy Of Sciences Of The United States Of America*, vol. 99(21), pp. 13382-13386, 2002.

[YJBT01] S. H. Yook, H. Jeong, A. L. Barabási and Y. Tu. Weighted evolving networks. *Physical Review Letters*, vol. 86(25), pp. 5835-5838, 2001.

[Zac77] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, vol. 3, pp. 452-473, 1977.

[ZD03] Y. Zhang and N. Deo. Expected Value of the Diameter of a Random Graph. *Congressus Numerantium*, vol. 161, pp. 211-221, 2003.