Seventh FRAMEWORK PROGRAMME
FP7-ICT-2009-5 - ICT-2009-1.6
Future Internet experimentally-driven research

SPECIFIC TARGETED RESEARCH OR INNOVATION PROJECT

Deliverable D3.1

"Graph-based topology modelling"

Project description
Project acronym: **EULER** Project full title: **Experimental UpdateLess Evolutive Routing** Grant Agreement no.: **258307**
Document properties
Number: **FP7-ICT-2009-5-1.6-258307-D3.1** Title: **Graph-based topology modelling** Responsible: **Computer Technology Institute & Press "Diophantus" (EL)** Contributor(s): **All partners** Dissemination level: **Public (PU)** Date of preparation: **December 2011** Version: **1.0**

Contents

Context and document summary

Since its inception four decades ago, the Internet has now turned into a major success. During the recent years, it has rapidly evolved both in terms of scale and of available functionalities. At first glance, this success may be in contrast to the traditional principles of system design as, even if it was never centrally designed to evolve the way we know it today, it has done so based on a rather mysterious self-evolving process. Today, several sciences have included in their research agendas the challenging goal of explaining Internet evolution. This includes a deep understanding of its evolution up to now as well as convincing predictions about its future.

Clearly, whether Internet will keep on being successful strongly depends on the performance of its core functionality, routing. Somewhat surprisingly, routing Internet traffic seems to be performed at acceptable levels today. However, our vision of the Internet as the global medium for future communication and information exchange (e.g., the so-called Internet of things) goes far beyond its current scale. Sustaining routing performance is thus a major objective of Internet research, and the central objective of EULER.

EULER follows a multi-dimensional approach in order to achieve the ambitious goal of improving today's Internet routing and guarantee sustainability for the decades to come. One of these dimensions includes a deep understanding of the Internet topological structure using graph-theoretical notions, models, and techniques. Within EULER, we strongly believe that such an understanding will provide invaluable information for improving routing performance in at least two respects. First, identifying particular (probably hidden) structural properties that are inherent in the underlying graph representing connections between Internet routers and/or autonomous systems and relate them to the behavior of different routing algorithms will be informative on which is the best routing alternative. Secondly, new routing algorithms could exploit the specific properties of this graph by explicitly including the particular property values into their specification.

The investigation of the interrelation of graph properties with realistic models of the Internet graph is the general subject of the activities performed within Task T3.1. The ultimate goal is to shed light on how routing is affected by the graph properties and thus provide input to the design of new routing algorithms (this suggests a strong connection with Task T2.2). Traditionally, Graph Theory has studied extensively concrete random graph models, such as the now-classical Erdős-Rényi model [58], and has provided rigorous characterizations for several structural properties in these graphs. However, as it has become apparent during the last fifteen years, even though they have been proved to be important in several settings, such models can explain neither the evolution of the Internet nor its structural properties. For example, two properties that seem to characterize the Internet graph are a power law on the distribution of the node degrees and a high clustering coefficient (informally meaning that two nodes that are neighbors of the same node are more likely to be connected).

Starting from the seminal work of Barabasi and Albert [13], a series of studies have proposed (random) graph models that produce graphs with properties close to the ones observable in the Internet. Still, there is not a unique model that is widely acknowledged to accurately produce Internet-like graphs. During the first year of EULER, our activities within Task T3.1 have focused on a careful investigation of the related models proposed in the literature and an as extensive as possible selection of graph properties that are more likely to affect routing performance. Besides a thorough investigation of the related literature and new theoretical results, we have also implemented several graph models and algorithms for property testing on which our activities will be heavily based during the next period. Preliminary results from this implementation are already available and are reported in this document.

The rest of this document is structured as follows. In Chapter 1, we survey random graph models that have been proposed in the literature. Our reference point is the Erdős-Rényi model but our focus is on the most widely acceptable models for properties observable on the Internet. A long list of graph properties are presented in Chapter 2. Our focus is on properties that, intuitively, are likely to affect routing performance. In Chapter 3, we present an experimental investigation of properties on different graph models. This investigation gives a flavor of the approach that we are using in the context of Task T3.1. We remark that, for several properties, measuring their values is a highly non-trivial task given the size of realistic topologies. For example, the definition of hyperbolicity (see Section 2.15) implies an algorithm that runs in time $\Omega(n^4)$, where n is the number of network nodes. Clearly, it is prohibitive to compute the exact hyperbolicity value in snapshots of the Internet topology (e.g., CAIDA maps); so, efficiently computable approximate solutions are sought here. In addition, the exploitation of particular property values from a routing algorithm will be possible only if this property can be computed in a decentralized way. So, investigating the strengths and limitations of distributed algorithms for property testing is an important intermediate task; such an investigation is discussed in Section 3.3. Many of the preliminary results presented in Chapter 3 have been made possible by the use of experimental tools that will be further exploited in the context of Task T4.2. We present a brief overview of them (the libraries *Grph* and Sage) in Chapter 4. We conclude with future plans in Chapter 5.

Chapter 1

Topology models

This chapter provides an overview of existing models for generating random graphs. Our purpose is to present models that generate graphs with structural properties that are close to those observed on today's Internet. Our focus is both on static models as well as on dynamic ones. We begin with the well known Erdős Rényi models; even though these models do not really reflect the structure of modern networks, they have found many applications and can serve as a basic reference model.

1.1 Erdős-Rényi models

The Erdős-Rényi (ER) model [58] generates a graph uniformly at random among the ones having a given number of edges (and nodes). Two variants of the model have been proposed in the literature:

- $G(n, m)$: this model creates a graph with n nodes and m edges that are randomly chosen among the $n(n-1)/2$ possible ones with replacement.
- $G(n, p)$: this model creates a graph with n nodes and each edge is created independently with a given probability p.

The obtained graphs have Poisson degree distributions and low clustering coefficient. Therefore, they do not actually model Internet-like networks. More precisely, concerning the degree distribution of the Erdős-Rényi $G(n, p)$ random graph model, the probability p_k of a vertex to have degree k is $p_k = \binom{n}{k} p^k (1-p)^{n-k}$, and, as n grows to infinity, we have $p_k \sim \frac{ze^{-z}}{k!}$ with $z = p(n-1)$; this is a Poisson distribution. The clustering coefficient is $CC_{\text{random}} = \frac{<k>}{n}$, where $< k >$ is the average degree [34]. This means that the clustering coefficient of a random graph is close to zero for a graph with a large number of nodes. The clustering coefficient, however, is independent of the number of nodes for many real-world networks. Furthermore, many real world networks have stronger ties within a community than outside and this property does not appear in random graphs produced by the ER model.

1.2 Power law degree based models

We proceed by presenting a series of models that preserve a power law property in the degree distribution. We distinguish between non-evolutive and evolutive models.

1.2.1 Non evolutive

Power law random graph models retain the simplicity and ease of analysis of the Erdős-Rényi model, while removing its weaknesses. As such weaknesses in the ER model, we consider the unrealistic Poisson degree distribution and the fact that all node degrees have the same expected value. These properties are not verified in many large graphs that arise in various applications, where they have diverse degree distributions (see [5, 13, 83]).

However, most such models only attempt to match the degree distribution of real graphs and not of other patterns. In most random graph models, the probability that two neighbors of a node are themselves connected is in the order of $O(N^{-1})$ where N is the number of nodes in the graph G. This is exactly the clustering coefficient of the graph and approaches zero for large N. For many real-world graphs, $\frac{CC}{<k>}$ where $< k >$ is the average degree of G, is independent of N [7]. Also, many real world graphs (such as the WWW) exhibit the existence of communities of nodes with stronger ties within the community than outside; ER random graphs do not show such a behavior.

Non-evolutive models can be further sub-divided as follows:

- Models defined by a given degree sequence: models that follow this approach include the Power Law Random Graph (PLRG) model of Aiello et al. [4] and the Exponential Cutoffs model of Newman, Strogatz, and Watts [109].
- Models defined by an expected degree sequence: the Generalized Random Graph (GRG) model of Chung [41] belongs in this category.

Even though, by definition, these models produce power law degree distributions, they do not provide insights about how a network comes to have such a degree distribution.

Power law random graph (PLRG). Given a power law degree distribution, node degrees are randomly assigned to match the given distribution. Edges are formed by randomly linking two nodes till no node has extra degrees left. There are two parameters: α and β and the number of nodes of degree k is given by $\frac{e^\alpha}{k^\beta}$. Hence, the PLRG degree distribution is $p_k \propto k^{-\beta}$ where β is the power law exponent.

The PLRG model belongs to the family of random graph models and creates undirected graphs with possible self-loops and multiedges. It is a static model and exhibits the following properties: it follows a power law, as discussed above, with α and β being the user defined exponents, its diameter is $O(\log N)$ while the clustering coefficient approaches 0 for large values of N. Some connectivity properties of $p(\alpha, \beta)$ as a function of β are presented in the following:

- when $\beta < 1$, the graph is almost surely connected,
- for $1 < \beta < 2$, a giant component exists, and smaller components are of size $O(1)$,
- for $2 < \beta < \beta_0 \sim 3.48$, a giant component exists, and the smaller components are of size $O(\log N)$,
- for $\beta = \beta_0$, the smaller components are of size $O(\log N / \log \log N)$, and
- for $\beta > \beta_0$, no giant component exists.

Thus, regarding the appearance of a giant component, there is a phase transition at $\beta = \beta_0 = 3.48$; there is also a change in the size of the smaller components at $\beta = 2$.

The exponential cutoffs model. This is another method to generate graphs with a given degree distribution. Now, the node degrees are independent identically distributed random integers drawn from a given degree distribution. For a choice of these degrees, called the degree sequence, a graph is chosen uniformly at random among the set of graphs with this particular degree sequence. Node degrees are randomly assigned to match the given degree sequence. Edges are formed by randomly linking two nodes till no node has extra degrees left. There are three parameters: C, γ and κ, the cutoff parameter. The probability that a node has k edges is given by

$$p_k \quad = \quad Ck^{-\gamma}e^{-k/\kappa}. \tag{1.1}$$

The results of this model are consistent with those of PLRG when $\kappa \to \infty$. The model generates undirected graphs with possible self-loops and multiedges as well. It is static and exhibits the following properties: its clustering coefficient approaches 0 as the graph size N increases and the diameter is $O(\log N)$. Analytic expressions are known for the average path length of the produced graphs and this typically tends to be somewhat less than that in real-world graphs (see [14]).

Generalized random graph model. The model is specified in [41]. It generates graphs with a given expected degree sequence $\bar{k} = (k_1, \ldots, k_n)$ for vertices $v_1, \ldots, v_n$. The edge between vertices v_i and v_j is chosen independently with probability p_{ij}, with $p_{ij} = \rho k_i k_j$, where $\rho = \frac{1}{\sum_i k_i}$. Furthermore, it must be $\max_i k_i^2 < \sum_i k_i$ so that the probability p_{ij} is well-defined and in order to ensure that the degree sequence $\bar{k}$ can indeed be realized. This defines a probability measure P on the space of all simple graphs and thus induces a probability measure on $G(k)$ by conditioning on having degree k. The classical random graph $G(n,p)$ can be viewed as a special case of $G(\bar{k})$ by taking $\bar{k}$ to be $(pn, pn, \ldots, pn)$. Some indicative properties of the produced graphs (see also [42]) are the following:

- All generated graphs typically have a unique large connected component that contains a constant fraction of the edges.
- When $\beta = 3$, the average distance between pairs of nodes in the same connected component is $\Theta(\log n / \log \tilde{d}) = \Theta(\log n / \log \log n)$, where $\tilde{d}$ denotes the second-order average degree.
- When $2 < \beta < 3$, the average distance between pairs of nodes is almost surely $O(\log \log n)$ if the average degree is strictly greater than 1 and the maximum degree is sufficiently large.
- When $\beta = 3$ the graph has diameter almost surely $\Theta(\log n)$.
- When $2 < \beta < 3$ the diameter of the core is almost surely $O(\log \log n)$.

1.2.2 Evolutive models

Evolutive models usually apply one of the following two mechanisms:

- Preferential attachment: New nodes are attached to nodes that are already well connected; new nodes prefer existing ones with high degree. More formally, the probability

Name	PA model	Characteristics	Reference
Linear preferential attachment (BA)	$\Pi(k_i) \approx k_i$	Produces a graph whose node degrees follow a power law distribution with exponent (or scale index) $\gamma = 3 : p(k) \propto k^{-3}$	Barabasi and Albert [13]
Linear preferential attachment with rewiring (AB)	$\Pi(k_i) \approx k_i + 1$		Albert and Barabasi [6]
Generalized linear preferential attachment (GLP)	$\Pi(k_i) \approx k_i - \beta$		Bu and Townsley [32]
Linear preferential attachment with initial attractiveness (DM)	$\Pi(k_{in,i}) \approx k_{in,i} + A$	Produces a graph whose node in-degrees follow a power law distribution with exponent (or scale index) $\gamma = 2 + A/m$, where $A \geq 0$ is the initial attractiveness of a site	Dorogovtsev et al. [54, 101]
Linear preferential attachment with intrinsic fitness	$\Pi(k_i) \approx \eta_i k_i$	When fitness parameters η_i are drawn randomly from a uniform [0,1] distribution, the degree distribution $p(k) \propto k^{-(C+1)}/\log k$ with exponent $C + 1 = 2.255$	Bianconi and Barabasi [18]
Non-linear preferential attachment (KR)	$\Pi(k_i) \approx k_i^{\alpha}$	For $\alpha \neq 1$ does not preserve scale-free nature of the network	Krapivsky et al. [86]

Table 1.1: Variants of evolutive models.

$\Pi(k_i)$ that a new node will select an existing node i depends on the degree k_i of node i. The underlying principle finds its root in the "rich get richer principle". This is measured by the assortativity coefficient.

- Incremental growth: New nodes and edges are continuously added to the graph. Edges can also be "rewired".

Table 1.1 summarizes the different models and their variants that have been studied since the seminal work of Barabasi and Albert [13]. The models that are considered below are those that introduce a new concept that extends the initial paradigm developed in [13].

Note that the Barabasi-Albert (BA) model is not the only one that has been extended to incorporate rewiring (which has led to the AB model). Similar extensions have been proposed in the literature for the DM [54, 101] and KR [86] model. The main limitation of the BA model or its rewiring variants is that it does not reproduce well the clustering coefficient observed in the Internet topology. The GLP model has been explicitly defined so as to better reproduce this property.

Linear preferential attachment (Barabasi-Albert model). The BA model generates an undirected network in the following way. It starts with m_0 nodes, where m_0 is "small" and, at each step, one node is added along with $m \leq m_0$ edges which link the new node to m existing nodes (this models the growth). The probability $\Pi(k_i)$ of choosing one of the existing nodes i of degree k_i as an endpoint for the new edges is given by

$$\Pi(k_i) = \frac{k_i}{\sum_{j=1}^{N} k_j}. \tag{1.2}$$

The effect of this equation is that nodes which already have many edges connected to them will get even more edges (this reflects the "rich get richer" scenario mentioned above). After t

steps the model leads to a random network topology whose number of nodes is $N(t) = m_0 + t$ and the number of links is $E(t) = e_0 + mt$.

Continuum Theory can be used to calculate the dependence of the degree k_i of a given node i on time. This degree will increase every time a new node enters the system and is connected to node i; the probability of this event is $\Pi(k_i)$. Assuming that k_i is a continuous real variable $k_i(t)$, the rate at which k_i changes over time is expected to be proportional to $\Pi(k_i)$. Consequently, k_i satisfies the following expression

$$\frac{\partial k_i}{\partial t} = m\Pi(k_i) = m\frac{k_i}{\sum_{j=1}^{N} k_j}. \tag{1.3}$$

Since $\sum_{j=1}^{N} k_j = 2E(t) - m = 2m(t-1)$, we obtain

$$\frac{\partial k_i}{\partial t} = m\frac{k_i}{\sum_{j=1}^{N} k_j} = m\frac{k_i}{2m(t-1)} = \frac{k_i}{2(t-1)}. \tag{1.4}$$

The solution of this equation, with the initial condition that vertex i added at time t_i verifies $k_i(t_i) = m$, is

$$k_i(t) = m\left(\frac{t}{t_i}\right)^{\beta}, \beta = 1/2. \tag{1.5}$$

This equation indicates that the degree of all nodes evolve in the same way and increases with time as a power law with exponent $\beta = 1/2$.

The BA model creates scale-free undirected graphs with a single connected component and constant average degree. It is static and exhibits the following properties. The average degree is $< k >= 2E/N \to 2m$, the node degree distribution follows $p(k) \to 2m^2k^{-3}$ for $t \to \infty$. Thus, the probability that a node has k links follows a power law with exponent $\gamma = 3$. In [27] it is shown that, for large N the diameter grows as $O(\log N)$ for $m = 1$ and as $O(\log N/\log\log N)$ for $m \geq 2$. Thus, this model displays the small-world effect. The clustering coefficient is $CC \propto N^{-0.75}$. In [85], the authors identify two correlations in the BA model. First, the degree and age of nodes are positively correlated, i.e., older nodes have a higher mean degree. The second correlation is that nodes with similar degree are more likely to be connected (see assortativity in Section 2.10). However, this probability goes to 0 as $N \to \infty$.

Linear preferential attachment with rewiring (Albert-Barabasi model). The BA model incorporates only one mechanism for incremental network growth: the addition of new nodes that connect to the nodes already in the system. However, in real systems, a series of microscopic events shape the network evolution, including the addition or rewiring of new edges or the removal of nodes or edges. Any local change in the network topology can be obtained through a combination of four elementary processes: addition or removal of a node and addition or removal of an edge. However, in reality, these events happen simultaneously. For example, the rewiring of an edge is a combination of an edge removal and addition. Starting with a small set of m_0 nodes, the AB model combines three operational processes:

- With probability p, add m ($m \leq m_0$) new edges. One of the endpoints of each new edge is randomly selected and the other endpoint is preferentially selected so that the probability that it is node i with degree k_i is $\Pi(k_i) = \frac{k_i+1}{\sum_{j=1}^{N}(k_j+1)}$.

- With probability q, rewire m links. This means to randomly select node i and then choose one of its incident edges. This edge is removed and replaced by a new edge $e_{ij'}$, reconnecting node i to some other node j' chosen with the probability given above. This process is then repeated m times.

- With probability $1-p-q$, add a new node with m edges. One endpoint of these m edges is the new node; the other endpoints are selected among the nodes already present in the system with probability $\Pi(k_i)$. This was the only process used in the BA model.

The probability that a node i increases its degree depends only on k_i and the quantities that characterize the network (i.e., the parameters p, q, m and the number of nodes and links). Assuming that k_i changes continuously, the probability $\Pi(k_i)$ can be interpreted as the rate at which k_i changes over time. The AB model is both scale-free and exponential. It creates undirected graphs with possible self-loops and multiedges. It is static and exhibits either a power law or an exponential node degree distribution depending on q. In particular, it belongs to the scale-free regime when $q < q_{max} = \min(1-p, (1-p+m)/(1+2m))$. In this case, the degree distribution is a power law with exponent $\gamma = \frac{2m(1-q)+1-p-q}{m} + 1$. While a power law tail is present in any point of this regime, the scaling is different from that predicted by the simpler scale-free BA model. First, for small k the probability saturates at $p(\kappa(p,q,m))$. Second, the exponent $\gamma(p,q,m)$, characterizing the tail of $p(k)$ for $k >> \kappa(p,q)$, changes continuously with p, q, and m, predicting a range of exponents between 2 and ∞. This allows to account for the wide variations seen in real networks, for which γ varies from 2 to 3. Furthemore, it belongs to the exponential regime when $q > q_{max}$. In this case, continuum theory fails to predict the behavior of the system. In this regime, evaluation of $p(k)$ by means of numerical simulations shows that $p(k)$ develops an exponential tail as $q \to 1$. This transition to an exponential demonstrates that growth is an essential condition for power law scaling. The exponential behavior of $p(k)$ in this regime indicates that for this choice of parameters the model belongs to the class of networks defined by the ER model.

Generalized linear preferential attachment (GLP). In [37], the authors show that the preference for connecting to high-degree nodes is stronger than that predicted by linear preferential attachment. The GLP model removes the edge rewiring process and modifies the linear preferential attachment equation of the AB model to achieve higher preference for nodes with high degrees. It couples preferential attachment with the incremental growth mechanism in order to generate topologies that more closely model the Internet, its clustering coefficient, and the characteristic path lengths. For this purpose, GLP considers addition of new links between existing nodes. Henceforth, it captures two events corresponding to the addition of a new node and the addition of a link. A graph is constructed in the following way. We begin with m_0 nodes connected through $m_0 - 1$ edges and, at each time step, we perform one of the following two operations:

- With probability p we add $m \leq m_0$ new links. Node i is chosen with probability $\Pi(k_i)$ as the endpoint of such a link. This operation incorporates the fact that new links preferentially connect to popular nodes.

- With probability $1-p$ we add a new node with m new links. Each link is connected to node i already present in the system with probability $\Pi(k_i)$.

In the GLP model, the preferential attachment probability for choosing node i with degree k_i is defined as follows:

$$\Pi(k_i) = \frac{k_i - \beta}{\sum_{j=1}^{N}(k_j - \beta)}, \tag{1.6}$$

where $\beta \in (-\infty, 1)$ is a tunable parameter that governs the GLP process and indicates the preference for a new node (edge) connected to more popular nodes. The smaller the value of β, the less preference is given to high degree nodes. Being a tunable parameter, β can be adjusted such that nodes have a stronger preference for high degree nodes than in the BA model and, since $\beta < 1$ by definition, there is a nonzero probability that nodes of degree one acquire new links.

GLP belongs to the family of scale-free models and creates undirected graphs with self-loops and multiedges. It is static and satisfies the following properties. The degree distribution is a power law with exponent

$$\gamma = \frac{2m - \beta(1-p)}{(1+p)m} + 1, \tag{1.7}$$

while the clustering coefficient is closer to that of the Internet compared to the BA, AB, and PLRG models.

Linear preferential attachment with initial attractiveness (Dorogovtsev-Mendes model). While the BA model generates graphs with a power law degree distribution, the power law exponent is $\gamma = 3$. This value does not reproduce the exponent observed for evolving networks such as the Internet. For this purpose, Dorogovtsev et al. [54, 101] propose a one-parameter extension of the BA model which allows γ to take values in the range $[2, \infty)$. The DM model works as follows. At each time step, a new vertex is added to the network, and, simultaneously, m new directed edges appear that have origins at non-specified vertices or even from outside of the network. Note the difference with the BA model. The target endpoints of the new edges are distributed among vertices according to the following rule. The probability $\Pi(q_i)$ that a new edge points to some vertex i is proportional to $q_i + A$, where q_i is the in-degree of node i while the parameter A models the initial attractiveness of each site, and governs the probability that young sites gain new edges. Note that if each new vertex is the source of all the m new edges then $k_i = q_i + m$, and the degree of each vertex is determined by its in-degree. If, in addition, $A = m$, then new edges are distributed with probability proportional to k_i; in this case, the DM model is equivalent to the BA model.

The DM model creates scale-free directed graphs with possible self-loops and multiedges. It is static and the degree distribution is found to be a power law with exponent

$$\gamma = 2 + \frac{A}{m}. \tag{1.8}$$

Thus, depending on the values of A and m, γ can take any value in $[2, \infty)$. It is important to observe that the initial attractiveness does not destroy the scale-free nature of the degree distribution; it only changes the exponent.

Linear preferential attachment with intrinsic fitness (Bianconi-Barabasi model). The BA model assumes that all nodes increase their degree following a power law time dependence with the same dynamic exponent $\beta = 1/2$. Recall that, using Continuum Theory, we have obtained that $k_i(t) = m(\frac{t}{t_i})^{\beta}$ with $\beta = 1/2$. As a consequence, the oldest nodes have

the highest number of edges, since they had the longest lifetime to accumulate them. However, numerous examples indicate that in real networks the degree of a node and its growth rate do not depend only on its age. For example, on the Web, some documents acquire a large number of edges in a very short time through a combination of interesting content and marketing strategies [3], and some research papers acquire many more citations than others. Several studies have offered models that address this phenomenon.

Bianconi and Barabasi [18] propose a model that assumes the existence of a fitness characteristic of the nodes that modifies the preferential attachment and models their competition for links. The different fitness translates into multiscaling in the dynamical evolution: node degrees depend not only on time but also on their fitness. This model assigns to each node i a fixed fitness parameter η_i that represents the intrinsic ability of the node to compete for edges. The idea is that a node which is added later could overtake older nodes in terms of degree if the newer node has a much higher fitness value. A graph is constructed as follows. We begin with a small number of nodes m_0 and, at every step, we add a new node i with fitness η_i were η is chosen from the distribution $\rho(\eta)$. Each new node i has m links that are connected to the nodes already present in the system. The probability Π_i that a new node will connect to an already present node i depends on the degree k_i and on the fitness η_i of that node:

$$\Pi_i = \frac{\eta_i k_i}{\sum_{j=1}^{N} \eta_j k_j}. \tag{1.9}$$

This formulation ensures that fitness and degree jointly determine the rate at which new edges are attached to a given node.

Again, Continuum Theory enables us to predict that node i will increase its degree k_i at a rate that is proportional to the probability Π_i and to the parameter m according to the following expression:

$$\frac{\partial k_i}{\partial t} = m\Pi(k_i) = m\frac{\eta_i k_i}{\sum_{j=1}^{N} \eta_j k_j}. \tag{1.10}$$

Assuming that the time evolution of k_i follows a power law with a dynamic exponent $\beta(\eta_i)$ depending on the fitness, we have

$$k_i(t, t_i) \quad = \quad m(\frac{t}{t_i})^{\beta(\eta_i)}. \tag{1.11}$$

The dynamic exponent $\beta(\eta)$ is bounded, i.e., $0 < \beta(\eta) < 1$ because a node always increases the number of links in time (so, $\beta(\eta) > 0$) and $k_i(t)$ cannot increase faster than t so, $(\beta(\eta) < 1)$. Thus, $\beta(\eta)$ satisfies:

$$\beta(\eta) = \frac{\eta}{C}, C = \int \rho(\eta)\frac{\eta}{(1-\beta(\eta))}d\eta. \tag{1.12}$$

Equation (1.11) indicates that nodes with higher fitness increase their degree faster than those with lower fitness. Thus, the fitness model allows for late but fit nodes to take a more central role in the network topology.

Note that the fitness model developed by Bianconi and Barabasi has been extended by Ergun and Rodgers [59] to incorporate additional operations, such as addition of edges, which affect the exponents. The resulting model is referred to as the "additive-multiplicative" fitness model. The Bianconi Barabasi model belongs to the family of scale-free models for certain choices of the fitness distribution $\rho(\eta)$; the model is not robust against changes in the

functional form of the fitness distribution. It is a static model that creates undirected graphs and exhibits the following properties concerning the degree distribution:

- When the fitness parameters η_i are drawn randomly from a uniform $[0, 1]$ distribution, Equation (1.12) gives $C = 1.255$ and each node has a different dynamic exponent given by $\beta(\eta) = \eta/1.255$. In this case, the probability $p(k)$ that a node has degree k follows a power law of exponent equal to $C + 1 = 2.255$ with an inverse logarithmic correction, i.e., $p(k) \propto \int \frac{C}{\eta} \frac{1}{k^{1+C/\eta}} d\eta \sim \frac{k^{-(C+1)}}{\log k}$.

- For the case where all fitness values are the same, this model becomes the BA model.

- When the distribution $\rho(\eta)$ decays exponentially, the degree distribution $p(k)$ follows a stretched exponential behavior and $k(t)$ follows a complex combination of logarithmic and power law behavior. This indicates that $p(k)$ is not robust against changes in the functional form of fitness; certain choices of $\rho(\eta)$ can result in a non-power law distribution.

Non-Linear preferential attachment. Preferential attachment makes the assumption that the likelihood of receiving new edges increases with the node degree. For instance, the BA model assumes that the probability $\Pi(k_i)$ that a node attaches to node i is proportional to the degree k_i of node i. This assumption involves the hypothesis that $\Pi(k_i)$ depends linearly on k_i. The latter has been demonstrated incidentally by Krapivsky et al. [86] who replaced the linear dependence on k_i by a non-linear dependence of the form k_i^α. Depending on the value of the coefficient α, the authors have identified two distinct phases:

- Sublinear case $0 < \alpha < 1$: the obtained node degree distribution has a stretched exponential behavior. More precisely, the behavior describes the crossover from the BA model to the linking without preference ($\alpha = 0$) that produces exponential degree distributions.

- Superlinear case $\alpha > 1$: most of the connections come to the oldest vertex. Furthermore, for $\alpha > 2$, there is a finite probability that a given vertex is connected to all other vertices. In other terms, the "winner-takes-all" phenomenon emerges in this situation: almost all nodes have a single edge, connecting them to the oldest vertex.

The formal analysis of Krapivsky et al. [86] demonstrates that the scale-free nature of the network is destroyed for non-linear preferential attachment. The only case where the topology of the network is scale free is when the preferential attachment is asymptotically linear. This also means that non-linear attachment models are of little interest for the generation of Internet-like topologies. However, this does not mean that linear preferential attachment always provides scale-free networks (or, more precisely, networks with power law degree distributions).

1.2.3 Other models

In the following we present two models which do not introduce a radically new paradigm but rely on the previous ones.

The BRITE topology generator. BRITE is a parameterized topology generator which aims to model different aspects that lead to the power law behavior observed in Internet-like topologies such as preferential connectivity, incremental growth, node placement strategies, and connection locality. Nodes are distributed in a plane divided into $h \times h$ squares. Each one of these high-level squares is further subdivided into smaller $\ell \times \ell$ low-level squares. Each low-level square can be assigned at most one node. A random placement of nodes in the plane is achieved by selecting the low-level squares randomly and dropping a node in each of them so that collisions are avoided. To achieve a heavy-tailed distribution of nodes, for each one of the high-level squares, the generator picks a number of nodes n to be assigned to that square according to a bounded Pareto distribution given by:

$$f(n) = \frac{ak^{\alpha}n^{-\alpha-1}}{1-(\frac{k}{P})^{\alpha}}. \tag{1.13}$$

A node is then placed randomly in one of the $\ell \times \ell$ low-level squares. A parameter m controls the number of neighbor nodes to which a new node connects when it joins the network (or in other words, the number of new links to be added to the topology). The greater the value of m, the denser the generated topology is. A *candidate neighbor set* refers to the set of nodes from which a neighbor is selected for a newly considered node. A second parameter controls incremental growth:

- Nodes are all placed at once in the plane before adding any links. At each step, a node is randomly selected and links are used to connect it to m candidate neighbors from *all* other nodes.
- Nodes are placed in the plane gradually one at a time as they join the network. In this case, a new node considers as candidate neighbors only those nodes that have already joined the network. Initially, a small randomly connected backbone of m_0 nodes is generated. The remaining nodes are then connected.

The last parameter controls preferential connectivity and locality:

- For each newly added node, it selects a neighbor using Waxman's probability function. This process is repeated in order to determine all connections of the new node.
- By preferential connectivity, a newly considered node connects to a candidate neighbor node with probability $\frac{d_i}{\sum_{j \in C} dj}$, where d_i is the current outdegree of node i, and C is the set of candidate neighbor nodes. This process implies that a new node joining the network selects with high probability those nodes with high out-degree. This is repeated to connect v to m nodes.
- Both preferential connectivity and connection locality are considered. In this case, for a newly considered node, for each candidate neighbor node a Waxman's probability is computed. This gives preference to nearby nodes. Then, the final probability of connecting to node i is $\frac{w_i d_i}{\sum_{j \in C} w_j d_j}$. This process is repeated to connect v to m nodes.

According to the authors of [100], rank, out-degree, hotplot, and eigenvalue exponent in the corresponding distributions seem to fit well with the observed exponents of the Internet when topologies are generated with both preferential connectivity and locality. However, average path length approaches 12 for a topology of 16K nodes when diameter is about 30 hops and with a clustering coefficient close to 0.16.

Positive feedback preference model (PFP). The PFP model [138] is an evolution of the interactive growth (IG) model. In this model, nodes arrive in the graph and connect to one or more hosts where some of these hosts then develop new connections with other nodes of the network (e.g., in order to "better serve" the new node). The selection of hosts and of the nodes to which the hosts connect to is made according to a slightly superlinear preferential attachment rule. The IG model can be seen as a particular case of the PFP with simpler connection rules, and with a linear preferential attachment. The main advantage of the PFP model over the IG model seems to be the ability of also reproducing the maximal quantities.

In order to construct a graph, we start from a small random graph and then, at each iteration, we add a node i and select one of the three operations:

1. With probability p, we select two nodes j, k and connect (i, j) and (j, k). j is called the "host", and k is a "peer". The new node is connected to the host, which in turn is connected to a peer.

2. With probability $q \leq 1 - p$, we select three nodes j, k, l and connect (i, j), (j, k) and (j, l). The new node is connected to the host which in turn is connected to two peers.

3. Finally, with probability $1 - p - q$ we select three nodes j, k, l and connect (i, j), (i, k) and (j, l). The new node is connected to two hosts, one of which is connected to a peer.

In the above, whenever a node needs to be selected, the selection is made according to the following probability distribution:

$$\Pi(i) = \frac{k_i^{1+\delta \log_{10} k_i}}{\sum_j k_j^{1+\delta \log_{10} k_j}}. \tag{1.14}$$

Note that the interactive growth (IG) model is obtained as a particular case of this model by setting $p = 0$ and $\delta = 0$. Typical values of the parameters used to simulate the AS network are $p = 0.3, q = 0.1$, and $\delta = 0.048$. It must be noted that, although the model has only three parameters, several parameters are "hidden" in the definitions of the three cases (number of hosts, number of edges added, etc.).

The PFP model is a random graph model that creates undirected graphs with possible self-loops and multiedges. There does not seem to be any formal or approximate analysis of the properties of PFP networks. Based on simulations, the authors of [138] claim that, for $p = 0.3$ and $q = 0.1$, the PFP model accurately represents the following properties of the PFP graph: rich club connectivity, degree distribution, characteristic path length, average and maximal triange/quadrangle coefficients, average and maximal betweenness and average k_{nn}. The IG model, on the other hand, does not seem to accurately reproduce the maximal quantities (maximal degree, maximal triangle/quandrangle coefficient, maximal betweenness) and the average k_{nn}.

The Inet topology generator. The Inet model aims to produce random networks with characteristics similar to those of the Internet AS-level representation. Inet cannot produce networks with less than 3037 nodes, which was the number of ASs on the Internet in November 1997.

The Inet model is essentially based on three parameters: the total number of nodes in the topology, the fraction of degree-one nodes (the default is 0.3) and the size of the plane used

for node placement (the default is $10,000$). The *Inet-1.0* model [81] generates a topology by placing N nodes on an n by n plane. Each node is assigned an out-degree d at a given rank r where d grows exponentially over time according to:

$$d = e^{pt+q} r^R \tag{1.15}$$

where p, q, and R are known constants, and t is the number of months since November 1997.

Then a full mesh is used to connect the top τ most connected nodes. For these nodes, 25% of their edges are connected to randomly selected nodes with out-degree 2. To create a fully connected topology, the remaining nodes are either connected to one of these τ nodes or connected to a node that can reach one of these τ nodes. The Inet-1.0 model has a second phase where the top k most connected nodes are expanded into networks with n nodes each. This phase is used to expand the top most connected ASs into networks with router-level connectivity. In [81], they use $\tau = 5$ and omitted the second phase of the generation process since they were only interested in AS-level topology.

The *Inet-2.0* [81] follows the basic design of Inet-1.0, but uses a more systematic approach to generate the node out-degree distribution and to connect nodes in the topology. Let N be again the number of nodes and k be the fraction of nodes with out-degree 1. Assuming exponential growth rate of number of ASs, the number of months (t) it would take the Internet to grow from its size in November 1997 to N is first computed. With the computed t, the rank-out-degree distribution is then computed using Equation (1.15) and the frequency f_d of an out-degree d is computed according to:

$$f_d = e^{at+b} d^O \tag{1.16}$$

where a, b and O are known constants.

Equation (1.15) captures the out-degree distribution of only 98% of the nodes. Accordingly, the frequency-out-degree distribution Equation (1.16) is used to assign the out-degrees of the top 2% of the nodes. Next, a feasibility test is performed to ensure the connectivity of the network which is constructed as follows: first by forming a spanning tree using nodes with out-degree of at least 2, then by attaching nodes with degree 1 to the spanning tree, and finally by matching the remaining unfilled degrees of all nodes with each other. Then, a spanning tree among nodes with degree larger than 1 is computed. Let G be the graph to be generated; G is initially empty. Then, a node with out-degree larger than 1 that is not in G is connected to a node in G with a probability that depends on the out-degree of the node in G and the sum of out-degrees of all nodes already in G that still have at least one unfilled out-degree. Next, the kN nodes with out-degree 1 are connected with proportional probability. Finally, the remaining free out-degree nodes in G are connected starting from the node with the largest out-degree first. In making these connections, nodes with free out-degrees are randomly picked using proportional probability.

Inet-2.2 [82] extends Inet-2.0 in order to address the issue that in large networks, with more than $30,000$ nodes, not enough nodes with degree between 2 and 20 are generated. Consequently, Equation (1.16) is not directly applied to generate nodes with degree between 2 and 20. Instead, the number of nodes with degree d is computed by multiplying the fraction of the degree component d^O in the overall sum of 20 degree exponents by the number of nodes with degree between 2 and 20. For example, the number of nodes with degree 2 is

the following:

$$f_2 = \frac{2^0}{\sum_{d=1}^{20} d^O} N_{2-20}, \tag{1.17}$$

where N_{2-20} is the number of nodes with degrees between 2 and 20.

Inet-3.0 [136] was derived by analysing two properties of Inet-2.2: the complementary cumulative distribution function $\bar{F}(d) = \sum_{i>d}^{\infty} f(d)$ which is the fraction of nodes with degree greater than d, and the ratio of the minimum vertex cover of a graph to the size of the graph $\frac{|VC|}{N}$. Thus, a new exponential growth law was proposed:

$$\bar{F}(d) = e^c d^{at+b} \tag{1.18}$$

where c, a, and b are known constants and t is the number of months since November 1997. Equation (1.18) is used to calculate the degree distribution for all but degree 1 nodes and the top 3 nodes; the fraction of degree 1 nodes is left up to the user as a parameter. As the fraction of degree 1 ASs on the Internet has remained steady around 0.3, the default of this value is set to 0.3. The Outdegree-Rank exponential growth law defined in [81] is used to calculate the degrees of the top 3 nodes as it is done in Inet-2.2.

By studying the vertex cover (using an approximation) of the Internet topologies and of graphs generated by Inet-2.2, it has been shown that the vertex cover of Inet-2.2 topologies was about 50% larger than those of the Internet and this difference was increasing. This is due to the preference of low degree nodes to connect to other low degree nodes. Adding these small degree nodes increases the vertex cover as they do not cover many edges.

To address this issue, [136] weights the original linear preference function as follows. Let d_i and d_j be the degrees of nodes i and j respectively, and $f(d_i)$ and $f(d_j)$ be the frequency of those degrees. Then, w_i^j is the weighted value of d_j with respect to d_i and is defined as:

$$w_i^j = d_j \cdot \max\left(1, \sqrt{\left(\log \frac{d_i}{d_j}\right)^2 + \left(\log \frac{f(d_i)}{f(d_j)}\right)^2}\right). \tag{1.19}$$

Then, the probability that a node i with degree d_i connects to a node j of degree d_j is:

$$P(i,j) = \frac{w_i^j}{\sum_{k \in G} w_i^k}. \tag{1.20}$$

This modification has profound changes on the connectivity of nodes in Inet-3.0 compared to Inet-2.2 and matches quite closely the Internet topology.

1.3 Configuration models

One of the main approaches currently used for modeling complex networks consists in sampling a graph uniformly at random in a given class of graphs having some properties inspired from real-world observations (e.g., specific number of nodes, number of links, or degree distribution). This approach has the advantage that the obtained graphs may be considered as representative of the class, and that formal approaches are often possible (and rather general) for the study of such models [26, 110]. However, sampling such graphs is not trivial in general. This section presents the main models currently used for doing so.

Three main approaches coexist, each with its own advantages and drawbacks: The first one is to design an ad-hoc construction process and prove that it achieves uniform sampling. The second one is to start with a particular (not random) graph in the class and then shuffle it in order to make it random. Finally, one may use a generic construction process based on the construction of links from random pairs of connection points (often called half-links or stubs). None of these approaches succeeds in providing the best solution for all cases met in practice. For some sets of properties to capture, only a model based on the shuffling approach is available. For others, an *ad-hoc* construction process is known, which has the advantage of being much more efficient. As a consequence, various models exist based on each approach, and the choice depends on the properties to capture.

We detail the current state-of-the-art on these models below; for each model, we indicate the key properties it captures and discuss their strengths and weaknesses. The models are presented in order of increasing complexity (in terms of the property they capture), although this order may not be strict. We remark that in random graphs with usual parameters, the average distance (and diameter) is generally small and scales as the logarithm of the number of nodes n, or $\frac{\log n}{\log \log n}$. This is true for all models below.

The basic configuration model. The configuration model [4, 110] generates a random graph in the class of all graphs with a given degree sequence $n_k, k \geq 0$. Therefore the number of nodes is $n = \sum_k n_k$ and the number of links is $m = \sum_k k \cdot n_k$. This model works as follows: for each node, one chooses its degree according to the sequence; then, one creates as many connection points as its degree; and finally, one connects pairs of connection points chosen uniformly at random in the set of all connection points. See Figure 1.1.

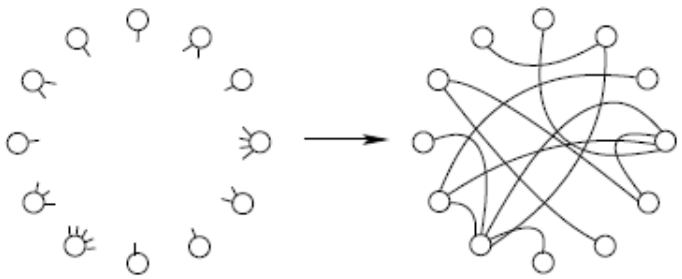


Figure 1.1: Construction of a random graph with prescribed degree sequences with the configuration model: first, nodes are drawn and each is assigned a degree with respect to the given sequence and as many connection points; then, links are built by choosing uniformly at random pairs of connection points.

This construction scheme may induce multiple links and loops. In practice, one generally simply removes them but this changes the degree distribution and induces some bias [25, 134, 33]. This is generally neglected. We discuss other solutions below (see the text related to SCCM). Notice that a degree sequence (a series of node degrees) is not a degree distribution. In practice, one often needs a random graph with prescribed degree *distribution* (not sequence). One then has to first sample a degree sequence from the degree distribution. In some cases, the obtained sequence is not realizable, meaning that there exists no graph with this degree sequence (for instance, if the sum of the obtained degrees is odd). One then

has to drop this degree sequence, sample another one and restart the process. We also remark that ER graphs can be obtained with the configuration model by using a Poisson prescribed degree distribution.

Correlated configuration model. In addition to their degree distribution, real-world network often exhibit nontrivial degree correlations [104, 116], which means that the probability for a node to be connected to another node depends on their degrees. This is formalized in practice by a joint degree sequence n_{ij} which gives the number of links between nodes of degree i and j.

The configuration model may easily be extended to produce random graphs with prescribed degree sequence and joint degree sequence [116, 33]: one creates for each node an appropriate number of connection points (as many as its degree) like in the basic configuration model; then for each i and j one samples n_{ij} random pairs of connection points attached to a node of degree i and one of degree j to form links.

Notice that, if one does not have a joint degree sequence but a joint degree distribution from which to sample a sequence, the obtained sequence may not be realizable. Dropping it as previously and restarting the construction process will eventually lead to a random graph as expected, but this may need many iterations. If this is a problem, one may use a shuffling approach like the one described in [104].

Simple connected configuration model (SCCM). The configuration model and its variants do not provide simple or connected graphs in general. This is a requirement in many practical cases though, but there is no known extension of this model which reaches this goal. As said above, one generally just removes loops and multiple edges, and even all components except the largest ones. This induces a bias which is generally neglected in practice.

One may however build a random simple connected graph with prescribed degree sequence using a shuffling approach. It consists in obtaining an initial simple connected graph with the required degree distribution and then iterating the following operation, called link swap: select two links (a, b) and (c, d) at random and exchange their endpoints, i.e., replace them by links (a, c) and (b, d). Note that edge swaps do not change the degree distribution (see Figure 2). One performs this link swap only if it does not disconnect the graph and creates no multiple link or loop. Eventually, this process leads to a random graph in the considered class [131, 71].

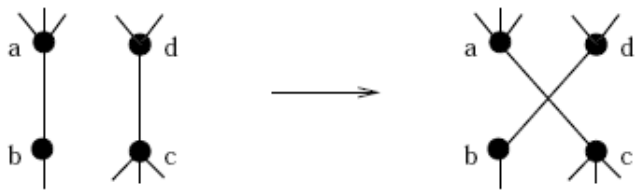


Figure 1.2: Link swap.

There are known procedures for producing the initial simple connected graph with prescribed degree sequence, if no additional information is provided [77, 78]. In practice, one

often has an initial graph obtained from real-world measurements and this model may be used to obtain a *randomized* version of it. The main issue of this approach is that, although it is known that the process will lead to a uniform random sampling if the number of link swaps is large enough [131, 134], there is no clear indication of the actual number of link swaps needed to be performed in practice. There are empirical ways to decide on this, in general by performing swaps until some properties of interest of the graph do not evolve anymore [71, 102]. One may also use this approach to generate (not necessarily connected) simple graphs with prescribed degree sequence, or other variants. The key point is to ensure that all graphs in the class are reachable by iterating link swaps, and a few easy probabilistic properties (e.g., see [75]).

Clustered configuration model (CCM). One of the key properties of complex networks is their clustering coefficient (and transitivity ratio) which is entirely determined by the degree of each node and the number of triangles containing the node.

One may extend the configuration model into a clustered configuration model [126, 107] by defining for each node v its degree $d(v)$ and the number $\Delta(v)$ of triangles containing it. One then defines the degree-triangle sequence of a graph as the series $n_{i,j}$ indicating that there are $n_{i,j}$ nodes with degree i contained in j triangles. Given such a degree-triangle sequence, one generates a random graph by choosing the degree and number of triangles of each node, then creating for each node the corresponding number of link connection points and triangle connection points, and then choosing random pairs of link connection points to create links and triples of triangle connection points to create triangles.

Bipartite model. A bipartite graph $B = (\top, \bot, E)$ is a graph in which nodes belong to two different classes $\top$ and $\bot$ and links exist only between nodes in different classes: $E \subseteq \top \times \bot$. Many real-world networks have a natural bipartite structure [91]. This includes the Internet, where routers are connected by swaps (direct wires between routers may be seen as degree two swaps). As a consequence, generating random bipartite graphs is interesting in many contexts.

The configuration model may easily be extended for doing so [75, 110]: given two degree sequences $\top_k$ and $\bot_k$, one chooses accordingly the degree of each node and then creates for each node as many connection points as its degree, and finally connects pairs of connection points on both sides chosen uniformly at random (see Figure 1.3). Notice that the same principle can be applied to produce multipartite or colored random graphs.

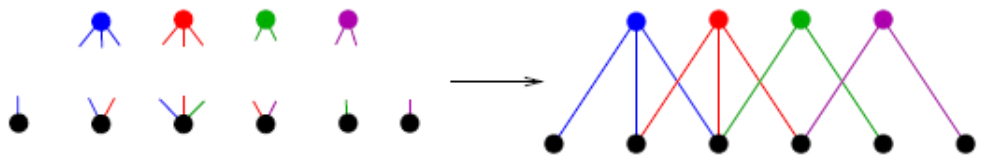


Figure 1.3: Construction of a random bipartite graph with prescribed degree sequences.

1.4 Matrix-based models

In this section, we present two models that are based on the recursive construction of the adjacency matrix.

1.4.1 Recursive matrix graph model

The Recursive Matrix graph model (R-MAT) [35] is a procedural generator which tries to find simple mechanisms to generate graphs that match the properties of real graphs. R-MAT generates the graph by operating on its adjacency matrix in a recursive manner. There are several design goals that R-MAT tries to achieve. The generated graph should match several graph patterns, power law degree distributions, hop-plots and eigenvalue plots, etc. It should also be able to generate graphs exhibiting deviations from power laws as observed in some real-world graphs. Furthermore, graphs should preferably exhibit a strong community effect and should be able to generate directed, undirected, or weighted graphs with the same methodology. Moreover, the number of design parameters should be as small as possible and a fast parameter-fitting algorithm is desirable. Finally, the generation algorithm should be efficient and scalable.

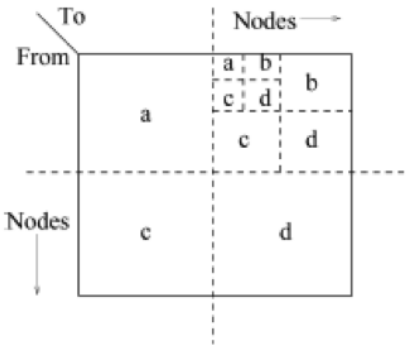


Figure 1.4: Partitioning of the adjacency matrix in R-MAT.

The basic R-MAT model is able to generate a directed graph of predefined number of nodes and edges. The user has to define the number of nodes and edges in the graph. Therefore, R-MAT creates directed graphs with 2^n nodes and $|E|$ edges where both values are provided by the user. A graph is created in the following way: it begins with an empty adjacency matrix and divides it into four equal-sized partitions. Each of the four partitions is chosen with probabilities a, b, c, d, respectively (so that $a+b+c+d = 1$). Then, the chosen partition is again subdivided into four smaller partitions and this procedure is repeated until it reaches a simple cell (a 1×1 partition). The nodes (i.e., row and column) corresponding to this cell are linked by an edge in the graph. This process is repeated $|E|$ times to generate the full graph.

The R-MAT generator uses only 3 parameters and it has been experimentally demonstrated in [35] that the resulting graphs have power law distribution of nodes, small diameter and also match several other properties. The basic R-MAT model generates directed graphs but it is possible to generate all the other types of graphs by slightly modifying it. Undirected graphs can be generated by making all the edges in the basic directed graph symmetric, while

for weighted graphs, the number of duplicate edges in each cell of the adjacency matrix plays the role of the weight of that edge.

On the positive side, the authors of [35] have run experiments on large graphs with $N = 75,879$ and $|E| = 508,960$ and have verified that R-MAT has a low graph generation time compared to other generators; furthermore, it seems possible to use R-MAT to generate Internet-like graphs. On the negative side, even though the R-MAT model shows promising results on experiments, there has not been any thorough analytical study of this model. Moreover, it might not provide enough degrees of freedom to match all varieties of graphs as it has only 3 parameters, whereas no algorithm is provided in [35] to determine the values of these parameters that produce realistic networks. Finally, R-MAT does not have adaptability since the number of nodes in the network has to be fixed at design phase. If we want to add a new node to the graph, the whole process has to be repeated.

1.4.2 Kronecker graph model

The Kronecker graph model [93] is based on a recursive construction using a matrix operation called the Kronecker product. The main intention behind the model is to create self-similar graphs, recursively.

Before introducing the definition of the Kronecker graph, we remind the definition of the Kronecker product. Given two matrices $A = [a_{i,j}]$ and B of sizes $a \times b$ and $n \times m$ respectively, the Kronecker product matrix C of dimensions $(a \cdot n) \times (b \cdot m)$ is given by

$$C = A \otimes B = \begin{pmatrix} a_{1,1}B & a_{1,2}B & \dots & a_{1,m}B \\ a_{2,1}B & a_{2,2}B & \dots & a_{2,m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}B & a_{n,2}B & \dots & a_{n,m}B \end{pmatrix} \tag{1.21}$$

A Kronecker graph has an adjacency matrix which is a result of the Kronecker product of the adjacency matrices of two or more graphs. In this particular graph model, an initial matrix K_1 is defined. The elements in the initial matrix are the design parameters. Then, Kronecker product will be repeatedly applied to this initial matrix, until the size of the resulting matrix reaches the number of nodes in the network.

The design parameter of the Kronecker graphs is the initial adjacency matrix K_1. Using this initial adjacency matrix, we can generate a graph with any number of nodes and edges using the Kronecker product. Kronecker graphs exhibit multinomial degree distributions, both for in-degrees and out-degrees. Moreover, they usually follow the densification power law (DPL) with densification exponent $a = \log E_1 / \log N_1$ where E_1 and N_1 are the number of edges and nodes in the initial adjacency matrix. If K_1 has diameter D and has a self-loop on every node, then for every product k, the graph K_k also has diameter D.

Stochastic Kronecker graphs. The basic Kronecker graphs lead to the staircase effect due to self-similarity. This, however, is not the case in real-world graphs, hence, the stochastic Kronecker model has been proposed in order to remove undesired staircase effects [93].

There are two common methods to overcome this staircase effect. The first is to uniformly at random flip some edges, i.e., uniformly at random select entries of the graph adjacency matrix and flip them ($1 \to 0$, $0 \to 1$). However, this will corrupt the degree distribution of the nodes. So, the second way is to allow a weighted initiator matrix, i.e., values of entries

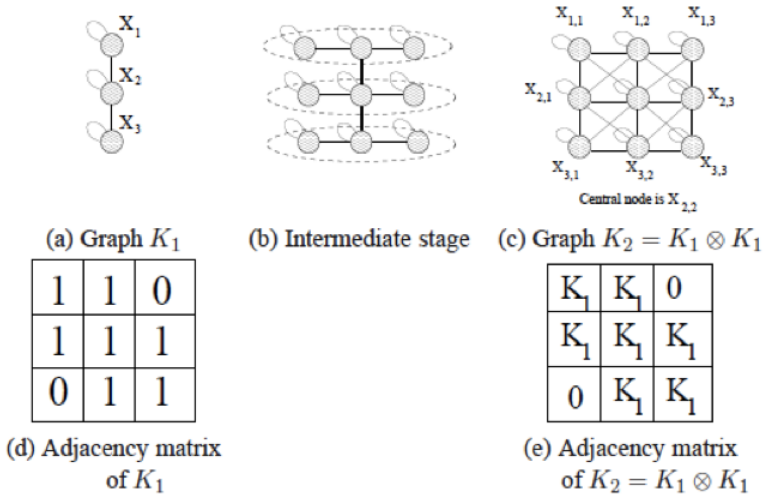


Figure 1.5: Example of Kronecker graph generation.

of K_1 are not restricted to values $0, 1$ but rather can be any nonnegative real number. This would not work, as the mechanism would selectively remove edges and thus the low degree nodes, which would have low weight edges, would get isolated first.

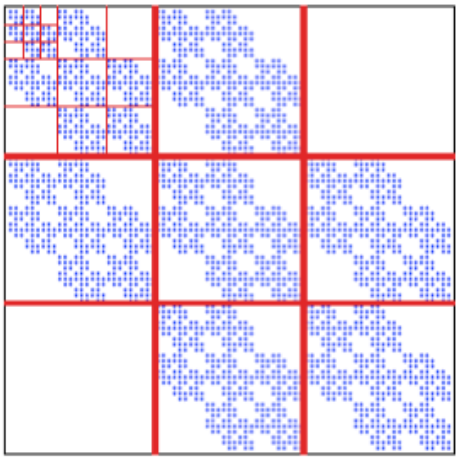


Figure 1.6: Example of staircase effects.

None of these methods can be used when we wish to generate real-world graphs. Therefore, in [93] a new methodology was used in the proposed model to tackle these problems. The authors allowed entries of the initiator to take values on the interval $[0, 1]$ by relaxing the assumption that an entry of the initiator matrix takes only binary values. Intuitively, this means that each entry of the initiator matrix encodes the probability of that particular edge appearing.

The intuition for fast generation of Stochastic Kronecker graphs comes from the recursive nature of the Kronecker product. Generating a Stochastic Kronecker graph K on N nodes naively takes time $O(N^2)$ but a linear time algorithm in the number $|E|$ of the (expected)

number of edges of K has been presented in the same work.

Kronecker graphs can naturally model bipartite graphs. Instead of starting with a square $N_1 \times N_1$ initiator matrix, one can choose arbitrary $N_1 \times M_1$ initiator matrix, where rows define the left, and columns the right side of the bipartite graph. Kronecker multiplication will then generate bipartite graphs with partition sizes N_1^k and M_1^k. With respect to graph distributions, P_k defines a distribution over all graphs, as it encodes the probability of all possible N_1^{2k} edges appearing in a graph by using an exponentially smaller number of parameters (just N_1^2). Even a very small number of parameters, e.g., 4 (2×2 initiator matrix) or 9 (3×3 initiator), is enough to accurately model the structure of large networks.

Stochastic Kronecker graphs form an extension of Erdős-Rényi random graph model. For example, if one takes $P_1 = [\theta_{ij}]$, where $\theta_{ij} = p$, we obtain exactly the Erdős-Rényi $G(n, p)$ model. The recursive nature of Stochastic Kronecker graphs makes them also related to the R-MAT generator [35]. The difference between the two models is that in R-MAT one needs to separately specify the number of edges, while in Stochastic Kronecker graphs initiator matrix P_1 also encodes the number of edges in the graph. Similar to deterministic Kronecker graphs the number of nodes in a Stochastic Kronecker graph grows as N_1^k and the expected number of edges grows as $(\sum_{ij} \theta_{ij})^k$. This means one would want to choose values θ_{ij} of the initiator matrix P_1 so that $\sum_{ij} \theta_{ij} > N_1$ in order for the resulting network to densify.

The authors of [93] have done lot of simulations to validate their model. They used the citation graph for High-Energy Physics Theory research papers from pre-print archive ArXiv, with a total of $N = 29,555$ papers and $|E| = 352,807$ citations for their simulation. They generated the Kronecker graph with K_1 being a star graph on 4 nodes, and the Stochastic Kronecker graph ($\alpha = 0.41$, $\beta = 0.11$ – bottom row). The simulation results are shown below.

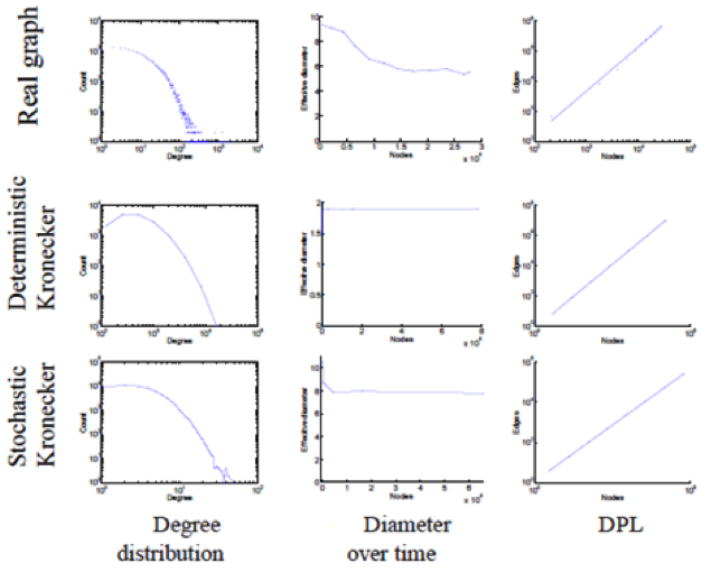


Figure 1.7: Simulation results on citation networks.

It can be seen that the deterministic Kronecker model already matches the qualitative structure of the degree distributions of the real network, as well as the temporal patterns represented by the densification power law and the stabilizing diameter. However, the deterministic nature of this model results in staircase behavior. Meanwhile, the Stochastic

Kronecker graphs smooth out these distributions, further matching the qualitative structure of the real data, and they also match the shrinking-before-stabilization trend of the diameters of real graphs. Notice that the Stochastic Kronecker graphs qualitatively match all the patterns very well.

The authors of [93] show that Kronecker graphs have several static properties (heavy-tailed degree distribution, small diameter, etc.) and several temporal properties (densification, shrinking diameter) that are exhibited by real networks. They also formally prove all of the aforementioned properties; most of these proofs are not technically involved. With respect to parameter setting, Leskovec et al. [93] also proposed algorithm KRONFIT, a fast, scalable algorithm to estimate Stochastic Kronecker initiator, which can then be used to create a synthetic graph that exhibits the properties of a given real network. Experiments on very large graphs with $N = 29,555$ and $|E| = 352,807$ using KRONFIT algorithm have verified that Kronecker graph generator has a low graph generation time compared to other generators. Furthermore, the Kronecker graph provides enough degrees of freedom to match all varieties of graphs as we can change the initial matrix; this is an advantage compared to R-MAT. Also, it is enough to use 2×2 or 3×3 initiators in order to fit the properties of real-world networks well. Also, contrary to the R-MAT generator, the Kronecker graph model is adaptable as we do not need to define the number of nodes in the network at the design phase. If at some point we wish to add a new node to the graph, we can run the Kronecker product on the existing matrix. Therefore, we can start the generation from the very last phase.

1.5 Special purpose models

We complete the overview of graph/network models by discussing two models that have been proposed to model the Internet topology taking into account low-level (i.e., technology-specific) characteristics.

IGen. IGen generates router-level topologies of the Internet [117]. The methodology for graph generation consists of six main steps (see Figure 1.8). For each step, the user of IGen can specify her own design goals.

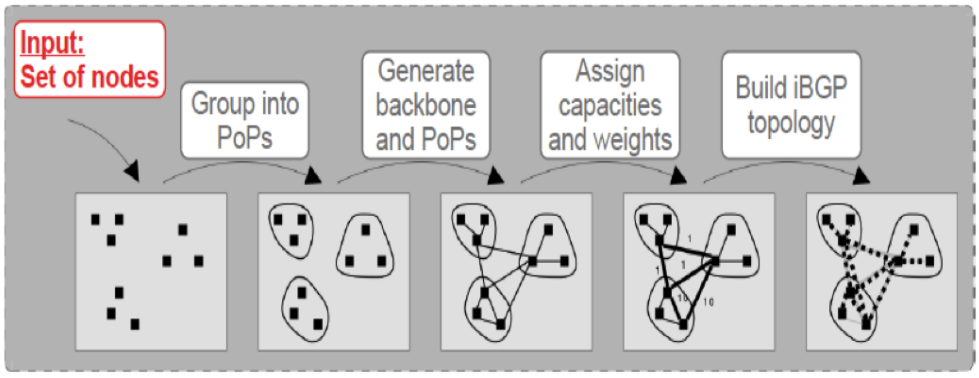


Figure 1.8: 6 main steps of graph generation with IGen.

1. Placing routers: The geographical location of nodes is specified.

2. Identification of PoPs: The nodes are grouped into PoPs, based on their geographical location. Within each PoP (which can be thought of as an AS of the Internet), nodes that will be part of the backbone are identified.

3. Building the topologies of PoPs: Within each PoP, the n most (geographically) central nodes are selected as backbone routers. These nodes are densely connected together using for instance a tour that guarantees 2-edge-connectivity or a clique. Then, the remaining nodes of the PoP, which model access nodes, are connected to the PoP's backbone nodes using at least k edges (see Figure 1.9).

4. Building the backbone topology: The current version of IGen uses several such heuristics in order to build the backbone.

5. Link capacities and path selection: In an optional step, capacities and IGP weights can be assigned to links. IGP weights will influence the selection of intra-domain paths by an IGP.

6. iBGP topology design: Finally, the network topology can be overlayed with a graph of iBGP sessions. Different iBGP graphs are possible. IGen can automatically generate the topology of iBGP sessions within the network. This logical topology can strongly differ from the physical topology.

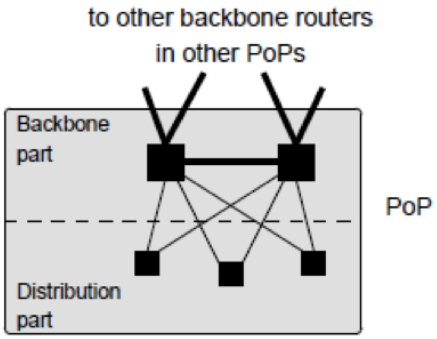


Figure 1.9: Connectivity within a PoP.

IGen generator captures different aspects of the topology of the Internet such as the redundancy of peering links, peering policies, geographic spread of routers, and router-level domain topologies. Therefore, this topology is more detailed than topologies obtained from probabilistic models. Four different network design heuristics have been proposed. They can be used to address different features of the network. For instance, the hybrid MST-SPT heuristic is suitable to build low-cost network but without robustness. On the other hand, a Delaunay triangulation produces a network that is resilient to the failure of links and nodes. Therefore, IGen can be used to generate networks with different aspects. The authors of [117] did not discuss about satisfying any properties of the Internet such as power low exponent, clustering coefficient and diameter. Hence, determining whether these are satisfied is still an open question.

GT_ITM. This model aims to produce random hierarchical graphs using three levels of hierarchy: transit domains, stub domains, and LANs attached to stub nodes. It builds the graph piecewise, where the pieces correspond to domains at the different levels. The connectivity within a domain is dealt with separately from that between different domains.

The network properties are governed by two sets of parameters. The first set defines the relative size of the three levels in the hierarchy while the second one affects the connectivity within a domain and the connectivity between domains at the same or higher and lower levels. The generation process begins at the top (transit domain) level of the hierarchy, and proceeds down to the lowest (LAN) level. Nodes within a level are generated in a rectangular sub-region of the overall space occupied by the graph. The scale of the sub-region in which the nodes are distributed is changed for each type of network, with LANs having the smallest scale. In many of the steps, edges must be generated so that a sub-graph is connected and a specified edge number is reached. Two implementations are available, namely the Transit-Stub (TS) and the Tiers model.

The main steps for graph generation are as follows:

1. A location in the plane is chosen for each transit domain from the overall space occupied by the graph. Edges are generated between transit domains so that the domains are connected according to the second set of parameters.

2. For each transit domain, nodes are placed in a sub-region centered around the transit domain location so that transit domains contain the pre-specified number of nodes. The specified number of edges are generated within each transit domain.

3. Particular transit nodes from Step 2 are selected as the endpoints of the inter-transit domain edges from Step 1.

4. The locations for the stub domains are chosen. For each stub domain, nodes are placed in a sub-region centered around the chosen location. Edges are generated within each stub domain so that the connectivity requirements defined by the second set of parameters are met.

5. Each stub domain is connected to a transit node by an edge, connected to a particular stub node and a particular transit node. Additional edges can be added from stub domains to transit nodes.

6. The locations for the LANs are chosen. For each LAN, nodes representing host systems are placed in a star around a center router.

7. Each LAN is connected to one stub node by an edge from the center router to the stub node. Additional edges can be added from center routers to stub nodes.

In the Transit-Stub model, representation of hosts systems is not supported; hence, all nodes are supposed to be routers. The model produces connected stub-graphs by repeatedly generating a graph according to the edge number, and checking the graph for connectivity. Extra edges from stub domains to transit nodes are added by random selection of the domains nodes. The model also supports an additional parameter that indicates the number of stub-to-stub edges. These edges are also added by random selection of the domains and nodes involved. Nodes are associated to several different types of information such as transit or

stub node, domain to which each node belongs to, domain local identifier, and primary transit node where a stub node is attached. Routing policy weights are associated with the edges.

In the Tiers model, three levels of hierarchy (WAN, MAN, LAN) correspond to transit domains, stub domains and LANs of the basic model. The Tiers model produces connected sub-graphs by joining all the nodes in a single domain using a minimum spanning tree. The use of a minimum spanning tree is a crucial feature of Tiers and is particulary appropriate since it is sometimes used in reality as the basis for laying out large networks. For redundancy reasons (within a domain), edges are added to the closest nodes in the network, in increasing order of Euclidean distance. For inter-domain redundancy, extra edges are added to the closest nodes in the next higher domain. Edge weights reflect the processing delay and bandwidth constraints.

Chapter 2

Graph properties

In this chapter, we present an overview of identified graph properties that are inherent in large-scale networks and are likely to affect routing performance, as well as algorithms for testing them. We survey various graph invariants which are proven effective either for the design of compact routing schemes, or for distance decreasing routing schemes (greedy routing). For all of them, we present various definitions that might be equivalent or are alternative definitions studied so far. The complexity of computing these invariants is discussed with respect to how large-scale networks are known to reflect these properties. Finally, we give some hints on the impact of these parameters on routing schemes.

Mainly, the properties we consider can be distinguished according to several criteria. Some of them can be recognized in polynomial time; in this case, it will be interesting to know when such recognition algorithms are practical in huge networks. For instance, the hyperbolicity of a graph (see Section 2.15) can be determined in time $O(N^4)$ which is almost unreachable for graphs with more than $N = 10,000$ vertices. On the other hand, some properties corresponding to well known NP-complete decision problems lead to very efficient routing schemes in specific graph classes. Also, some properties are emphasized for their interest in routing, while others are considered because they provide an efficient way to characterize large scale networks.

We begin the discussion on graph properties by considering the degree distribution first. Then, we describe properties that express how nodes are linked together. The related parameters can be determined in polynomial time but the most interesting ones such as the length of paths, the clustering coefficient, betweenness centrality, and hyperbolicity require more than a quadratic computational time in terms of the network size. Such properties have been widely studied in the literature and give an interesting and efficient way to characterize real networks (e.g., power law degree distribution, etc.). They will be used in the experimentation phase of the project to generate models and graphs reflecting reality. On the other hand, the structure of the shortest paths will be used in the design of efficient routing schemes. For example, properties such as chordality, growth, and doubling dimension facilitate compact routing algorithms; such issues are discussed in the corresponding sections below.

2.1 Node degree distribution

Definitions. Let $G = (V, E)$ be a graph with vertex set V such that $|V| = n$ and edge set E such that $|E| = m$. The degree of node u is defined as the number of edges incident to u,

i.e., $k_u = |\{e \in E | u \in e\}|$. The average node degree $\bar{k}$ is defined as $\bar{k} = \frac{2m}{n}$. Let $n(k)$ denote the number of nodes of degree k. The node degree distribution $P(k)$ is the fraction of nodes of degree k: $P(k) = \frac{n(k)}{n}$. Sometimes, $P(k)$ is considered to be the number (instead of the fraction) of nodes of degree k. One may also consider the cumulative node degree distribution (fraction of nodes with degree at most k) and the inverse cumulative degree distribution (at least k). The inverse cumulative distribution may be much more suitable for visual inspection, in particular when the distribution is heterogeneous (e.g., power law). A related notion is the *rank distribution*. The rank of a vertex is its index in an ordering of the vertices by decreasing degrees. The information of the degree of a vertex as a function of its rank provides extra information for visual inspection.

Complexity, recognition and generation. Network topologies with higher average node degree $\bar{k}$ are "better-connected" and, consequently, are likely to be more robust than topologies with lower average node degree. However, a detailed topology characterization based only on the average degree is limited as graphs with the same average node degree can have very different topological structure.

Computing the degree distribution of a graph is in general trivial. However, automatically deciding on its type is a subtle statistical problem (involving fit of empirical data with models, fit assessment, etc.). Real-world graphs are generally supposed to have heterogeneous (power law) degree distributions, while the ER basic model generates graphs with homogeneous (Poisson) degree distribution. There is a controversy regarding the degree distribution of Internet topologies due to measurement bias.

Faloutsos et al. [60] have identified power laws in the degree distribution of both router- and AS-level graphs. Formally, a random variable X or its corresponding cumulative distribution function $F(x)$ ($F(x) = P[X \leq x]$, $x \geq 0$) is said to follow a power law distribution with scaling index (or tail index) $\gamma > 0$ if $P[X > x] = 1 - F(x) \approx cx^{-\gamma}$ as $x \to \infty$.

Power law distributions are referred to as scaling distributions because if the random variable X follows a power law distribution, then the conditional distribution of X given that $X > w$ is given by $P[X > x | X > w] = \frac{P[X>x]}{P[X>w]} \approx c_1 x^{-\gamma}$ where constant c_1 is independent of x and given by $c_1 = w^{\gamma}$. Thus, (at least) for large values of x, $P[X > x | X > w]$ is identical to the (unconditional) distribution $P[X > x]$, except for a change in scale.

The ER model generates graphs with Poisson degree distributions, while the basic configuration model and its variant SCCM generate graphs with prescribed degree distribution (including Poisson or power law). Also, the BA model (and its variants such as GLP) generates graphs with power law degree distribution using preferential attachment (see Section 1.2.2).

2.2 Joint node degree distribution

Definitions. Let $m(k_1, k_2)$ denote the total number of edges connecting nodes of degrees k_1 and k_2, and m the total number of edges. The joint degree distribution (JDD) also referred to as the node degree correlation matrix is the fraction of edges that connect k_1- and k_2-degree nodes: $P(k_1, k_2) = \mu(k_1, k_2)\frac{m(k_1,k_2)}{2m}$, where $\mu(k_1, k_2) = 1$ if $k_1 = k_2$ and $\mu(k_1, k_2) = 2$ otherwise.

Complexity, recognition and generation. The JDD contains more information about the connectivity of a graph than the node degree distribution. Indeed, one can retrieve the node degree distribution $P(k)$ and average node degree $\bar{k}$ by observing that $\sum_{k'} P(k,k') = \frac{kP(k)}{\bar{k}}$, where $P(k,k')$ is the probability that a randomly selected link connects a node of degree k with a node of degree k'.

Algorithmic implications. The node degree distribution $P(k)$ determines the fraction or number of nodes of a given degree in the network. However, the node degree distribution does not provide information on the interconnection between these nodes: given $P(k)$, the usual structure of the neighborhood of a node of a given degree does not follow some specific pattern. The joint degree distribution provides the information about 1-hop neighborhoods around a node.

2.3 Connected components

Definitions. A graph $G = (V, E)$ is *connected* if there is a path in G between every pairs of vertices. A graph is k-vertex-connected, or simply k-connected, if there is a set $S \subset V$ of k vertices such that $G \setminus S$ is not connected, and for any subset $S' \subset V$ of cardinality at most $k-1$, the graph $G \setminus S'$ is connected. We define similarly k-edge-connected graphs.

Complexity, recognition and generation. Testing if a graph is connected can be done using a centralized algorithm (for instance, using a depth-first-search algorithm) in linear time. The problem of determining the k-connectivity of a graph can be solved in polynomial time using the maximum flow-minimum cut duality theorem. Furthermore, the problem of partitioning a connected graph into 2-connected components can also be solved in linear time [130].

Algorithmic implications. Connectivity is the minimum requirement for a network topology in order to communicate and the more k-connected the network topology is, the more resilient to failures the network is. More precisely, if the graph is 2-connected, it will survive any single failure. However, in practice a 2-connected graph is more reliable and survives more than one failure, since the probability that failures occur simultaneously on two vertices forming a cut of the graph is low.

2.4 Path length and distance

When studying routing in networks, the first questions that come in mind are about the length of a shortest path from node u to node v, and about the diameter of the graph. The literature on how to compute a shortest path, a single source shortest path tree, or the set of shortest paths between all pairs is enormous and we refer to [45] for a state-of-the-art survey.

In the following, we discuss some of the graph properties derived from the knowledge of the shortest path tree from a particular node, or of the all pairs shortest paths.

Definitions. Let $G = (V, E)$ be a connected graph, and let $d(u, v)$ be the shortest path length from node u to node v in G. We have:

Eccentricity. The eccentricity $\epsilon(u)$ of node $u \in V$ is the maximum graph distance between u and any other vertex $v \in V$:

$$\epsilon(u) = \max_{v \in V \setminus \{u\}} d(u, v). \tag{2.1}$$

Diameter. From the definition of the eccentricity, we can define the diameter of a graph as the maximum graph eccentricity of its vertices,

$$D_G = \max_{u \in V} \epsilon(u). \tag{2.2}$$

Radius. We define the radius of a graph as the minimum graph eccentricity of its vertices,

$$R_G = \min_{u \in V} \epsilon(u). \tag{2.3}$$

Mean eccentricity of a graph. The mean eccentricity $\bar{\epsilon}_G$ of a graph has been defined in [17] as the average value of the eccentricity of its vertices. So we have:

$$\bar{\epsilon}_G = \frac{1}{|V| - 1} \sum_{v \in V} \epsilon(v). \tag{2.4}$$

Average shortest path length from a vertex. The average shortest path length of a graph from a vertex $u \in V$, denoted $\bar{d}(u)$, is the average of the distances from u to any other vertices of the graph. So we have:

$$\bar{d}(u) = \frac{1}{|V| - 1} \sum_{v \in V \setminus \{u\}} d(u, v). \tag{2.5}$$

Average shortest path length. The average shortest path length of a graph, denoted $\bar{d}_G$, is also know as the *Wiener index* of the graph. It is the average of the distances between all the vertex pairs of the graph. So we have:

$$\bar{d}_G = \frac{1}{|V|(|V| - 1)} \sum_{u \in V} \sum_{v \in V \setminus \{u\}} d(u, v) = \frac{1}{|V|} \sum_{u \in V} \bar{d}(u). \tag{2.6}$$

Distance distribution. The distance distribution $d(x)$ or simply the shortest path length distribution is defined as the probability that a random pair of vertices (u, v) are at a distance x edges from each other. Closely related are the average distance d_m and the standard deviation σ, also referred to as the distance distribution width since distance distributions in Internet graphs have a characteristic Gaussian-like shape.

Note that the normalized version of distance distribution $d(x)$, that is the distance distribution divided by the total number of pairs n^2 (self-pairs included), is a critical metric for topology comparison analysis.

Hotplot distribution. The hotplot distribution $hp(d)$ is defined as the probability that a random pair of vertices (u, v) are at a distance *strictly less* than d edges from each other.

From the above definitions, we have:

$$R_G \leq \epsilon(u) \leq D_G \qquad \forall u \in V \tag{2.7a}$$
$$R_G \leq \bar{\epsilon}_G \leq D_G \tag{2.7b}$$
$$\bar{d}(u) \leq \epsilon(u) \leq D_G \qquad \forall u \in V \tag{2.7c}$$
$$\bar{d}_G \leq \bar{\epsilon}_G \leq D_G \tag{2.7d}$$

It is however not possible to compare the radius of the graph and the average shortest path length of a vertex or a graph, since these values can be either lower or higher than the radius, depending on the general expansion of the graph.

Other related notions can be found in the literature. However, some of them are only new names for already known notions, and some others have multiple definitions. For instance, the *characteristic path length* is defined in [95] to be the average shortest path length ($\bar{d}_G$) of a graph, but [32] proposed an alternative definition: the characteristic path length of a graph is the median of the means of the shortest path lengths connecting each vertex $v \in V$ to all other vertices, that is the median of $\bar{d}(\cdot)$ for all network nodes.

Complexity, recognition and generation. All these notions can be computed in polynomial time and are dominated by the computation of all pairs shortest paths. However, for very large graphs, the computation time could become prohibitive. Therefore, several authors have proposed approximation algorithms for computing some of these values. In particular, approximation algorithms as well as methods computing lower and upper bounds on the diameter of a graph have been proposed in [96]. These methods are observed to perform very well in practice. In addition, [39] proposes a linear time algorithm for approximating the diameter of δ-hyperbolic graphs with an additive error of 2δ, and the radius of δ-hyperbolic graphs with an additive error of 3δ. A significant research effort has been made to obtain close formulas for some graph classes. See e.g., [17] for the mean eccentricity of the de Bruijn graph, and [123] for bounds on the eccentricity of de Bruijn and Kautz digraphs.

Algorithmic implications. Random graphs typically have small diameter, usually $\log |V|$, and CAIDA AS maps have even smaller diameters (diameter 10 for graphs from $20,000$ to $36,000$ nodes). Furthermore, adding a few links to a graph reduces its diameter, and alternatively, removing some links may increase it. However, the mean eccentricity of the graph is in general more stable to small topological changes. In fact, the diameter is a worst case for distance, and the average distance and/or the effective diameter (the smallest value e such that at least 90% of pairs of nodes are at a distance at most e) provide more information on the length of the routes, on the robustness of the network against topological changes, and, therefore, on its structure.

For instance, short average distance and narrow distance distribution width break the efficiency of traditional hierarchical routing; these are the main reasons of inter-domain routing scalability issues in the Internet. Also, distance distribution plays a vital role in robustness of the network to worms. Indeed, worms can quickly contaminate a network that has small distances between nodes. In this respect, topology models that accurately reproduce observed distance distributions will lead to the development of techniques to quarantine the network from worms.

2.5 Clustering coefficient

Informally, the clustering coefficient measures the clumpiness of a graph, and has relatively high values in many graphs. In [125] it is also referred as the "network density". Essentially, there are two variants of this parameter; the local and the global clustering coefficient.

Local clustering coefficient. As for many graph properties, different notation and terminology is used in the literature to express exactly the same notion. Indeed, the local clustering coefficient $CC_G(u)$ of an undirected graph $G = (V, E)$ characterizes the density of connections in the environment of a node $u \in V$ [99, 135].

The clustering coefficient of a node is commonly defined as the ratio of the number of edges connecting the d_u neighbors of $u \in V$ (d_u is the degree of node u), over the maximum possible $\binom{d_u}{2} = d_u(d_u - 1)/2$ (number of edges of a clique of order d_u). Let $\Gamma(u)$ be the set of neighbors of u with $|\Gamma(u)| = d_u$, and let $E(\Gamma(u))$ be the set of edges between vertices in $\Gamma(u)$. We have,

$$CC_G(u) = \begin{cases} \frac{2 \cdot |E(\Gamma(u))|}{d_u(d_u-1)} & \text{when } d_u \geq 2, \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

An alternative definition due to [105] uses the notion of "triangle", i.e., a set of three vertices each of which is connected to the others and more precisely a clique of order three. A "connected triple centered on vertex u" means a central node u connected to the other two; the flanking nodes are unordered. We have,

$$CC_G(u) = \begin{cases} \frac{\text{number of triangles connected to vertex } u}{\text{number of connected triples centered on vertex } u} & \text{when } d_u \geq 2, \\ 0 & \text{otherwise.} \end{cases} \tag{2.9}$$

These two definitions are essentially the same since there is a triangle connected to vertex u, say (u, a, b), if and only if a and b are neighbors of u and there is an edge between them (i.e., $a, b \in \Gamma(u)$ and $(a, b) \in E(\Gamma(u))$).

Global clustering coefficient. Concerning the global clustering coefficient C_G of a graph G, there is no common agreement; so, there are mainly three definitions.

- In [105], the global clustering coefficient is called *transitivity* and occurs if and only if triangles exist in the graph G. This can be used to measure the transitivity ratio $tr(G)$ of a graph G as:

$$tr(G) = 3 \times \frac{\text{number of triangles in G}}{\text{number of connected triples in G}}. \tag{2.10}$$

  The equation counts the fraction of connected triples which actually form triangles; the factor of three is due to the fact that each triangle corresponds to three triples. Like the local clustering coefficient, this parameter aims to capture a notion of local density in the graph. It is expected to be much higher than the graph density in real-world graphs.

  Another equivalent formulation, in which a path of length two refers to a directed path with two edges and starting from a specified vertex, is:

$$tr(G) = 6 \times \frac{\text{number of triangles in the graph}}{\text{number of paths of length two}}. \tag{2.11}$$

This definition shows that $tr(G)$ is also the mean probability that the friend of your friend is also your friend.

We remark that there is often confusion between the clustering coefficient and the transitivity ratio, which are however different.

- A second definition reverses the order of the operations of taking the ratio of triangles to triples and of averaging over vertices. It tends to weight the contributions of low-degree vertices more heavily, because such vertices have a small denominator in local clustering coefficient. It leads to very high variance in the clustering coefficients of low-degree nodes (e.g., a degree 2 node can only have $CC_G(u) = 0$ or 1).

- Alternatively, in [135] and [99], the global clustering coefficient of a graph is defined as the average of the local clustering coefficient of its vertices. So, we have,

$$CC_G = \frac{1}{N} \sum_{u \in V} CC_G(u). \tag{2.12}$$

As the local clustering coefficient is not defined for nodes of degree zero or one, some authors compute the average of the local clustering coefficient of nodes of degree at least two. Let $V_{>1} \subset V$ be the set of vertices of degree more than one. We have,

$$CC'_G = \frac{\sum_{v \in V_{>1}} CC_G(v)}{|V_{>1}|}. \tag{2.13}$$

The results given by these definitions of the global clustering coefficient can actually be quite different. Intuitively, a high clustering coefficient should imply high transitivity ratio and vice versa. However, one may build families of graphs where one parameter tends to 0 and the other to 1.

Complexity, recognition and generation. The computation of the transitivity ratio (respectively, the local clustering coefficient) requires efficient ways to compute the number of triangles in the graph (respectively, connected to vertex u). An algorithmic survey is available for exact in-memory computation while online and approximate algorithms also exist [90]. In [8], the authors describe a deterministic algorithm for counting the number of triangles in a graph. Their method takes $O(N^\omega)$ time, where $\omega < 2.376$ is the exponent of matrix multiplication [45]. However, this is more than quadratic in the number of nodes and might be impractical for very large graphs. Also, an algorithm for counting triangles when the graph is in streaming format is proposed in [12]. That is, the data is a stream of edges which can be read only sequentially and only once. The advantage of streaming algorithms is that they require only one pass over the data, and so they are very fast; however, they typically require some temporary storage, and the aim of such algorithms is to minimize this space requirement. The algorithm proposed in [12] is a $O(\log N)$-space randomized algorithm for the case when the edges are sorted on the source node. They also show that, if there is no ordering on the edges, it is impossible to count the number of triangles using $o(N^2)$ space. Finally, in [124], the authors describe an approximation algorithm for weighted clustering coefficient.

In general, random graphs (ER, CM, SCMM, etc.) have a low transitivity ratio (compared to their density), while real-world graphs generally have a high one. Models which capture

this property remain rather rare (bipartite model, DM model, CCM). An interesting fact about the clustering coefficient is that it is almost always larger in real-world graphs than in a random graph with the same number of nodes and edges. Watts and Strogatz [135] report a clustering coefficient of 0.79 for the actor network (two actors are linked if they have acted in the same movie), whereas the corresponding random graph has a coefficient of 0.00027. For the power grid network, the coefficient is 0.08, much greater than 0.005 for the random graph.

With respect to small-world graphs, the model in [135] has high clustering coefficient and low diameter, while according to [53] it holds that $CC(k) \propto k^{-1}$ for scale-free graphs generated in a particular fashion, where k denotes the node degree.

[118] investigate the plot of $CC(k)$ versus k for several real-world graphs. They find that $CC(k) \propto k^{-1}$ gives decent fits to the actor network, the WWW, the Internet AS-level graphs and others. However, for certain graphs like the Internet-router-level graphs and the power grid graph, $CC(k)$ is independent of k. The authors propose an explanation for this phenomenon: they say that the $CC(k) \propto k^{-1}$ scaling property reflects the presence of hierarchies in the graph. Both the Router and power-grid graphs have geographical constraints (it is uneconomic to lay long wires), and this presumably prevents them from having a hierarchical topology.

In ER random graphs, let the probability that any two neighbors of a node are themselves connected be the connection probability $p = \frac{<k>}{N}$, where $< k >$ is the average node degree with k the node degree. Therefore, the global clustering coefficient is

$$CC_{\text{random}} = p = \frac{< k >}{N}. \tag{2.14}$$

In most random graph models, the probability that two neighbors of a node are themselves connected is $O(N^{-1})$. This is exactly the clustering coefficient of the graph and tends to zero for large N. However, for many real-world graphs, $\frac{CC_G}{<k>}$ is independent of N (see Figure 9 from [7]).

In general, power law degree distribution is closer to real world graphs. For the Generalized linear preferential attachment (GLP) model, [32] find by numerical simulation that the clustering coefficient for graphs generated by means of the GLP model is much closer to that of the Internet than the BA, AB, and Power Law Random Graph (PLRG [4]) models. The bipartite model with preferential attachment (a variation of the bipartite model) also induces large clustering coefficients.

Algorithmic implications. The local clustering coefficient $CC_G(u)$ expresses the local robustness in the graph and thus has practical implications: the higher the local clustering coefficient of a node is, the more interconnected are its neighbors. The global clustering coefficient expresses the average robustness of the network to topological modifications.

Graphs with high local/global clustering coefficients have a very good behavior in dynamic environments with respect to routing. Indeed, the removal of a single link has a very small impact on the global clustering coefficient, and links appearing between nodes which already have a neighbor in common only increase the clustering coefficient.

2.6 Doubling dimension

Definitions. The doubling dimension of a graph G, and more generally of a metric space (X, d), is the least k such that any ball of radius R can be covered by 2^k balls of radius $R/2$ [76].

Complexity, recognition and generation. The problem of determining the doubling dimension of a graph is NP-complete. A 3-approximation algorithm with time complexity in $O(2^k n \log \Delta)$ is presented in [51]. Fraigniaud et al. [68] studied the doubling dimension of the Internet and they verified that the ball growth of the Internet, as well as its doubling dimension, is large. The doubling dimension is much smaller when restricting the measures to balls of large enough radius. It has also been considered in the context of small world and augmented graphs [66, 67].

Algorithmic implications. Efficient compact routing schemes have been proposed for static graphs with bounded doubling dimension, both in the labelled and in the name-independent models. In the labelled model, routing schemes with multiplicative stretch depending on the doubling dimension and using logarithmic routing tables have been proposed in [36, 1]. More precisely, [36] shows how to perform $(1 + x)$-stretch routing on metrics for any $0 < x < 1$ with routing tables of size at most $(d/x)^{O(d)} \log^2(D)$ bits with at most $(d/x)^{O(d)} \log(D)$ entries in any graph with doubling dimension d and diameter D. The tables can be computed in a distributed way using a distributed breadth-first-search algorithm; this result has been extended in [129]. A name-independent routing scheme for graphs with doubling dimension d is provided in [1]. It achieves constant multiplicative stretch and requires tables of size $2^{O(d)} \log n$ bits. Moreover, it has been proven that any name-independent routing scheme using $o(dn)$ bits has stretch at least 3.

2.7 Growth

Definitions. A weighted undirected network is Δ growth-bounded if the number of nodes at distance $2r$ around any given node is at most Δ times the number of nodes at distance r around the node. G has growth dimension s if G is 2^s growth-bounded. An alternative definition is the following: an undirected graph $G = (V, E)$ is called growth-bounded if there exists a polynomial bounding function $f(r)$ such that for every $v \in V$ and $r \geq 0$, the size of any maximal independent set in the r-neighborhood $\Gamma_r(v)$ is at most $f(r)$.

Complexity, recognition and generation. The problem of determining the value Δ bounding the growth of a graph is NP-complete. In [68] it is shown that the ball growth of the Internet is large. We remark that any metric with constant growth-bound has constant doubling dimension, while the opposite is not always true [76].

Algorithmic implications. Abraham and Mahlki [2] proposed a name-independent compact routing scheme for graphs with bounded growth. Given a weighted undirected network and $\epsilon > 0$, their routing scheme allows routing along paths of stretch $1 + \epsilon$ and uses routing tables of only $O(1/\epsilon^{O(\log \Delta)} \log^5 n)$ bits per node with high probability. Duchon et al. [57] proved that if a graph has *moderate* growth, i.e., the size of the balls of radius r is bounded by

$r^{O(1/(r \log r))}$, then it can be turned into a navigable small world (augmented by the addition of some "long links").

2.8 Treewidth

Definitions. The treewidth has been defined by Robertson and Seymour in their Graph Minor Theory series [120]. The interested reader may look at the survey of Bodlaender [20]. Roughly speaking, the treewidth of a graph measures the similarity of the graph to a tree. More formally, a *tree-decomposition* of a graph G consists of a tree T and a family $(X_t)_t \in V(T)$ of subsets of vertices of G (called bags) satisfying the following three properties:

- the union of all the X_t's is $V(G)$;
- for any edge $e = (u, v) \in E(G)$, there is a bag containing both u and v;
- for any $v \in V(G)$, the set $S = \{t \in V(T) | v \in X_t\}$ of bags containing v must induce a subtree of T.

The width of the decomposition is equal to the size of the biggest bag minus one and the *treewidth* of G is the minimum width over all tree-decompositions of G. A *path-decomposition* of G is a tree-decomposition where T is constrained to be a path and the *pathwidth* of G is the minimum width over all path-decompositions of G.

There are several equivalent definitions of the treewidth of a graph (e.g., see [20]). The treewidth of a graph G can be seen as the minimum over all chordal supergraphs H of G of the maximum clique in H. A k-tree is any graph obtained from a clique of size $k + 1$ by iteratively adding a vertex adjacent to some subclique of size k. The treewidth is the smallest k such that G is a partial k-tree, i.e., a subgraph of a k-tree. Treewidth has an algorithmic interpretation in the *graph searching* context: it is equal to the minimum k such that $k + 1$ cops can capture an arbitrary fast and visible fugitive in G.

Complexity, recognition and generation. Computing the treewidth of a graph is NP-complete [10]. Since the class of graphs with treewidth at most k is minor-closed, it holds that the problem of deciding the treewidth is Fixed Parameter Tractable (FPT). An algorithm for computing whether the treewidth of a n-node graph is at most k (and computing a corresponding decomposition) in time $f(k) \cdot n$ has been proposed in [21] but the running time is impractical even for $k = 3$.

The best known approximation algorithm has approximation ratio $\sqrt{\log(\text{treewidth})}$ [61]. Several heuristics have been proposed by Bodlaender at al. [22] and are based on elimination ordering of the vertices (e.g., based on the degree). It would be interesting to design a polynomial time algorithm to decide whether a graph has treewidth at most 3 that could be implemented in practice.

Trees are the (connected) graphs with treewidth 1, while outerplanar graphs have treewidth 2. We remark that, not surprisingly, the Internet (CAIDA) has a large treewidth [49]. Even though it is easy to generate partial k-trees, as far as we know nothing is known concerning the complexity of uniformly generating graphs with treewidth k. Also, k-chordal graphs with maximum degree Δ have treewidth at most $\Delta(\Delta - 1)^{k-3}$ [23].

Algorithmic implications. In general, a tree-decomposition with small width for a graph can be exploited to solve several problems using dynamic programming on the decomposition. Courcelle [46] proved that any problem of the Monadic Second Order Logic can be solved in polynomial time in the class of graphs of bounded treewidth. Moreover, the treewidth is a measure of the connectivity of a graph. Hence, naturally, several compact routing schemes have been proposed for bounded treewidth graphs [24, 47].

A *bramble* in a graph G is a set $(B_i)_i \in I$ of connected subgraphs of G that are pairwise "touching" (either B_i and B_j intersect or there is an edge between a vertex in B_i and a vertex in B_j for any $i, j \in I$). The order of a bramble is the size of a minimum transversal, i.e., the minimum subset of vertices intersecting all elements of the bramble. A graph has treewidth k if and only if it has a bramble of order $k+1$ [9, 127]. For instance, a planar graph has a large treewidth if it contains a large grid as a minor (and thus a large bramble). An apparently interesting question is whether such a structure may be related to communities (see below) or may be used for routing.

2.9 Modularity

Definitions. Modularity has been introduced in [108] in order to check whether a partition of the nodes of a graph represents a good "community structure". Informally, a community is a group of nodes that have a lot of internal links and few external links. The name comes from the study of social networks, which are often observed to be non-homogeneous: some groups of people have relationships between them much denser than average. Communities also appear in biological networks, image segmentation, www networks, etc. [63], and the identification of communities in very large graphs (from thousands to millions of nodes) has become a central challenge in the field of complex networks.

The very definition of community structure is problematic. One naive attempt is to take the minimum cut of the graph. However, this may lead to a very unequal and uninteresting split. One possible remedy is to correct this defect of the cut size as a criterion by blending it with a criterion favouring equal size. A relevant successful attempt was made by Shi and Malik [128] in the field of image segmentation. Shi-Malik's criterion is called the *normalised cut* (NCut).

The formal definition of NCut for a partition into k communities C_i is as follows:

$$\mathrm{NCut} = \sum_i \frac{\mathrm{edges}(C_i, \overline{C_i})}{w_i}, \tag{2.15}$$

where w_i is the weight of a community, i.e., the sum of degrees (or strength) of nodes in C_i.

A drawback of that criterion is that it allows to pick the best k-partition, once we have chosen k, but does not allow to compare meaningfully a k-partition with a k'-partition, for $k \neq k'$. Therefore, we have to find the "best" k by other means; this is the defect fixed by modularity.

Given a partition into k classes of nodes, the relative weight $p_i = w_i/(\sum_j w_j)$ is a fraction of the total weight. Now consider the quantity

$$\mathrm{DiversityInd} = 1 - \sum_i p_i^2, \tag{2.16}$$

which is known as the diversity index in ecology and under many other names in other fields. The diversity tends to be high when the partition has many equal-sized classes, and tends to be low when one class is much bigger than the others. Given a partition, call

$$Q = \text{DiversityInd} - \text{Cut} \tag{2.17}$$

the modularity of the partition. It is maximal for a partition that strikes the right balance between a small cut size (which, alone, tends to produce unequal two-way partitions or even the trivial one-way partition) and a large diversity (which, alone, is optimized by one-node communities). This definition has been extremely popular. One of the reason is that it has several nice interpretations.

The original definition by Girvan and Newman [108] reasons as follows. Given a community C_i, we may compute how many edges are inside the community, which we denote by $\text{edge}(C_i, C_i)$, and how many expected nodes are inside this same community in a random graph, constrained to have the same degrees. In other words, we shuffle the edges in order to maintain the same degree for every node. Let us denote by $\text{ExpectedEdges}(C_i, C_i)$ this quantity. Then, we define the modularity Q as

$$Q = \frac{1}{m} \sum_i \text{edge}(C_i, C_i) - \text{ExpectedEdges}(C_i, C_i), \tag{2.18}$$

where m is the total number of edges and $1/m$ is a normalising factor making Q a quantity between -1 and 1.

If A is the adjacency matrix and $\deg_j$ is the degree of node j, the modularity is then computed as

$$Q = \frac{1}{2m} \sum_i \sum_{j \in C_i} A_{ij} - \frac{\deg_i \deg_j}{2m}. \tag{2.19}$$

Another equivalent definition performs a simple random walk on the graph: the random walker leaves a node at every time step by one of the edges with equal probability (or a probability proportional to the weight of the edge in the case of a weighted graph). The stationary distribution of the community C_i is then simply p_i. Then, consider $p_{ii'}$ the probability to be in C_i and transition to $C_{i'}$ at next step. Then

$$Q = \sum_i p_{ii} - p_i p_i \tag{2.20}$$

which is the probability to stay in the same community, compared to the probability that two random walkers would be in the same community. If this probability is high, it indicates that a random walker tends to be trapped into C_i.

This latter definition lets room for generalizations: If by $p_{ii'}$ we now mean the probability to be in community i at time zero and be in community i' at time s, for a parameter s, then we obtain a quantity called *stability* of the partition for time s [50, 88]. This parameter allows us to find a multiscale hierarchy structure: for high s, only one or two optimal communities are found, while for small s, every node is optimal as a community. This also generalizes another parameter-dependent variant of modularity, called Potts model [119].

Complexity, recognition and generation. Checking whether a given graph admits a partition of modularity larger than a certain real number Q_0 has been proved to be NP-complete [30]. Hence, modularity optimization is hard. Therefore, all the algorithms aiming at finding the modularity-optimal partition are rather crude heuristics, especially when we know that most of the recent algorithms aim at treating millions or billions of nodes, which forces them to run in quasi-linear time and yet make a decent job in finding a community structure with high modularity.

Let us list a few algorithms of community detection, some of them based on modularity optimization, some not. All of the algorithms that optimize modularity can be adapted to optimize stability instead. We give a succinct, incomplete description; see the references for more details. The *Girvan-Newman algorithm* [70] is a hierarchical divisive algorithm (i.e., it cuts the graph in two, then every part in two, etc.). This algorithm runs in time $O(n^3)$ and is mainly of historical interest. The *greedy modularity optimization* [43] starts from isolated nodes and adds links one after another in order to increase modularity; it runs in time $O(n \log^2 n)$. The *Louvain algorithm* [19] merges and moves nodes in order to optimize modularity in a greedy way. It is essentially linear in the number of edges; this method is both very fast and relatively accurate. The *Markov clustering algorithm* [133] describes a random walk on the graph, and alternates between two steps: (i) take the row-stochastic matrix M and square it; (ii) square every entry of M then renormalize every row. It is time and space consuming, but variants of it, where we only keep the k largest entries, cost Nk^2 steps. *Infomap* [121] optimizes a criterion that is very different from modularity and is based on information theory. It mixes greedy search and simulated annealing and seems to give good results, although it is slow. *Walktrap* [92] exploits the fact that a random walker tends to be trapped for a long time in a community. Spectral algorithms, based on exploiting the dominant eigenvectors of the combinatorial or normalized Laplacian, date back to 1973 [62], but many variants designed to optimize specific criteria have appeared in the last ten years [128, 106, 52]. The costly operation here is the computation of eigenvectors of a sparse matrix.

There exist a few classes of artificial graphs, with clearly defined communities, that are used as benchmarks. For example, the Ravasz-Barabasi graph [118] is deterministic. We start with a 5-clique: four peripheral nodes and one central. Then, we add four copies of it and connect the 4×4 peripheral nodes to the central node of the original clique. Again, we add four copies of this graph and link the $4 \times 4 \times 4$ peripheral nodes of those copies to the central node of the original. This is to be repeated a finite number of times. Lancichinetti, Fortunato and Radicchi [89] have produced a family of random benchmarks, where the communities are of various size regulated by a power law, and the degree distribution follows a power law as well.

Algorithmic implications. The existence of communities indicates that the graph is "almost disconnected". On disconnected graphs, algorithms can be applied on every connected component separately. When there are communities, this may be converted into "weakly coupled" computations on every community. This is similar to parallel computation, where the tasks are split in order to minimise the required communication between the different processors.

If a community structure for a graph is known, this suggests ways to route efficiently, first by routing at the level of communities (find the target community), and then by finding the target node in its community.

2.10 Assortativity coefficient

Definitions. In order to define the assortativity coefficient we will first introduce the remaining degree distribution: the normalized remaining degree distribution $Q(k)$ of a given node is defined to be the number of edges leaving that node minus the edge used to reach that node by following a randomly chosen edge on the graph. That is $Q(k) = \frac{(k+1)P(k+1)}{\sum_j jP(j)}$, where $P(j)$ is the probability that a randomly chosen vertex on the graph has degree j. The quantity e_{jk} is also defined to be the joint probability distribution of the remaining degrees of the two vertices at each endpoint of a randomly chosen edge. This quantity is symmetric in undirected graphs ($e_{jk} = e_{kj}$), and satisfies $\sum_{jk} e_{jk} = 1$ and $\sum_j e_{jk} = Q(k)$.

Then, the normalized assortativity coefficient is

$$r = \frac{1}{\sigma_Q^2} \sum_{j,k} jk \left(e_{jk} - Q(j)Q(k)\right) \tag{2.21}$$

where

$$\sigma_Q^2 = \sum_k k^2 Q(k) - \left[\sum_k kQ(k)\right]^2 \tag{2.22}$$

is the normalization term (the maximal value of the numerator that is achieved on a perfectly assortative network) that corresponds to the variance of the distribution $Q(k)$.

A second (equivalent) formulation of this coefficient can be derived from the Joint Degree Distribution (JDD) by setting the joint degree distribution of the nearest neighbors $P(k, k')$ such that $\sum_{k,k'} P(k, k') = 1$ and $P(k, k') = P(k', k)$.

Algorithmic implications. We can classify networks according to the assortativity coefficient r, where $-1 \leq r \leq 1$.

- Disassortative networks (i.e., with assortativity coefficient $-1 \leq r < 0$) exhibit an excess of radial links, i.e., links connecting nodes of dissimilar degrees. These networks are vulnerable to both random failures and targeted attacks. Vertex covers in disassortative graphs are smaller, which is important for applications such as traffic monitoring.
- Assortative networks (i.e., with assortativity coefficient $0 < r \leq 1$) exhibit an excess of tangential links, i.e., links connecting nodes of similar degrees.

2.11 Rich-club connectivity

Definition. For $\rho = 1, \ldots, n$, the rich club connectivity (RCC) $\Phi(\rho/n)$ is the ratio of the number of links in the subgraph induced by the ρ largest-degree nodes to the maximum possible number of such links $n(n-1)/2$. It is thus a measure of how close the subgraphs induced by highly connected nodes are to cliques [138, 97].

There is no uniformly agreed notion of when a network has a rich club connectivity. The most common criteria are that $\Phi(0.01)$ should be high and that the curve $\Phi(\rho/n)$ is flat for small values of ρ. These criteria suffer from several problems: the value above which $\Phi(0.01)$ is considered high depends on the general connectivity of the graph, and the shape of the curve can actually be strongly influenced by the degree distribution. To avoid these issues, an alternative ratio has been defined in [80] and [44]: it is the ratio of the number of links in the

subgraph induced by the ρ largest-degree nodes to the expected number of links that would be present in a maximally random version of the network with the same degree distribution. A network has then a RCC when this ratio is above one. This new ratio leads to significantly different conclusions about which networks have a RCC.

Note that the power law coefficient of $\Phi(\rho/n)$ is also sometimes used. Finally, some authors define $\Phi(k)$ for an integer k as the ratio between the number of edges in the subgraph induced by all nodes with degree larger than k and the number of edges that would be present in a complete graph of the same size. This definition is equivalent to the initial one, up to a change of variable depending on the degree distribution.

Complexity, recognition and generation. $\Phi(\rho/n)$ can easily be computed (for one or all values) by directly applying the definition and the time complexity is $O(n \log n + |E|)$.

According to [138] the PFP (positive feedback preference) model reproduces accurately the RCC behavior of the AS graph, with $\Phi(0.01) = 0.3$ for appropriate parameters. The same holds for the IG model (interactive growth), a particular case of the PFP, with $\Phi(0.01) = 0.32$ (see Section 1.2.3). On the other hand, the Barabasi-Albert (and clearly, the ER) models do not produce graphs with a RCC.

Algorithmic implications. In [138] it is claimed (without a formal proof) that networks having a rich club (sort of clique behavior for high degree nodes) *and* a dissortative connectivity (highly connected nodes are also connected to nodes with low connectivity) have relatively short paths between every pair of nodes, contain many alternative reachable paths and are less robust under node attack. According to [138], the AS graph is considered to have a rich club connectivity, with $\Phi(0.01) = 0.27$, and the rich club would consist of the core-tier. However, [44] claims that the AS graph does not have rich club connectivity, because Φ is lower than what it would have been in a maximally random network with the same degree distribution. They make the same claim for the network of protein interactions. The networks of air transportation and of scientific collaboration have a RCC, even when using the methodology of [44]. In general, the RCC coefficient contains strictly less information than the joint degree distribution.

2.12 Centrality

In the following, we describe four different variants of the centrality property, namely the *degree centrality*, the *betweenness centrality*, the *closeness centrality*, and the *eigenvector centrality*, each of them expressing different characteristics of the topology.

2.12.1 Degree centrality

Definition. The degree centrality is defined as the number of links incident to a node (i.e., the number of ties that a node has). Degree is often interpreted in terms of the immediate risk of node for catching whatever is flowing through the network (such as a virus, or some information). If the network is directed, then we usually define two separate measures of degree centrality, namely indegree and outdegree centrality. Indegree is a count of the number of ties directed to the node, and outdegree is the number of ties that the node directs to others.

For positive relations such as friendship or advice, we normally interpret indegree as a form of popularity, and outdegree as gregariousness.

For a graph $G = (V, E)$ with N vertices, the degree centrality $C_D(v)$ of a vertex $v \in V$ is defined as $C_D(v) = \frac{\deg(v)}{N-1}$. From the centrality of a node, we can define the centralization of a graph. Let v^* be the node with highest degree centrality in G. The degree centrality of the graph G is defined as follows:

$$C_D(G) = \frac{\sum_{v \in V \setminus \{v^*\}} C_D(v^*) - C_D(v)}{H}, \tag{2.23}$$

where H is a normalizing factor, such that $C_D(G) \in [0, 1]$. We now show how to compute it. First, we find $X = (Y, Z)$, the N-node connected graph that maximizes $\sum_{y \in Y \setminus \{y^*\}} C_D(y^*) - C_D(y)$, with y^* being the node with highest degree centrality in X. Clearly, this quantity is maximized when the graph X contains one node that is connected to all other nodes and all other nodes are connected only to this one central node, that is X is a star graph. So we have $H = N - 2$, and the degree centralization of G reduces to:

$$C_D(G) = \frac{\sum_{v \in V \setminus \{v^*\}} C_D(v^*) - C_D(v)}{N - 2}. \tag{2.24}$$

Recall that $\sum_{v \in V} \deg(v) = 2|E|$, and since $C_D(v) = \frac{\deg(v)}{N-1}$, we obtain

$$C_D(G) = \frac{\left(\sum_{v \in V \setminus \{v^*\}} \deg(v^*)\right) - (2 \cdot |E| - \deg(v^*))}{(N-1)(N-2)} = \frac{N \cdot \deg(v^*) - 2 \cdot |E|}{(N-1)(N-2)}. \tag{2.25}$$

The concept of centralization of a graph could be defined for any concept of centrality.

Complexity, recognition and generation. Calculating degree centrality for all nodes V in a graph takes $\Theta(V^2)$ in a dense adjacency matrix representation of the graph, and calculating it for the edges E in a graph takes $\Theta(E)$ in a sparse matrix representation.

2.12.2 Betweenness centrality

Definitions. The betweenness centrality of a vertex (respectively, an edge) is the number of shortest paths in the graph passing through that vertex (respectively, edge). For a graph $G = (V, E)$ of order N, the betweenness $C_B(v)$ of vertex v is computed as follows. We start by computing for each pair of vertices (s, t) the set $P_{s,t}$ of all possible shortest paths between them. We then determine for each pair of vertices (s, t) the fraction of shortest paths that pass through a specified vertex v. Finally, we sum the fraction of shortest paths passing through v over all pairs of vertices (s, t). In other words, $C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$, where σ_{st} is the number of shortest paths from s to t, and $\sigma_{st}(v)$ is the number of shortest paths from s to t that pass through a vertex v [103]. This may be normalised by dividing through the number of pairs of vertices not including v, which is $(n-1)(n-2)$ for digraphs and $(n-1)(n-2)/2$ for undirected graphs. For example, in an undirected star, the center vertex (which is contained in every possible shortest path) would have a betweenness of $(n-1)(n-2)/2$ (1, if normalised) while the leaves (which are contained in no shortest paths) would have a betweenness of 0.

Similarly to the betweenness centrality of a vertex or an edge, we define the *endpoint betweenness* of a pair u, v of vertices as the the number of shortest paths passing through

both u and v. We also define the *source proximal betweenness* (respectively, *target proximal betweenness*) as the number of shortest paths starting form a given source vertex (respectively, ending at a given target vertex).

Complexity, recognition and generation. Calculating the betweenness and closeness centralities of all the vertices in a graph involves calculating the shortest paths between all pairs of vertices on a graph. This takes $\Theta(V^3)$ time using a modified version of the Floyd-Warshall algorithm allowing to find all shortest paths between two nodes. On a sparse graph, Johnson's algorithm may be more efficient, taking $O(|V|^2 \log |V| + |V| \cdot |E|)$ time. On unweighted graphs, calculating betweenness centrality takes $O(|V| \cdot |E|)$ time using Brandes' algorithm [29].

In calculating betweenness and closeness centralities of all vertices in a graph, it is assumed that graphs are undirected and connected with the allowance of loops and multiple edges. When specifically dealing with network graphs, graphs are generally considered without loops or multiple edges to maintain simple relationships (where edges represent connections between two people or vertices). In this case, using Brandes' algorithm will divide final centrality scores by 2 to account for each shortest path being counted twice [29].

The algorithms generating such topologies satisfy the following properties: they exhibit incremental generation, deterministic algorithms guarantee that the topologies have the property, while stochastic algorithms lead to topologies where the property exists with high probability.

2.12.3 Closeness centrality

Definitions. There is no common agreement on the closeness centrality definition in the literature. Here, we survey the most commonly used definitions. In the network theory, closeness is a sophisticated measure of centrality defined as the mean geodesic distance (i.e., the shortest path distance) between a vertex v and all other vertices reachable from it:

$$C_C(v) = \frac{\sum_{t \in V \setminus \{v\}} d_G(v,t)}{N-1} = \bar{d}(v). \tag{2.26}$$

where $N \geq 2$ is the size of the "connectivity component" V of the network reachable from v. Closeness can be regarded as a measure of how long it will take to spread information from a given vertex to other reachable vertices in the network [111]. We remark that this notion is sometimes called the *barycenter centrality*.

Some authors define closeness to be the reciprocal of this quantity, in the following way. The closeness $C'_C(v)$ of a vertex v is the reciprocal of the sum of the distances to all other vertices of $V \setminus \{v\}$ [122],

$$C'_C(v) = \frac{1}{\sum_{t \in V \setminus \{v\}} d_G(v,t)} = \frac{1}{\bar{d}(v)}. \tag{2.27}$$

A variant is the *random-walk centrality*, introduced in [114], that measures the speed at which randomly walking messages (i.e., hot-potato routing) reach a vertex from elsewhere in the network. The *information centrality* [113] is another closeness measure, which bears some similarity with the random-walk centrality [114]. In essence it measures the harmonic mean length of paths ending at a vertex v, which is smaller if v has many short paths

connecting it to other vertices. Another variant, proposed in [48] in order to measure the network vulnerability, modifies the definition so that it can be used for disconnected graphs: $C''_C(v) = \sum_{t \in V \setminus \{v\}} 2^{-d_G(v,t)}$. The total closeness of the graph is also easier to compute.

Complexity, recognition and generation. The main difficulty stems from finding the matrix of distances between all pairs of nodes. This can be done in $D \cdot N^2$ steps, where D is the diameter and N the number of nodes. The algorithms generating such topologies satisfy the following properties: they exhibit incremental generation, deterministic algorithms guarantee that the topologies have the property, while stochastic algorithms lead to topologies where the property exists with high probability.

2.12.4 Eigenvector centrality

Definitions. Let $G = (V, E)$ be a graph of order N, and let A be its adjacency matrix. Hence $A_{i,j} = 1$ if the i-th node is adjacent to the j-th node, and $A_{i,j} = 0$ otherwise. More generally, the entries in A can be real numbers representing connection strengths, as in a stochastic matrix. Let λ be the eigenvalue of A and let v be the eigenvector. The *eigenvector centrality* [28] is the principal eigenvector of the adjacency matrix A of the graph. This equation leads to the interpretation that a node that has a high eigenvector score is one that is adjacent to nodes that are themselves high scorers. The idea is that even if a node influences just another node, who subsequently influences many other nodes (who themselves influence more nodes), then the first node in that chain is highly influential. At the same time, eigenvector centrality provides a model of nodal risk such that a node's long-term equilibrium risk of receiving traffic is a function of the risk level of its contacts.

Eigenvector centrality thus provides a measure of the importance of a node in a network. It assigns relative scores to all nodes in the network based on the principle that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. Let x_i denote the score of the i-th node, and let the centrality score of that node be proportional to the sum of the scores of all nodes which are connected to it. Hence

$$x_i = \frac{1}{\lambda} \sum_{j \in V,\ (i,j) \in E} x_j = \frac{1}{\lambda} \sum_{j=1}^{N} A_{i,j} x_j. \tag{2.28}$$

In vector notation, this can be rewritten as the eigenvector equation $\mathbf{Ax} = \lambda \mathbf{x}$. In general, there will be many different eigenvalues λ for which an eigenvector solution exists. However, the additional requirement that all the entries in the eigenvector should be positive implies (by the Perron-Frobenius theorem) that only the greatest eigenvalue results in the desired centrality measure. The i–th component of the related eigenvector then gives the centrality score of the i–th node in the network.

Google's PageRank is a variant of the Eigenvector centrality measure. It computes the stationary probability distribution of a simple random walk on the graph, which happens to be the dominant left eigenvector of $D^{-1}A$, where D is the diagonal matrix of outdegrees.

Complexity, recognition and generation. Power iteration is one of many eigenvalue algorithms that may be used to find this dominant eigenvector. Its complexity depends crucially on the spectral gap of the matrix. Eigenvector centrality is related to other notions of centrality as well.

2.13 Chordality

Definitions. The chordality of a graph G is the length (i.e., the number of vertices) of the longest induced cycle in G. In other words, a graph is k-chordal if every cycle of length at least $k+1$ contains a chord. Graphs with chordality 3 are also known as *chordal graphs*. They have interesting connectivity properties: e.g., in a chordal graph, any minimal separator is a clique [72]. A graph is chordal if and only if it admits a perfect elimination ordering, i.e., an ordering of its vertices $v_1, v_2, \ldots, v_n$ such that, for any $i < n$, v_i is simplicial ($N[v_i]$ induces a clique) in $G[v_i...v_n]$ [72].

Complexity, recognition and generation. Deciding if a graph is chordal can be done in linear time [72]. The problem of computing the chordality of a graph is NP-complete; the argument proceeds as follows. Let G be an induced subgraph of a grid and let H be obtained from G by subdividing all its edges once. There is a correspondence between any induced cycle in H and any cycle in G. Since computing the longest cycle in a subgrid is NP-complete (see [79]), the result follows. Furthermore, finding the longest induced path is $W[2]$-complete [38]. Moreover, the problem is FPT in planar graphs [84]; this result relies heavily on the Graph Minors framework.

Relation to other properties. The chordality of a graph G is related to many other graph parameters. For instance, the treewidth of a graph G can be seen as the minimum over all chordal supergraphs H of G of the maximum clique in H. Moreover, chordal graphs are known to be at most 1-hyperbolic, where 1-hyperbolic chordal graphs have been characterized in [31], and a δ-hyperbolic graph is at least $\lfloor \delta/2 \rfloor/2$-chordal [137]. However, there is no relation between the k-chordality and its density or its average path length since it is possible to build graphs with arbitrary gap between those values.

Algorithmic implications. This class of graphs received particular interest in the context of routing: both in the compact routing framework, but also in greedy routing through the "small-world" phenomenon. Dourisboure and Gavoille [56] proposed routing tables of at most $\log^3 n / \log\log n$ bits per node that are computable in time $O(m + n\log^2 n)$, and induce a routing scheme with additive stretch $2\lfloor k/2 \rfloor$ in the class of k-chordal graphs. Also, Dourisboure [55] proposed routing tables of size at most $\log^2 n$ bits that are computable in polynomial time, but exhibit additive stretch $k+1$. Furthermore, a distributed algorithm has been proposed to compute routing tables of size $2k_v \log n$ bits per node v having degree k_v that are computable in time $4D$ and lead to a routing scheme with additive stretch $k-1$ in the class of k-chordal graphs with diameter D (see [112]). Graphs with small chordality can be efficiently augmented (by adding few links) to get the good properties of small worlds in terms of routing [64].

2.14 Spectral properties

Definitions. Studies in spectral graph theory (see [40] for more details) use the values of eigenvalues of the adjacency matrix and of the Laplacian matrix of the graph. More specifically, the following notions are exploited: The *graph spectrum* is defined as the set of eigenvalues λ of its adjacency matrix A. The *combinatorial Laplacian* L of a graph $G = (V, E)$

is a $N \times N$ symmetric matrix with one row and column for each node defined by $L = D - A$, where D is the degree matrix, which is the diagonal matrix formed from the vertex degrees and A is the adjacency matrix. The eigenvalues of the combinatorial Laplacian L, denoted by λ_i, are enumerated in increasing order: $\lambda_1 = 0 \leq \lambda_2 \leq \cdots \leq \lambda_N$. The *normalized Laplacian* L_N of a graph $G = (V, E)$ is a $N \times N$ symmetric matrix with one row and column for each node defined by $L_N = D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2}$, where I is the identity matrix, D is the degree matrix, which is the diagonal matrix formed from the vertex degrees and A is the adjacency matrix.

Algorithmic implications. Spectral graph theory offers an alternative approach for studying structural graph properties, using well known and powerful tools from mathematics. The second smallest eigenvalue of the Laplacian matrix of a graph is greater than 0 if and only if G is a connected graph. Note that the eigenvector corresponding to the second smallest eigenvalue (i.e., the algebraic connectivity) of the Laplacian matrix of a graph is known as the Fiedler vector.

2.15 Hyperbolicity

Definitions. Gromov [73] defines the notion of δ-hyperbolic metric spaces using the notion of δ-thin triangles. Given any three points x, y, and z of a hyperbolic metric space, the triangle (x, y, z) is δ-thin if any point of the geodesic joining x and y is at distance at most δ of one of the geodesics joining x to z or y to z. A δ-hyperbolic space is a geodesic metric space in which every geodesic triangle is δ-thin. For instance, trees are 0-hyperbolic: a geodesic triangle in a tree is just a subtree, so any point on a geodesic triangle is actually on two edges. The standard Euclidean space is ∞-hyperbolic; i.e., it is not hyperbolic. In fact, the value of δ represents the curvature of the metric space; the higher the value δ is, the less curved the metric space is.

An alternative definition given by Gromov [73], and called the 4-*points condition*, is the following. A metric space is δ-hyperbolic if for any four points u, v, w, x the two larger of the distance sums $d(u, v) + d(w, x)$, $d(u, w) + d(v, x)$, $d(u, x) + d(v, w)$ differ by at most 2δ. These notions extend to connected graphs, and we say that a connected graph $G = (V, E)$ equipped with its standard graph metric d_G (shortest path distance) is δ-hyperbolic if the metric space (V, d_G) is δ-hyperbolic. In other words, a connected graph $G = (V, E)$ is δ-hyperbolic if it satisfies the 4-points condition. The hyperbolicity of a graph measures its tree-likeness. The less the value of δ is, the more the graph looks like a tree.

Complexity and recognition. Determining the hyperbolicity δ of a graph of order N can be done in time $O(N^4)$. In fact, since all all 4-tuples have to be checked, the time complexity is in $O(\binom{N}{4})$. Unfortunately, this time complexity in almost impractical for large graphs and approximation algorithms are needed. For instance, for a graph of order $10,000$, the number of 4-tuples to consider is higher than $4 \cdot 10^{14}$, which requires approximately 100 hours of computation assuming that 10^9 4-tuples are evaluated per second.

A 2-approximation algorithm, with running time in $O(\binom{N}{3})$, is obtained fixing one vertex and evaluating all possible 4-tuples containing that vertex [39].

Relation with other properties. First, observe that the hyperbolicity of a graph is bounded by its diameter divided by two. Secondly, although both the treewidth and the hyperbolicity of a graph express how the graph is close to a tree, there is no relation between these parameters. Indeed, one can easily build a graph with large treewidth and large hyperbolicity (e.g., a grid), or a graph with large treewidth and small hyperbolicity (e.g., a clique), or a graph with small treewidth and large hyperbolicity (e.g., a cycle). Thirdly, a k-chordal graph has hyperbolicity at most $\lfloor k/2 \rfloor/2$ [137], but the knowledge of the value δ of the hyperbolicity gives no information on the possible value of the k-chordality of the graph. However, it has been proved that chordal graphs are at most 1-hyperbolic, and 1-hyperbolic chordal graphs have been characterized in [31].

Chapter 3

Experimental evaluation and comparison

In this chapter, we report numerical results from an investigation of graph models with respect to the properties they satisfy that has been carried out during the first year of EULER. This work has been conducted in cooperation with T4.2 since various tools (currently being) developed or relevant to that context have been used to perform either graph generation or graph property testing in our investigation. In addition, we discuss the issue of computing (i.e., testing or measuring) properties in a distributed fashion following simple computational models that are amenable to be used in a dynamic network environment.

3.1 Methodology and setting

We now describe the methodology and setting we have used. This study consists of the following main steps. First, we have aimed to construct graphs using different topology generators (based on the models provided in Chapter 1) and different parameters. The selection of the models and parameters that we have considered so far has been a process depending on the state-of-the-art in the related literature as well as on our findings during the investigation. For different sets of selected models and parameters, we have constructed many graphs of similar size. Then, we have tested properties in these graphs using code/tools available by the partners. The results from these tests are presented in Section 3.2. In the following, we give more details for the models and corresponding parameters we have used in our investigation. Our approach is graphically depicted in Figure 3.1. We remark that the results of the investigation have provided insights on which models and parameters are interesting for the particular graph size we have considered. This is captured by the feedback arrow in Figure 3.1.

Our plan has been to create graphs with $1,000$ nodes and, in particular, to generate 100 graph instances per generator and parameter profile. Five parameter profiles per each generator model have been planned, where the "middle" profile consists of the best known values (where available) that are/seem to be closer to today's Internet AS/routers graph; the remaining profiles have nearby parameters. As topology models, we have considered the following: Erdős-Rényi ($G_{n,p}$ and $G_{n,m}$), GLP, BRITE, R-MAT, and Inet. Note that we have not been able to find interesting parameters for the R-MAT model and the particular size of graphs we report here. Also, Inet generator can construct topologies with at least $3,037$

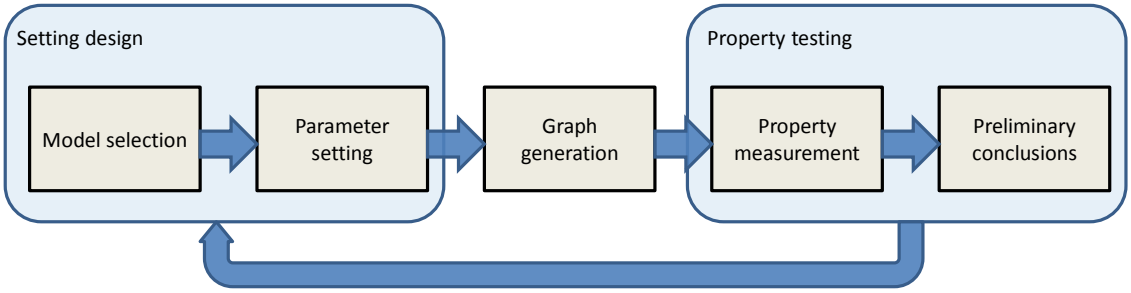


Figure 3.1: An overview of our approach.

nodes. So, these models have been excluded from the current investigation. Of course, they will be part of our future work which aims to extend the investigation reported here to more profiles (models, parameter values). Most of the properties described in Chapter 2 have been considered. In the following, we give more details on the models/parameters considered and selected as well as on the implementations.

The generated graphs are stored in separate files (one file per graph). The filename convention is as follows:

`number-of-nodes_profile-number_instance-number.model-name`

For example, `1000_1_5.glp` is the fifth generated instance of the first profile of the GLP model, and the graph has 1000 nodes. Each evaluated property of a graph is stored in a separate file. The file name is the same as the input graph file name for which the property has been determined. File extensions correspond to properties and are:

Extension	**Property name**
`cc`	Clustering coefficient
`tra`	Transitivity
`dd`	Degree distribution
`hyp`	Hyperbolicity
`bet`	Betweenness
`pl`	Path length

For example, `1000_glp_1_5.hyp` contains the exact hyperbolicity value and the statistical distribution of hyperbolicity of the 4-tuples of the fifth generated instance of the first profile of the GLP model, and the graph has 1000 nodes.

Each file containing the data for a graph has a first line describing the model name, parameters and the instance number, and is commented by a "#" as a first character. Then, the next lines describe the graph as an edge list. Each line contains a vertex id followed by a "␣" (space) character and then by a second vertex id to which it is connected. An isolated node is represented by a unique node id per line. For example, the first few lines of file `1000_1_5.glp` are as follows:

```
# profile:  Instance:  5, profile:  1, generator:  glp, seed:  15,
m0:  6, beta:  0.6447, p:  0.4695, m:  1.13
960  724
860  724
425  960
979  860
52   979
6    860
⋮    ⋮
```

We remark that this is similar to the format of the latest CAIDA maps.

Each computed property is stored in a separate file in a format that is easily usable by `gnuplot`. For instance, the clustering coefficient is stored in one line with the first number as the global clustering coefficient of the graph and then the local clustering coefficients ordered by node id. The first line describes the property format and is commented by a "#" character.

The generated graph files are compressed in one zip (200MB) archive that is available at the following address:

`ftp://ftp-sop.inria.fr/members/David.Coudert/EULER/Topologies.zip`

The archive content is organized as follows. The entire set of generated files are contained in the root folder of the archive. Each set of instances from the same model and profile are contained in a dedicated folder named after the model followed by the profile number. For example `brite_1` is the set of BRITE instances generated with profile 1. Each set of graphs generated with a specific profile is stored in a folder named by the profile number. For example, `profile_1` is used for the entire set of generated graphs with profile 1. The set of configuration files which describe the generation process is stored at the root folder. There is a configuration file for each profile and a configuration file for each model and profile in order to allow specific model graph generation.

We describe in Table 3.1 the parameters of the different models we use in our investigation. The exact parameters we have used to generate our instances (100 graphs per profile) are specified in Tables 3.2-3.5. The parameters have been chosen according either to known (i.e., theoretically proven) behavior, or to the original papers describing the corresponding model.

Table 3.1: Model parameters (number of nodes is 1000).

Model	**parameter**	**Description**
$G(n,p)$	p	Probability to add a new edge
$G(n,m)$	m	Number of edges
GLP	m_0	Number of initial nodes
	p	Probability to be in the case of adding new edges. A new node is added with probability $1-p$
	β	Preferential attachment parameter
	m	Number of new links added at each step
BRITE (Only flat topology parameters are considered)		

Continued on next page

Table 3.1 – continued from previous page

Model	parameter	Description
	h	Size of one side of the plane (≥ 1)
	ℓ	Size of one side of a high-level square (≥ 1)
	N	Number of nodes
	NP	Node placement (Random, HeavyTailed)
	m	Number of links added per new node (≥ 1)
	Model	Probability model (ASWaxman, ASBarabasiAlbert, RouterWaxman, RouterBarabasiAlbert)
	α	Waxman exponent
	β	Waxman exponent
	GT	How nodes join the topology (Incremental, Random)
	BWDist	Bandwidth assignment to links (Constant, Uniform, Exponential, HeavyTailed)
	MaxBW	Maximum link bandwidth value (> 0)
	MinBW	Minimum link bandwidth value (> 0)
Inet (Only topologies of at least 3 037 nodes can be generated. Model excluded from investigation)		
	k	Fraction of degree one node
	n	Size of the plane used for node placement
R-MAT (No parameters found to map Internet (Power law) topology. Pending)		
	a	Probability to add a new edge in matrix A
	b	Probability to add a new edge in matrix B
	c	Probability to add a new edge in matrix C
	S	2^S is the height of the matrix
	S_2	2^{S_2} is the length of the matrix
	E	Set of $\|E\|$ edges

Profile name	p	Theoretical property of chosen parameter
`1-gnp`	0.0005	No connected component of size $> O(\log n)$
`2-gnp`	0.001	One large connected component of size $> n^{2/3}$
`3-gnp`	0.003	One giant connected component
`4-gnp`	0.007	Connected graph
`5-gnp`	0.5	Largest clique of size $k(n)$ or $k(n)+1$

Table 3.2: Profile parameters for $G(1000, p)$ according to known (theoretically proven) behavior.

3.2 Numerical results

We have reported in Table 3.6 the average values we have computed for several properties per profile. These values are either average values over all vertices of the graphs, or global properties. All distributions of local properties are reported in Appendix A.

Concerning the 1-gnp, 2-gnp, and 5-gnm profiles, they have been used to test the validity

Profile name	m	Justification of choosen number of edges
`1-gnm`	2236	Same density as CAIDA AS map of 2004
`2-gnm`	2806	Same density as CAIDA AS map of 2011
`3-gnm`	5000	Denser topology
`4-gnm`	1500	Sparser topology
`5-gnm`	499500	Complete graph, only one topology needed

Table 3.3: Profile parameters for $G(1000, m)$ producing graphs with various densities.

Profile name	h	ℓ	m	NP	GT	Model	BWDist
`1-brite`	1000	100	1	HeavyTailed	incremental	ASWaxman	constant
`2-brite`	1000	100	1	Random	incremental	ASWaxman	constant
`3-brite`	1000	100	1	HeavyTailed	incremental	ASBarabasi	constant
`4-brite`	1000	100	1	Random	incremental	ASBarabasi	constant
`5-brite`	1000	100	1	HeavyTailed	random	ASWaxman	constant
`6-brite`	1000	100	1	Random	random	ASWaxman	constant
`7-brite`	1000	100	1	HeavyTailed	random	ASBarabasi	constant
`8-brite`	1000	100	1	Random	random	ASBarabasi	constant

Table 3.4: Profile parameters for BRITE. Observe that the BRITE model seems to behave the same with different values of h, ℓ and m.

Profile name	m_0	p	β	m	Justification of chosen parameter
`1-glp`	6	0.4695	0.6447	1.13	Reference parameters to map the Internet AS topology [32]
`2-glp`	20	0.55	0.75	1	According to [132], these values best fit the Internet

Table 3.5: Profile parameters for GLP. Average node degree seems to behave the same with different values of m_0 - Internet topology on AS-level: model, generation methods, and tool.

of the implementation of the random graph generators with respect to the corresponding models, and also to check validity of property tests. In particular, the 5-gnm graph is a clique of order 1 000 which is used only to check that computed values are correct for such an extreme graph which has diameter one, assortativity one, global clustering coefficient one, etc. These values are thus not reported. Furthermore, since 1-gnp and 2-gnp graphs are not connected and have small giant components, the comparison of distributions and values of properties with larger graphs provides rather insignificant information and, hence, the corresponding results are not reported here.

Concerning graphs generated using the BRITE model, almost all generated graphs were trees, and so these results are rather meaningless. We have thus chosen not to report on them in this deliverable either. However, we plan to perform experimentation on larger instances (i.e., with more than 3 000 nodes) for which we know that generated graphs have more interesting structure.

In order to compare our results to existing maps of the Internet, we have computed several properties for some CAIDA AS maps and of the main 2-connected components of these maps. The results are reported in Tables 3.7 and Table 3.8; see also the distribution of several properties in CAIDA AS maps in Appendix A.2 page 95. Although CAIDA AS maps provide only a partial view of the real AS network due to measurement bias, the reported results are interesting as reference values and also for testing the tools we use.

Connectivity. When studying communication networks, connectivity is the most basic structural property requirement for a topology, since if the topology is not connected it is not possible for every pair of nodes to communicate. Also, we use this property test only to validate the correctness of the implementation of gnp and gnm graphs generators; glp graphs have a unique connected component by construction. As expected with the chosen parameters for gnp profiles (see Table 3.2), the 3-gnp graphs have a giant connected component (of average size 939.9) and the 4-gnp graphs have a unique connected component with high probability (on average, a single node is missing). We observe a similar behaviour in the gnm profiles. Remark that as expected from the chosen parameters, the 3-glp graphs and 4-gnm graphs have similar structure since with the probability of choosing an edge of 0.003 for 3-gnp, the expected number of edges is 1498, that is two edges less than those of the 4-gnm graphs.

Now, concerning 2-connected components, the structure of the gnp and gnm graphs are also similar. That is, 3-gnp and 4-gnm graphs have a giant 2-connected component which is smaller than the giant connected component, thus indicating that the graphs have a huge core (the 2-connected component) to which many pending edges are connected. For the other gnp and gnm profiles, however, the giant 2-connected component is almost the giant connected component. The structure of 1-glp and 2-glp graphs is rather different since they have a core of around 1/4 of the nodes and a large number of pending edges or small 2-connected components are attached to that core.

Clustering coefficient. As explained in Chapter 2, the local clustering coefficient $CC_G(u)$ expresses the local robustness in the graph and thus has practical implications: the higher the local clustering coefficient of a node is, the more interconnected its neighbors are. The global clustering coefficient expresses the average robustness of the network to topological modifications.

In Table 3.6, we have reported the average global clustering coefficients for some of the

Property	**Profile**								
	3-gnp	**4-gnp**	**5-gnp**	**1-gnm**	**2-gnm**	**3-gnm**	**4-gnm**	**1-glp**	**2-glp**
Number of edges of G	1 493.7	3 496.7	249 745.5	2 235.7	2 805.9	5 000	1 494.9	1 555.5	1 617.8
Number of (non-isolated) nodes	939.9	999.1	1 000	988.2	996.1	999.9	941.1	1 000	1 000
Degree	3.2	7	499.6	4.6	5.6	10.0	3.2	3.1	3.2
Density	0.003	0.007	0.5	0.005	0.006	0.01	0.003	0.003	0.003
Diameter	14.1	7.0	2.0	9.75	8.2	5.4	14.3	7.74	7.0
Radius	8.5	5.0	2.0	6.1	5.5	4.0	8.5	4.1	4.0
Number of connected components	5.48	1.0	1.0	1.3	1.07	1.0	5.21	1.0	1.0
Largest connected component	939.84	999.12	1000.0	988.21	996.07	999.94	941.1	1000.0	1000.0
Number of blocks	168.83	7.42	1.0	52.69	20.54	1.47	167.38	756.34	797.36
Number of 2-connected components	1.07	1.0	1.0	1.02	1.0	1.0	1.05	1.35	1.17
Largest 2-connected component	771.93	992.7	1000.0	936.5	976.53	999.47	774.64	244.25	203.47
Number of connected triples	4 492.7	2 4428.3	$1.25 \cdot 10^8$	9 977.3	15 720.5	49 938.5	4 480.6	51 273.3	59 395.5
Shortest path length	6.2	3.8	1.5	4.8	4.2	3.3	6.2	3.4	3.2
Assortativity	0.84	0.90	1.0	0.87	0.89	0.92	0.84	0.19	0.30
Global clustering coefficient	0.0034	0.0028	0.0085	0.0084	0.0092	0.0112	0.0044	0.049	0.068
Closeness centrality (C'_C)	0.09	0.08	0.04	0.09	0.09	0.09	0.09	0.37	0.36
Vertex degree centrality	0.007	0.01	0.05	0.008	0.009	0.012	0.007	0.14	0.11
Vertex betweenness	0.02	0.005		0.01	0.007	0.004	0.02	0.12	0.07
Exact hyperbolicity	4.2	2.8	1.0	3.3	3.0	2.0	4.2	1.7	1.5
Heuristic hyperbolicity	3.4	2.0	1.0	2.8	2.5	2.0	3.4	1.1	1.0
Mean hyperbolicity	0.55	0.34	0.28	0.43	0.38	0.32	0.55	0.15	0.12
Greedy chromatic number	11.0	12.3	125	12.0	12.0	13.0	11.0	12.0	14.8

Table 3.6: Average values of some properties per graph profiles.

Table 3.7: Average values of properties for some CAIDA AS maps.

Property	**CAIDA AS maps**						
	2004/01/05	**2004/12/06**	**2005/12/05**	**2006/12/25**	**2007/12/03**	**2010/01/20**	**2011/01/16**
Number of nodes	16 301	18 501	20 889	23 918	26 690	33 508	36 878
Number of edges	32 955	38 265	41 820	49 089	53 336	75 001	103 485
Density	0.00025	0.00022	0.00019	0.00017	0.00015	0.00013	0.00015
Maximum degree	2 331	2 328	2 379	2 358	2 632	2 631	2 972
Average degree	4.04	4.14	4.00	4.10	4.00	4.48	5.61
Diameter	10	10	10	11	10	11	11
Radius	5	5	6	6	5	6	6
Avg. shortest path length	3.77	3.78	3.83	3.87	3.87	3.87	3.81
Number of triangles	74 559	96 402	85 173	121 734	102 546	270 237	1 138 473
Number of connected triples	8 845 504	10 234 780	11 133 854	13 093 380	14 937 246	23 098 568	34 069 864
Global clustering coefficient	0.025	0.028	0.023	0.028	0.021	0.035	0.10
Assortativity	0.022	0.022	0.022	0.022	0.021	0.028	0.040
Closeness centrality, C'_C	0.27	0.27	0.27	0.26	0.26	0.26	0.27
Vertex degree centrality	0.14	0.13	0.11	0.10	0.10	0.08	0.08
Exact hyperbolicity	2.5	2.5	2.5	2.5	2.5	2	2
Heuristic hyperbolicity	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Mean hyperbolicity	0.17	0.13	0.17	0.16	0.19	0.15	0.16

Property	Largest 2-connected component of CAIDA AS maps						
	2004/01/05	**2004/12/06**	**2005/12/05**	**2006/12/25**	**2007/12/03**	**2010/01/20**	**2011/01/16**
Number of nodes	10 424	11 910	13 319	14 949	16 348	20 940	23 214
Number of edges	27 061	31 666	34 236	40 105	42 972	62 406	89 783
Density	0.0005	0.00045	0.00039	0.00036	0.00032	0.00028	0.00033
Maximum degree	1 952	1 964	2 019	2 006	2 277	2 400	2 679
Average degree	5.19	5.32	5.14	5.37	5.26	5.96	7.74
Diameter	8	8	8	8	8	8	8
Radius	4	5	4	4	4	4	4
Avg. shortest path length	3.38	3.40	3.45	3.46	3.46	3.49	3.45
Number of triangles	74 536	96 378	85 137	121 695	102 486	270 159	1 138 362
Number of connected triples	6 556 124	7 677 814	8 327 001	9 937 872	11 189 315	18 226 477	28 307 912
Global clustering coefficient	0.034	0.038	0.031	0.037	0.027	0.044	0.12
Assortativity	0.022	0.024	0.022	0.024	0.021	0.028	0.049
Closeness centrality, C'_C	0.30	0.30	0.30	0.29	0.29	0.29	0.29
Vertex degree centrality	0.19	0.17	0.15	0.13	0.14	0.11	0.12
Exact hyperbolicity	2.5	2.5	2.5	2.5	2.5	2	2
Heuristic hyperbolicity	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Mean hyperbolicity	0.17	0.13	0.17	0.18	0.19	0.15	0.15

Table 3.8: Average values of properties of the largest 2-connected component of some CAIDA AS maps.

random graphs we have generated per profile. Surprisingly, the value obtained for the 1-glp profile differs from the value given in [32] for graphs generated with exactly the same parameters. More precisely, the value reported in [32] is 0.35 but we have computed it to be 0.049, that is 10 times less. This is apparently due to the particular graph size we use.

In order to check the validity of our implementations and our results, and to understand possible differences in the interpretation of the models, we did the following extra experiments. First, we have computed the global clustering coefficient for our set of graphs using two different software implementations, namely *Grph* (see Chapter 4) and *NetworkX*[1]. The results of the computations are exactly the same, 0.049, thus ensuring the validity of both our implementation and of the computed value. Next, we have generated new sets of graphs of 1 000 nodes with the 1-glp profile, and have used two distinct software implementations providing GLP graphs generators, namely *Grph* and *aSHIIP*[2]. The average values of the global clustering coefficient we have found are different. More precisely, for graphs generated with *Grph*, the average global clustering coefficient is 0.05, but for graphs generated with *aSHIIP* it is 0.02, so even lower than in [32]. This raised questions on the correctness of the available GLP generators implementation. Also, in order to estimate the accuracy of the

Property	Reference value [32]	*Grph*	*aSHIIP*
Clustering coefficient	0.35	0.045	0.024
Number of edges	18 346	14 000	30 250
Largest 2-connected component	–	1 984	8 222

Table 3.9: Comparison of the clustering coefficient for GLP instances of size 8 613 generated using the 1-glp profile and two different implementations.

GLP generators implementation, we have generated using both *Grph* and *aSHIIP* and the 1-glp profile a set of 100 graphs with 8 613 nodes, that is the size of the graphs used in [32]. We have reported in Table 3.9 the average number of edges of these graphs, the average global clustering coefficient, and the average size of the largest 2-connected components we have computed. We have also reported the values obtained in [32]. We observe a large deviation in the number of edges of the generated graphs compared to [32], and furthermore a large deviation of the size of the largest 2-connected components. From these values, we can explain the differences in the implementations of the GLP generators and so of the values of the clustering coefficients. Indeed, the global clustering coefficient is large if the graph has a large number of triangles. Clearly, graphs generated using *aSHIIP* have a very large 2-connected component, few pending edges (and so vertices of degree 1), but also very few triangles. Graphs generated using *Grph* have a smaller 2-connected component (around 1/4 of the vertices) and so more pending edges, but this 2-connected component contains many triangles. Last, graphs used in [32] have clearly a much larger number of triangles than graphs obtained with other generators. Unfortunately, since we do not have access to the GLP graph generator of [32], we cannot investigate further on the difference of interpretation of the GLP model.

Finally, we have computed the global clustering coefficient of some CAIDA AS maps and of the main 2-connected components of these maps. The results are reported in Tables 3.7 and Table 3.8. Although reported values have to be taken with caution due to measurement bias,

[1] NetworkX: `http://networkx.lanl.gov/`.

[2] A random topology generator of interdomain (*aSHIIP*): `http://wwwdi.supelec.fr/software/ashiip/`.

the computed global clustering coefficients values are closer to the values we have obtained on graphs generated with *Grph* and *aSHIIP*, and in particular for the largest 2-connected components, than to the value reported in [32].

From the above discussion and the values of the global clustering coefficient reported in Table 3.6 for all graph profiles, we can conclude that the global clustering coefficient does not provide enough information that could be useful for routing schemes. Indeed, this parameter evolves with the ratio of the number of triangles over the number of paths of length two. Therefore, a very dense graph, such as the graphs of the 5-gnp profile, has a small global clustering coefficient although such a graph will be very resilient to topological modifications.

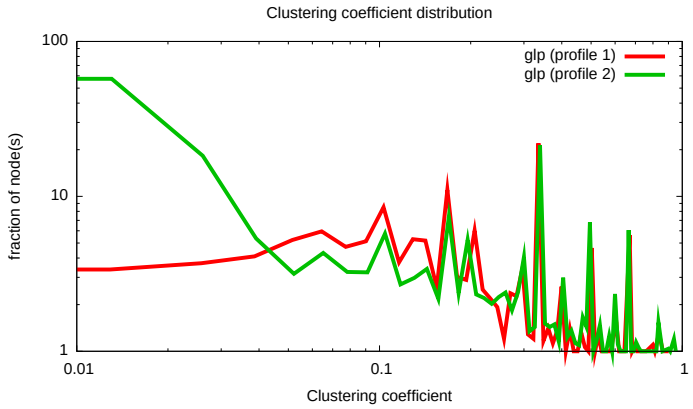


Figure 3.2: Local clustering coefficient distribution for profiles 1-glp and 2-glp (log log scale).

In Figure 3.2, we report on the local clustering coefficient distribution for graphs with profiles 1-glp and 2-glp generated with *Grph*. We observe for 1-glp that a large fraction of the nodes have a low clustering coefficient which is explained by the number of nodes that are not part of the main 2-connected component. On the other side, a small fraction of the nodes have a large clustering coefficient, and these nodes are those of the main 2-connected component. Consequently, a large fraction of the nodes are very sensitive to local topological changes, while nodes from the core of the graph have a very low sensitivity to local topological changes. Such information can certainly be exploited by routing schemes.

Length of paths. Initially, we observe that the average shortest path length of graphs with gnp and gnm profiles reported in Table 3.6 follows the expected values for such graphs, that is $\log_{\text{deg}}(N)$. For example, for the 4-gnp profile, we found an average shortest path length of 3.8 and we have $\log_7(1\,000) = 3.55$. This again allows us to validate the correctness of our implementations with respect to the models. Now, concerning glp profiles, we observe that the distance distribution reported in Figure 3.3 is concentrated around the average value of the shortest path length. Since these graphs have a low density (0.003), this indicates the presence either of hub nodes (nodes of very high degrees) or of a dense core with very low average shortest path length; the latter implies that the distance from vertices outside the core to the core is also low. Indeed, the presence of hub nodes is confirmed by the degree distribution reported in Figure 3.4 since some vertices have degree 100. Furthermore, specific measurements show that the largest 2-connected components of 1-glp graphs have average shortest path length of 2.6, while for 2-glp graphs the corresponding value is 2.5. The

corresponding distribution in CAIDA AS maps (see Figure A.16 in Appendix A.2 page 96) is similar.

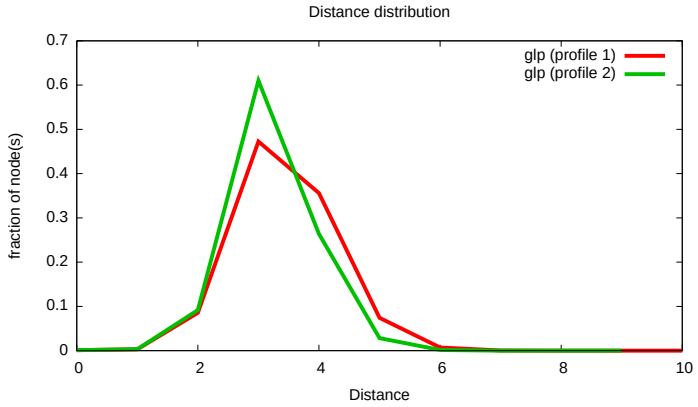


Figure 3.3: Average shortest path length distribution for 1-glp and 2-glp profiles.

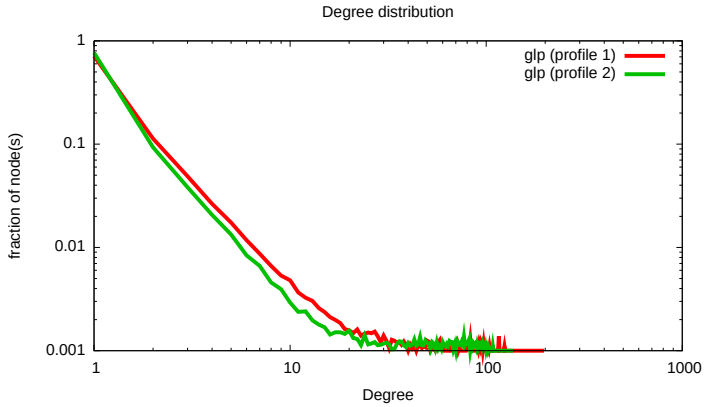


Figure 3.4: Degree distribution for 1-glp and 2-glp profiles (log log scale).

Assortativity. The assortativity of a graph expresses whether vertices are connected to vertices of similar degree or not. The higher the assortativity, the more vertices with the same degree are connected together. Since graphs of gnp and gnm profiles have the same degree with high probability, the assortativity of these graphs is high. In contrast, in glp graphs many vertices of high degrees (those from the largest 2-connected component) are connected to vertices with very low degree (the pending edges), thus decreasing the assortativity. However, 1-glp and 2-glp graphs are positively assortative which expresses that many similar nodes are connected together. For instance, it is likely that nodes of high degrees are neighbors.

Closeness centrality. The closeness centrality of a graph expresses how vertices are "close" to each other. From the definition of the closeness centrality $C_C(G)$, it is obvious that this value decreases with the average shortest path length since these two values are essentially the same. However, from the definition of the closeness centrality $C'_C(G)$, the correlation

with the average shortest path length is not obvious. Recall that $C'_C(u)$ for $u \in V$ equals $1/\bar{d}(u)$, and that the average values reported in Table 3.6 are those of the *centralization* and not only an average value of $C'_C(u)$ over all vertices $u \in V$. More precisely, we have $C'_C(G) = \frac{\sum_{v \in V \setminus v^*} C'_C(v^*) - C'_C(v)}{H}$, where v^* is the vertex of largest closeness centrality, and H is a normalizing factor. Consequently, a higher value of the closeness centrality indicates that the individual values of the closeness centrality of the vertices are very different from each other. For instance, in Figure 3.5, we observe that the closeness centrality of the vertices differ by at most 0.15 and that the four reported gnm profiles have the same shape (the shiftings are due to the different values of the average shortest path distances as reported in Table 3.6), thus explaining why these profiles have the same closeness centrality in Table 3.6.

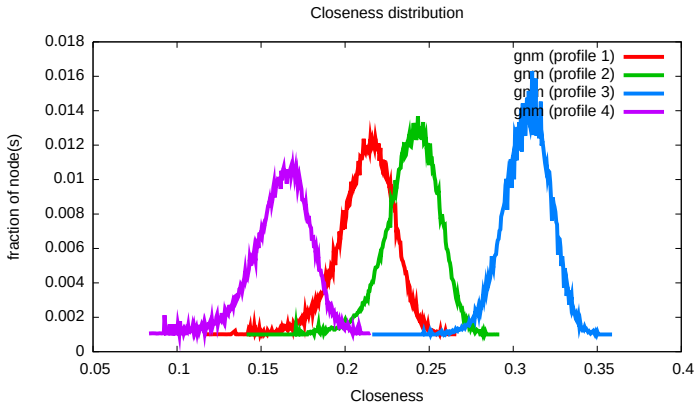


Figure 3.5: Closeness centrality distribution, C'_C, for gnm profiles.

In Figure 3.6, we report the closeness centrality of the vertices in glp profiles. As we can see from this figure, the behavior in the glp model is rather different. We observe that there is a peak value around 0.32 and, unlike gnm profiles, the distribution is not symmetric around the peak; there are many nodes on the left of the peak (spanning a rather wide range of closeness centrality values), while very few nodes have closeness centrality larger than the peak. This large deviation in the closeness centrality of the vertices explains why the closeness centrality of glp graphs reported in Table 3.6 is higher than that of gnm graphs. Observe also that the average value of the closeness centrality of the graphs with 1-glp or 2-glp profiles (0.37 and 0.36 respectively) is slightly larger than the values computed on CAIDA AS maps as reported in Tables 3.7 and 3.8 (pages 60 and 61), that is 0.27 for full maps and 0.29 for 2-connected components.

Vertex betweenness. The vertex betweenness expresses how a vertex is important for routing, that is how many shortest paths go through that vertex. Therefore, when removing a vertex of high betweenness from the graph, the average shortest path length may drastically increase, while the removal of a vertex with low betweenness value will impact few paths and hence have a low influence on the average shortest path length increase. In gnp and gnm graphs, all vertices have statistically the same importance and shortest paths are balanced in the graph, thus explaining the low value of the vertex betweenness centrality, while in glp graphs some vertices of very high degree play the role of hubs, as already observed above. This is confirmed by the distribution of the vertex betweenness reported in Figures 3.7 and 3.8.

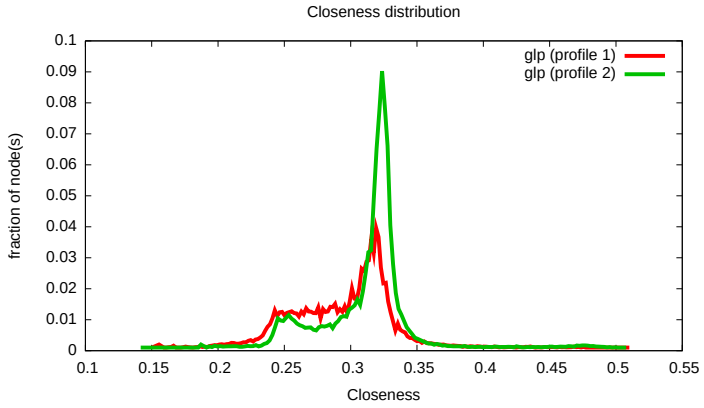


Figure 3.6: Closeness centrality distribution, C'_C, for glp profiles.

We observe that Figures 3.4 and 3.8 are essentially the same and that glp graphs have hub nodes.

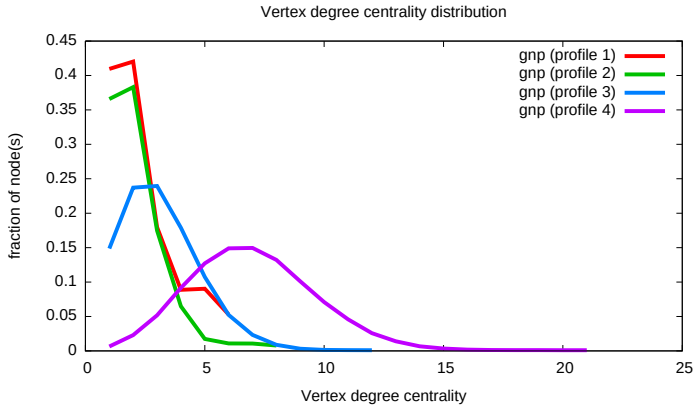


Figure 3.7: Vertex degree betweenness distribution for gnp profiles.

Edge betweenness. As for the vertex betweenness, the edge betweenness expresses how important is an edge for routing by reporting on the proportion of shortest paths using that edge. We have plotted in Figure 3.9 the edge betweenness of glp graphs. We observe that a small number of edges have a very high betweenness, meaning that not only some vertices of these graphs are hubs but also some edges are hub-like and belong to many paths. Therefore, the removal of some well chosen edges may drastically increase the average shortest path length and the overall routing. When considering the tiers structure of the Internet, many studies have reported that edges connecting Tier-1 nodes to each other belong to many shortest paths and so have a large betweenness.

Hyperbolicity. We have computed the hyperbolicity of all generated graphs as well as the statistical distribution of the 4-tuples hyperbolicity's distribution over 10^6 samples per graph, from which we have computed the *statistical hyperbolicity*. Recall that a simple heuristic

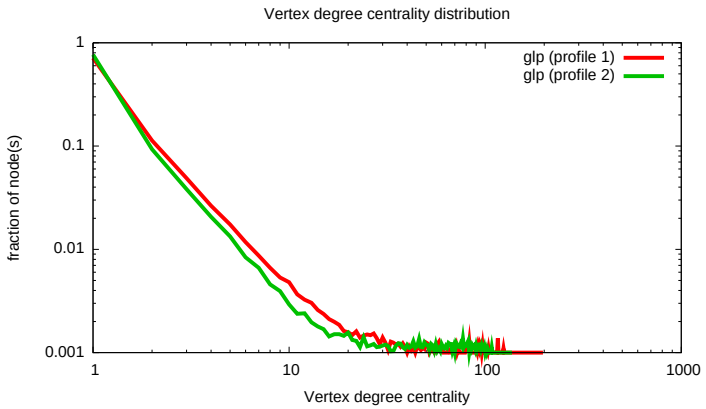


Figure 3.8: Vertex degree betweenness distribution for profiles 1-glp and 2-glp (log log scale).

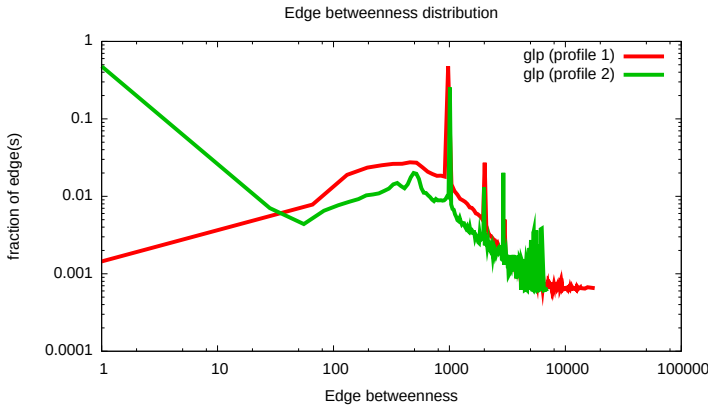


Figure 3.9: Edge betweenness distribution for profiles 1-glp and 2-glp (log log scale).

algorithm for obtaining a lower bound on the hyperbolicity of a graph is to take the maximum value of the hyperbolicity of a 4-tuple over a large number of randomly selected 4-tuples (here 10^6 randomly chosen 4-tuples). We can then compute for each profile the average value (line "Heuristic hyperbolicity" in Table 3.6 and column "Heur. Hyp." in Table 3.10) of the heuristic hyperbolicity of the graphs of that profile. From the hyperbolicity distribution of the 4-tuples, we also define and compute the *mean hyperbolicity* (line "Mean hyperbolicity" in Table 3.6 and column "Mean Hyp." in Table 3.10) of a graph as

$$\text{Mean-Hyp}(G) = 10^{-6} \sum_{h=0}^{2\hat{\delta}} Q[h/2] \cdot h/2, \tag{3.1}$$

where $Q[h/2]$ is the number of 4-tuples with hyperbolicity $h/2$, $\hat{\delta}$ is the heuristic value of the hyperbolicity, and 10^{-6} is the number of samples. When the exact distribution of the hyperbolicity is known, δ can be used instead of $\hat{\delta}$. In Tables 3.10 and 3.11, we report the average values we have obtained per graph profile.

Profile	Exact Hyperbolicity				Statistical Hyperbolicity	
	min $\leq$	Avg.	$\pm$ Dev.	$\leq$ max	Heur. Hyp.	Mean Hyp.
`3-gnp`	4.0 $\leq$	4.2	$\pm$0.25	$\leq$ 5.0	3.40	0.55
`4-gnp`	2.5 $\leq$	2.8	$\pm$0.24	$\leq$ 3.0	2.02	0.34
`5-gnp`	1.0 $\leq$	1.0	$\pm$0.00	$\leq$ 1.0	1.00	0.28
`1-gnm`	3.0 $\leq$	3.3	$\pm$0.25	$\leq$ 3.5	2.76	0.43
`2-gnm`	3.0 $\leq$	3.0	$\pm$0.00	$\leq$ 3.0	2.46	0.38
`3-gnm`	2.0 $\leq$	2.0	$\pm$0.11	$\leq$ 2.5	2.00	0.32
`4-gnm`	4.0 $\leq$	4.2	$\pm$0.25	$\leq$ 4.5	3.42	0.55
`1-glp`	1.5 $\leq$	1.7	$\pm$0.25	$\leq$ 2.0	1.10	0.15
`2-glp`	1.0 $\leq$	1.5	$\pm$0.16	$\leq$ 2.0	1.01	0.12

Table 3.10: Exact and statistical hyperbolicity of the generated graphs per profile.

Profile	Average distribution of 4-tuples hyperbolicity over 10^6 samples								
	0	0.5	1	1.5	2	2.5	3	3.5	4
`3-gnp`	29.4%	41.2%	21.1%	6.6%	1.5%	0.2%	0.0%	0.0%	
`4-gnp`	42.8%	46.7%	9.7%	0.7%	0.0%	0.0%			
`5-gnp`	48.4%	46.9%	4.7%						
`1-gnm`	35.8%	45.8%	15.6%	2.6%	0.2%	0.0%	0.0%		
`2-gnm`	39.1%	46.7%	12.7%	1.4%	0.1%	0.0%	0.0%		
`3-gnm`	44.9%	46.9%	7.9%	0.3%	0.0%				
`4-gnm`	29.3%	41.2%	21.1%	6.6%	1.5%	0.2%	0.0%	0.0%	0.0%
`1-glp`	71.2%	28.3%	0.4%	0.0%					
`2-glp`	76.0%	23.6%	0.4%	0.0%					

Table 3.11: Hyperbolicity distribution of the generated graphs per profile.

Using an appropriate tuning of the exact algorithm for computing the hyperbolicity of a graph, we were able to compute the hyperbolicity of several CAIDA AS maps, as reported

in the Table 3.12. Note that the computation of the hyperbolicity of the 2010 CAIDA map took 10 days of intensive computation on a PC equipped with 2 Intel Xeon processors with 4 cores (so 8 cores in total) each operating at 3.2GHz and with 64GB of RAM, and another 12 days for the 2011 CAIDA map. We have also determined the statistical distribution of the 4-tuples hyperbolicity's distribution over 10^7 samples, from which we have computed the statistical hyperbolicity, that is the heuristic hyperbolicity (Heur. Hyp.) and the mean hyperbolicity (Mean Hyp.) as defined in Equation (3.1). These results are useful for evaluating our generated graphs.

From Table 3.12, we know that the exact hyperbolicity of CAIDA AS maps is between 2 and 2.5. Due to the possible measurement bias in the construction of these maps, it is not possible to conclude whether the hyperbolicity is generally decreasing from 2.5 to 2 or not. Moreover, we observe that the statistical distribution of 4-tuples is roughly 2/3 with hyperbolicity 0, 1/3 with hyperbolicity 0.5, and less than 0.8% with hyperbolicity more than 1. We conclude that the number of 4-tuples inducing the exact hyperbolicity of the graphs is such a rare event that it cannot be observed using a statistical approach. Therefore, the heuristic hyperbolicity seems to better reflect the effective structure of the maps.

Hyperbolicity versus distances. From Table 3.6, we observe that the hyperbolicity of the generated graphs decreases with the average shortest path length, as well as with the density of the graph (which is correlated to the average shortest path length). Furthermore, the hyperbolicity is lower than both the radius and the average shortest path lengths but, more interestingly, we observe that the heuristic hyperbolicity is around half of the average shortest path length for gnp and gnm profiles and around one third for GLP graphs. Indeed, the hyperbolicity of a graph generally equals the hyperbolicity of its largest 2-connected component. Since graphs with profiles 1-glp and 2-glp have a small and dense core, it holds that these graphs have low hyperbolicity.

AS map name	Hyperbolicity	Avg. distribution of 4-tuples hyp. over 10^7 samples				Statistical Hyperbolicity	
		0	0.5	1	Number of 4-tuples with hyperbolicity 1.5	Heur. Hyp.	Mean Hyp.
2004/01/05	2.5	67.15%	32.19%	0.65%	41	1.5	0.17
2004/06/07	2	71.24%	28.42%	0.34%	16	1.5	0.15
2004/12/06	2.5	73.24%	26.53%	0.22%	10	1.5	0.14
2005/06/06	2.5	73.34%	26.46%	0.20%	5	1.5	0.13
2005/12/05	2.5	66.04%	33.76%	0.20%	2	1.5	0.17
2006/06/05	2.5	68.36%	31.17%	0.47%	20	1.5	0.16
2006/12/25	2.5	68.38%	31.34%	0.28%	3	1.5	0.16
2007/06/11	2	64.40%	35.02%	0.58%	24	1.5	0.18
2007/12/03	2.5	62.68%	36.55%	0.77%	21	1.5	0.19
2010/01/20	2	69.81%	30.04%	0.15%	1	1.5	0.19
2011/01/16	2	68.84%	30.60%	0.56%	1	1.5	0.16

Table 3.12: Exact and statistical hyperbolicity of some CAIDA AS maps.

3.3 Distributed computing models

We conclude this chapter by discussing an important issue that is related to both graph property measurements and the potential exploitation of these measurements by routing algorithms. Centralized solutions for property testing/measurements do not allow efficiently facing the dynamicity of the Internet networks (variations of both the topology and the traffic). Hence, it is natural to propose distributed or even localized solutions. That is, solutions where the nodes have *bounded memory* and *only local knowledge* of the network topology. The difference between *distributed* and *localized* algorithms is mainly relative to the time that is allowed to the nodes to exchange messages. Indeed, the less exchanges are allowed, the less the information can spread through the network and the more partial the knowledge the nodes have about the network is. However, the notion of distributed or localized algorithms is not well formalized since numerous models exist. Among other difficult points, one of the challenges of the EULER project it to determine relevant distributed models to be considered. That is, most of the properties that are relevant for routing schemes have very little chance to be computable with a pure local view of the topology. On the other hand, gathering all the information in one node and then spreading the result would not be realistic in the Internet (it would imply too much control traffic).

Some theoretical and over-simplified models were conceived for studying particular aspects of distributed protocols such as fault-tolerance, synchronism, locality, congestion, etc. In the model CONGEST (see the book of Peleg [115]) a network is represented by a graph whose nodes correspond to network processors and edges to inter-processor links. The communication is synchronous and occurs in discrete rounds. In each round every processor can send a message of size $O(\log n)$ through each of its outgoing links (where n is the number of nodes).

Some variations to the CONGEST model have been proposed. The general idea is to remove some restrictions (making it more powerful) in order to focus on some particular issues. In that spirit, Linial [94] introduced in a seminal paper the free model (also called LOCAL [115]). The only difference with the CONGEST model is that the restriction on the size of the messages is removed so that every vertex is allowed to send unbounded size messages in every round. By relaxing the constraint on the size of the messages, this model focuses on the issue of locality in distributed systems. For that reason, the answer to the question *What cannot be computed locally?* [87] (in the LOCAL model) appears to be a crucial one. In this context, Kuhn et al. [87] show that difficult problems like minimum vertex cover and minimum dominating set cannot be approximated well when processors can exchange arbitrary long messages during a bounded number of rounds. Grumbach and Wu [74] also use the CONGEST model. They are interested in *frugal computations*, i.e., computations where the total amount of information traversing each edge is $O(\log n)$. Their motivation was to understand the impact of locality. In fact, they show that for planar networks or networks of bounded degree, any first order logic formula can be evaluated frugally (in their setting). More recently, an important research effort is done to formalize localized models of computation [65, 69].

Since we need to face both locality and dynamicity issues, we are developing new techniques allowing to obtain global structural information from local (partial) views of the network. We have investigated the question of determining which graph properties can or cannot be computed using only local information. The distributed model we consider is a restriction

of the classical CONGEST (distributed) model and it is close to the simultaneous message model defined by Babai, Kimmel and Lokam [11]. More precisely, each of these n nodes only knows its own ID and the IDs of its neighbours and is allowed to send a message of $O(\log n)$ bits to some central entity, called the referee. We then investigate whether the referee is able to decide some basic structural properties of the network topology G or not. More formally:

Definition 1. *A* one-round protocol Γ *is a family* $(\Gamma^l_n, \Gamma^g_n)_{n\in\mathbb{N}}$*, where:*

- $\Gamma^l_n : \{1,\dots,n\} \times \mathcal{P}(\{1,\dots,n\}) \to \{0,1\}^*$ *is the* local function *of* Γ *for graphs of size* n*,*
- $\Gamma^g_n : (\{0,1\}^*)^n \to \{0,1\}^*$ *is the* global function *of* Γ *for graphs of size* n*.*

Given $G = (V = \{1,\dots,n\}, E)$*, the message vector of* Γ *on* G *is:*

$$\Gamma^l(G) = \Big(\Gamma^l_n(1, N_G(1)), \dots, \Gamma^l_n(n, N_G(n))\Big). \tag{3.2a}$$

The output of Γ *on* G *is:*

$$\Gamma(G) = \Gamma^g_n(\Gamma^l(G)). \tag{3.2b}$$

We define

$$|\Gamma^l(G)| = \max_{1\le i\le n} |\Gamma^l_n(i, N_G(i))|. \tag{3.2c}$$

Γ *is said to be* frugal *if:*

$$\max_{G \text{ graph of } n \text{ nodes}} |\Gamma^l(G)| = O(\log n). \tag{3.2d}$$

Notice that function Γ^l_n can be evaluated in any pair (i, N) where $i \in \{1,\dots,n\}$ and $N \subseteq \{1,\dots,n\}$. Then, $\Gamma^l_n(i, N)$ corresponds to the message sent to the referee by node i in a graph of n nodes when its neighborhood is N.

In [16], we show that even simple questions like "does G contain a square?" or "does G contain a triangle?" or "Is the diameter of G at most 3?" cannot be solved in general.

Theorem 2 ([16])**.** *There is no one-round frugal protocol allowing the referee to decide whether an arbitrary graph* G *contains a square or a triangle as a subgraph, or has diameter at most* 3*.*

On the other hand, the referee can decode the messages in order to have full knowledge of G when G belongs to graph classes like planar graphs, bounded-treewidth graphs and, more generally, bounded-degeneracy graphs.

Theorem 3 ([16])**.** *There exists a one-round frugal protocol allowing the referee to reconstruct graphs of bounded degeneracy.*

Questions related to the connectivity of arbitrary graphs are still open. We have continued our investigation by proposing variations of our distributed model. Following our framework, we have exhibited a hierarchy of problems and distributed models of computation. Without elaborating on the details, we have proposed four models of distributed computation, called SimAsync, SimSync, FreeAsync and FreeSync. SimAsync is exactly the model described in Definition 1. The remaining three models are more powerful since nodes are allowed to send sequentially their message to the referee and may have access to previous

messages. In particular, FreeAsync allows us to simulate asynchronicity of the network. We say that model Y is *more powerful* than model X, denoted by $X \leq Y$, if every problem that can be solved in X can also be solved in Y. Moreover, Y is *strictly more powerful* than X, denoted by $X < Y$, if $X \leq Y$ and there exists a problem $\mathcal{P}$ that can be solved in Y but not in X.

Theorem 4 ([15]). SimAsync $<$ SimSync $<$ FreeAsync $\leq$ FreeSync.

This allows us to identify certain network properties that can be decided with a pure local view (e.g., connectivity, existence of a square, etc.). However, for other properties we have shown that extra power (e.g., more communication rounds) is required in order to reach a conclusion [15]. The following table summarizes the results we got concerning connectivity. No (respectively, Ok) in row i and column j means that property i cannot (respectively, can) be computed in Model j.

	SimAsync	SimSync	FreeAsync	FreeSync
BFS	No	No	Ok in Bipartite graphs	Ok
Spanning-tree	?	?	?	Ok
Connectivity	?	?	?	Ok

The SimAsync Model is very constrained and so offers very limited power of computation to nodes. On the other hand, the FreeSync Model is very powerful but it requires too many computational resources and so is not realistic for practical applications. Hence, one crucial question for the EULER project is to design an intermediate localized model that would be realistic enough while allowing to compute basic properties such as connectivity, and possibly more elaborate properties.

Up to now, our study has focused on the case of static networks and it should be extended in order to include dynamicity. This requires the understanding of this dynamics and the introduction of suitable models of dynamicity. For this purpose, new algorithmic tools that take timing into account should be developed.

Chapter 4

Specification of experimental tools

This chapter provides a brief description of tools identified and/or developed within task T4.2 for performing experiments like the ones presented in Chapter 3. We refer to `http://wiki.sagemath.org/graph_survey` for a survey of existing software (tools and libraries) relevant to graph theory.

4.1 *Grph*

The objective of *Grph* is to provide researchers and engineers a suitable graph library for graph algorithms experimentation and network simulation. *Grph* is mainly a software library, but it also comes with a set of executable files for user interaction and graph format conversion; as such, it can be used autonomously. Performance and accessibility is the primary target of the *Grph* library. At every stage, it is designed to be efficient, both in terms of computation time and in terms of memory requirements. Its model considers mixed graphs composed of (un)directed simple- and hyper-edges. It allows to handle large dynamic graphs in the order of millions of nodes. *Grph* comes with a collection of base graph algorithms which is regularly augmented.

Grph uses the inherent parallelism of multi-core processors and multi-processor computers whenever possible, and performs caching of the results of graph algorithms in order to avoid re-computation (this is particularly useful in the context of network simulation). *Grph* integrates a bridge to native code (libraries or external applications) which allows the implementation of critical algorithms in C/C++. In order to do this, *Grph* resorts to on-the-fly compilation. Additionally, it provides a console-based interactive interface (shell), making experimentation more accessible. A list of some algorithms and bridges available in *Grph* is given in Table 4.1.

So far, most known users of the *Grph* library are part of INRIA and of the EULER project. *Grph* is distributed under the terms of a license defined by its contributors and is available for download. This license allows free usage and access to the source code. See `http://www-sop.inria.fr/mascotte/software/grph`

Grph enables its users to conduct experiments on large graph instances that cannot be dealt with in Java. In particular, it enables the creation and manipulation of graph instances composed of several millions of nodes. Most researchers and engineers willing to deal with graphs from a programmer's point of view resort to JUNG or JGraphT (Java), to Boost (C++) or Sage (Python). Because of its implementation language, *Grph* is in direct competition with JUNG and JGraphT; these projects were very active in the past but exhibit

very low activity during the last two years (i.e., no new release and almost no traffic on their mailing-list). In addition, in terms of comparative performance, benchmarks have shown that *Grph* clearly outperforms the competing graph libraries by a factor ranging from 1 to several thousands, depending on the operation considered.

Table 4.1: Some algorithms available in *Grph*.

Classification	**Methods**
Traversal	Breadth First Search (BFS), Depth First Search (DFS)
Shortest paths	BFS-based, Dijkstra, Bellman-Ford, Floyd Warshall
Clustering	Kernighan-Lin, number of triangles, chromatic number
Structural tests	Connected, complete, null, reflexive, regular, trivial, tree, simple graph, hypergraph, multigraph, acyclic, transitivity
Connected components	Set of connected components, bi-connected components, connected component containing a given vertex
Cuts	Test if two vertex sets form a cut in the graph, test if a set of vertex sets form a cut in the graph, set of edges of a given cut, cut size
Distances	Distance matrix, eccentricity, radius, diameter (distance matrix-based, 2-sweep approx), adjacency matrix, incidence matrix
Navigation	Adjacency and incidence matrices, degrees (in/out edge/vertex degrees), clustering coefficient, k-neighborhood, inclusion, equality, etc.
Centrality	Barycenter, closeness, edge-betweenness, endpoint-betweenness, path-betweenness, source-proximal betweenness, stress centrality, target proximal betweenness, vertex-betweenness, vertex (in/out) centrality
Topology generators	Inet, ring, chordal, BRITE, GLP, R-Mat, GNM, GNP, clique, bus, chain, star, grid, etc.
Unclassified	Topological sort, graph complement
Third-party algorithms	
libTW	Bridge to libTW (`http://www.treewidth.com/`) for the computation of tree width decomposition

A number of tools for statistics as well as for the computation of distributions, linear regressions, power-law exponents, etc. are also provided. A detailed list of algorithms is available at `http://www-sop.inria.fr/members/Luc.Hogie/grph/grph-algo.pdf`.

Our current and future plans include the design and development of a framework for the distribution of computational-intensive graph algorithms across a farm of servers as well as the addition of new property tests. These include bridging-centrality, localized bridging-centrality, group betweenness, vertex cover, growth, and hyperbolicity (exact, approximate, and heuristic algorithms). A preliminary implementation is already available but further

algorithmic improvements are sought.

4.2 Sage

Sage (see `http://www.sagemath.org`) aims to provide the arsenal mathematicans, researchers, and students need in order to perform calculations. The basic concept is to combine the power of many established software packages under a common Python-based interface. Even more than that, it provides powerful and unique algorithms in its own library. The mission of Sage is to "create a viable free open source alternative to Magma, Maple, Mathematica, and Matlab".

EULER's interest in Sage comes from the combination of a large collection of state-of-the-art graph algorithms, including bridges with `NetworkX`, a collection of algebra computation algorithms, and a simple interface with main linear programming solvers (GLPK, CBC, CPLEX, Gorubi). We use it both for research and educational purposes.

Sage was initially targeted to mathematicians and has now extended to a large community. It is used in computer science (e.g., algorithmics, combinatorics, graph theory, geometry, etc.) and in education. It is freely available and distributed under the General Public License. The community of users is large and distributions are downloaded approximately 6 500 times per month. The communities of both users and developers are animated through mailing-lists (about 50 messages per day for the developers community) and periodic workshops gathering developers and main users.

Through its large library of algorithms and user-interface, Sage constitutes an advance in the field of graph algorithms experimentation. By providing high-level building blocks, it facilitates the implementation of complex graph algorithms, thus allowing for rapidly testing a hypothesis. Furthermore, Sage allows for sharing the effort of optimizing the implementation among a community (e.g., one may propose a faster implementation of an existing algorithm).

Sage is mainly written in Python, Cython, and C by a worldwide community of developers (180 contributors for version 4.7.2 released on 2011-10-29, including 3 members of INRIA). It has around 300MB of source code. The user interface is in Python which makes it intuitive for beginners, while advanced users can also use Cython and C for implementing or optimizing demanding algorithms.

INRIA members have already contributed to more than 100 graph algorithms, added interfaces to specific linear programming solvers, and actively contributed to the improvement of the documentation. In particular, Nathann Cohen (former Ph.D. student of INRIA) wrote tutorials on graph theory and on linear programming in Sage, and he contributes to the french translation of the book "A tour of Sage".

Sage provides a set of algorithms for generating graphs and testing properties. An updated list of available topology generators and property tests is documented in the online reference manual of Sage (`http://www.sagemath.org/doc/reference/sage/graphs/graph.html`), and a list of graph theory software included in or interfaced with Sage is documented at `http://wiki.sagemath.org/graph_survey`.

However, one should be aware of Sage's limitations. In particular, the data structures chosen for simplifying both the implementation in Python and the ease of use (especially for beginners) are not fully optimized. Consequently, the memory consumption could be quite large, and some basic operations are slow compared to other software (e.g., *Grph*). Furthermore, the size of graphs that can be manipulated with Sage is currently limited to

65535 ($2^{16} - 1$) nodes. The main argument for this limitation is that the computation time of quadratic (or more) algorithms on larger graphs is enormous and requires a careful control of the memory usage. Also, one should prefer a dedicated implementation in C.

Within our current and future plans, in collaboration with task T4.2, we aim at contributing to Sage by proposing our implementations of some missing property tests and of new random graph generators for inclusion into future releases. This is important for the dissemination of the EULER's algorithmic results. In particular, observe that the evaluation of the hyperbolicity of some CAIDA AS maps summarized in Table 3.12 has been done using Sage development release 4.8alpha3. Our implementation will soon be proposed for inclusion in Sage.

Chapter 5

Conclusions

In this document, we have reported on activities performed in the context of Task T3.1 during the first year of EULER. We have presented the main (random) graph models that have been proposed in the literature in order to capture (among others) the characteristics of the Internet topology. We have also identified a rich set of graph properties; these are either properties that characterize Internet-like topologies or properties that are likely to facilitate routing and improve its performance. We have reported on a preliminary investigation of the value of several properties in graphs of relatively small size (i.e., 1 000 nodes) generated according to different existing models as well as in CAIDA AS maps. This investigation has benefited from several tools and libraries developed by the partners; this has revealed and exploited synergies with Task T4.2. Currently, we are extending our investigations to graphs of larger scale. Such an extension has become necessary during the numerical analysis reported in this document; often, working with small graphs has not provided much insight on the structure of significantly larger graphs. Testing (i.e., measuring) graph properties in larger graphs is an interesting and non-trivial computational problem in itself.

We remark that the investigations reported in Chapter 3 of this deliverable have been conducted in order to prepare the analysis of the routing schemes through extensive simulations and experimentation that will be conducted within tasks T3.3 and T4.3, respectively. More precisely, in order to evaluate the behavior of routing schemes with respect to particular topological properties, several sets of well-chosen graph topologies (satisfying specific properties) will be generated. Some will be Internet-like graphs and others will have very particular properties (e.g., bounded treewidth) and will be useful in verifying the theoretically predicted behavior. Furthermore, these graphs will be of various sizes. Topologies with few thousands of nodes will allow us to perform numerous simulations in order to achieve results with small statistical error; topologies with tens of thousands of nodes will be used for a more realistic validation of routing schemes, but computational and memory requirements will hinder the execution of a large number of simulations. On the other hand, experiments in testbed environments will be performed on topologies with at most 1 000 nodes. Therefore, we are interested in the generation of both large and small topologies that reflect the structure of the Internet.

In the discussion of our numerical and statistical investigation (Chapter 3), we have mentioned how the properties observed are expected to affect routing in general. Our current plan is to investigate this relation in more depth. Actually, in preliminary simulations with different routing algorithms (in particular, with greedy and compact routing algorithms) that

have not been reported yet, we have further observed that routing is performed efficiently (i.e., with low stretch, low congestion, and high fault-tolerance) in CAIDA maps as well as in random graphs of moderate size generated by models that are believed to capture core properties of the Internet (e.g., the GLP model). Still, the reason why the particular properties facilitate specific routing algorithms is not well-understood. This suggests one direction in the design of novel routing algorithms. In particular, such algorithms could exploit the values of particular graph properties in their specification. In this direction, we have initiated the study of simple computational models that define distributed and local mechanisms for property testing (see Section 3.3). Such distributed mechanisms could be used by routing algorithms in order to "learn" network characteristics and adapt their behavior to the topology and its changes. We expect that our work in this context will provide important input to Task T2.2. In particular, we plan to further study this distributed setting and focus on the evolution of some graph properties after (stochastic or predictable or known in advance) topological changes. This study includes the evolution of the treewidth of graphs subject to predetermined topological modifications, like those reported recently in [98].

Our holly grail in EULER is the development of a new graph model for the Internet that will encompass the advantages of the different models that have been proposed so far. Clearly, such an understanding of the Internet structure will have important benefits for routing.

Bibliography

[1] I. Abraham, C. Gavoille, A. V. Goldberg, and D. Malkhi. Routing in networks with low doubling dimension. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS)*, p. 75, 2006.

[2] I. Abraham and D. Malkhi. Name independent routing for growth bounded networks. In *Proceedings of the 17th Annual ACM Symposium on Parallel Algorithms (SPAA)*, pp. 49–55, 2005.

[3] L. Adamic and B. Huberman. Power-law distribution of the World Wide Web. *Science*, 287(5461):2115, 2000.

[4] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 171–180, 2000.

[5] W. Aiello, F. Chung, and L. Lu. Random evolution in massive graphs. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 510–519, 2001.

[6] R. Albert and A. Barabasi. Topology of evolving networks: local events and universality. *Physical Review Letters*, 85(24):5234–7, 2000.

[7] R. Albert and A.-L. Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.

[8] N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.

[9] O. Amini, F. Mazoit, N. Nisse, and S. Thomassé. Submodular partition functions. *Discrete Mathematics*, 309(20):6000–6008, 2009.

[10] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2):277–284, 1987.

[11] L. Babai, P.G. Kimmel, and S.V. Lokam. Simultaneous messages vs. communication. In *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, LNCS 900, Springer, pp. 361–372, 1995.

[12] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 623–632, 2002.

[13] A. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509, 1999.

[14] A.-L. Barabási. *Linked: The New Science of Networks*. Basic Books, 1st edition, May 2002.

[15] F. Becker, A. Kosowski, N. Nisse, I. Rapaport, and K. Suchan. Interconnection network with a shared whiteboard: Impact of (a)synchronicity on computing power. *arXiv:1109.6534*, 2011.

[16] F. Becker, M. Matamala, N. Nisse, I. Rapaport, K. Suchan, and I. Todinca. Adding a referee to an interconnection network: What can(not) be computed in one round. In *Proceedings of the 25th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, p. 7p, 2011.

[17] J.-C. Bermond, Z. Liu, and M. Syska. Mean eccentricities of de Bruijn networks. *Networks*, 30(3):187–203, 1997.

[18] G. Bianconi and A.-L. Barabasi. Competition and multiscaling in evolving networks. *Europhysics Letters*, 54(4):436, 2001.

[19] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, P10008, 2008.

[20] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998.

[21] H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996.

[22] H. L. Bodlaender and A. M. C. A. Koster. On the maximum cardinality search lower bound for treewidth. *Discrete Applied Mathematics*, 155(11):1348–1372, 2007.

[23] H. L. Bodlaender and D. M. Thilikos. Treewidth for graphs with small chordality. *Discrete Applied Mathematics*, 79(1-3):45–61, 1997.

[24] H. L. Bodlaender, J. van Leeuwen, R. B. Tan, and D. M. Thilikos. On interval routing schemes and treewidth. *Information and Computation*, 139(1):92–109, 1997.

[25] M. Boguna, R. Pastor-Satorras, and A. Vespignani. Cut-offs and finite size effects in scale-free networks. *European Physical Journal B*, 38:205–209, 2004.

[26] B. Bollobas. *Random Graphs*. Cambridge University Press, 2001.

[27] B. Bollobas and O. Riordan. The diameter of a scale-free random graph. *Combinatorica*, 24(1):5–34, 2004.

[28] P. Bonacich. Factoring and weighting approaches to clique identification. *Journal of Mathematical Sociology*, 2:113–120, 1972.

[29] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[30] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2008.

[31] G. Brinkmann, J.H. Koolen, and V. Moulton. On the hyperbolicity of chordal graphs. *Annals of Combinatorics*, 5:61-69, 2001.

[32] T. Bu and D. Towsley. On distinguishing between internet power law topology generators. In *Proceedings of the 21th IEEE Conference on Computer Communications (INFOCOM)*, pp. 37–48, 2002.

[33] M. Catanzaro, M. Boguna, and R. Pastor-Satorras. Generation of uncorrelated random scale-free networks. *Physical Review E*, 71(2):027103, 2005.

[34] D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys*, 38(1), 2006.

[35] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A recursive model for graph mining. In *Proceedings of the 4th SIAM International Conference on Data Mining (CDM)*, 2004.

[36] H. T.-H. Chan, A. Gupta, B. M. Maggs, and S. Zhou. On hierarchical routing in doubling metrics. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 762–771, 2005.

[37] Q. Chen, H. Chang, R. Govindan, and S. Jamin. The origin of power laws in internet topologies revisited. In *Proceedings of the 21st IEEE Conference on Computer Communications (INFOCOM)*, pp. 608–617. IEEE, 2002.

[38] Y. Chen and J. Flum. On parameterized path and chordless path problems. In *Proceedings of the 22nd Annual IEEE Conference on Computational Complexity (CCC 2007)*, pp. 250–263, 2007.

[39] V. Chepoi, F. Dragan, B. Estellon, M. Habib, and Y. Vaxès. Diameters, centers, and approximating trees of delta-hyperbolicgeodesic spaces and graphs. In *Proceedings of the 24th Annual Symposium on Computational Geometry (SOCG)*, pp. 59–68, 2008.

[40] F. Chung. Spectral Graph Theory. Regional Conference Series in Mathematics, No. 92, American Mathematical Society, 1997.

[41] F. Chung and L. Lu. The average distance in a random graph with given expected degrees. *Internet Mathematics*, 1(1): 91–113, 2003.

[42] F. Chung and L. Lu. Complex Graphs and Networks. Regional Conference Series in Mathematics, No.107, American Mathematical Society, 2006.

[43] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(66):66111, 2004.

[44] V. Colizza, A. Flammini, M. Serrano, and A. Vespignani. Detecting rich-club ordering in complex networks. *Nature Physics*, 2(2):110–115, 2006.

[45] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill, 2009.

[46] B. Courcelle and M. Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theoretical Computer Science*, 109(1&2):49–82, 1993.

[47] B. Courcelle and A. Twigg. Compact forbidden-set routing. In *Proceedings of the 24th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, LNCS 4393, Springer, pp. 37–48, 2007.

[48] C. Dangalchev. Residual closeness in networks. *Physica A*, 365:556, 2006.

[49] F. de Mongolfier, L. Viennot, and M. Soto. Autour du caractère arborescent de l'internet. In *Manifestation des Jeunes Chercheurs en Sciences et Technologies de l'Information et de la Communication*, 2010.

[50] J.-C. Delvenne, S. Yaliraki, and M. Barahona. Stability of graph communities across time scales. *Proceedings of the National Academy of Sciences*, 107, 2010.

[51] A. Don. *Indexation et navigation dans les contenus visuels: approches basées sur les graphes*. PhD thesis, Université de Bordeaux I, 2006.

[52] L. Donetti, M. A. Munoz. Improved spectral algorithm for the detection of network communities. In *Proceedings of 8th Granada Lectures*, AIP Conference Proceedings, 79, pp. 104–107, 2005.

[53] S. Dorogovtsev, A. Goltsev, and J. Mendes. Pseudofractal scale-free web. *Physical Review E*, 65(6):66122, 2002.

[54] S. Dorogovtsev, J. Mendes, and A. Samukhin. Structure of growing networks: Exact solution of the BarabasiAlberts model. *arXiv:cond-mat/0004434*, 2000.

[55] Y. Dourisboure. Compact routing schemes for generalised chordal graphs. *Journal of Graph Algorithms and Applications*, 9(2):277–297, 2005.

[56] Y. Dourisboure and C. Gavoille. Improved compact routing scheme for chordal graphs. In *Proceedings of the 16th International Conference on Distributed Computing (DISC)*, pp. 252–264, 2002.

[57] P. Duchon, N. Hanusse, E. Lebhar, and N. Schabanel. Could any graph be turned into a small-world? *Theoretical Computer Science*, 355(1):96–103, 2006.

[58] P. Erdős and A. Rényi. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Science*, 5:17–61, 1960.

[59] G. Ergun and G. Rodgers. Growing random networks with fitness *Physica A*, 303(1-2):261–272, 2002.

[60] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pp. 251–262, 1999.

[61] U. Feige, M. T. Hajiaghayi, and J. R. Lee. Improved approximation algorithms for minimum-weight vertex separators. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 563–572, 2005.

[62] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98):298–305, 1973.

[63] S. Fortunato and C. Castellano. Community structure in graphs. Chapter in *Encyclopedia of Complexity and System Science*, Springer, 2008.

[64] P. Fraigniaud. Greedy routing in tree-decomposed graphs. In *Proceedings of the 13th Annual European Symposium on Algorithms (ESA)*, LNCS 3669, Springer, pp. 791–802, 2005.

[65] P. Fraigniaud. Distributed computational complexities: are you Volvo-addicted or NASCAR-obsessed? In *Proceedings of the 29th Annual ACM Symposium on Principles of Distributed Computing, (PODC)*, pp. 171–172, 2010.

[66] P. Fraigniaud, E. Lebhar, and Z. Lotker. A doubling dimension threshold $\Theta(\log \log n)$ for augmented graph navigability. In *Proceedings of the 14th Annual European Symposium on Algorithms (ESA)*, LNCS 4168, Springer, pp. 376–386, 2006.

[67] P. Fraigniaud, E. Lebhar, and Z. Lotker. Recovering the long-range links in augmented graphs. *Theoretical Computer Science*, 411(14-15):1613–1625, 2010.

[68] P. Fraigniaud, E. Lebhar, and L. Viennot. The inframetric model for the Internet. In *Proceedings of the 27th IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1085–1093, IEEE, 2008.

[69] P. Fraigniaud, S. Rajsbaum, and C. Travers. Locality and Checkability in Wait-Free Computing. In *25th International Symposium on Distributed Computing (DISC)*, LNCS 6950, Springer, pp. 333–347, 2011.

[70] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[71] C. Gkantsidis, M. Mihail, and E. Zegura. The Markov chain simulation method for generating connected power law random graphs. In *Proceedings of the 5th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pp. 16–25, 2003.

[72] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. 2004.

[73] M. Gromov. Hyperbolic groups. *Essays in Group Theory*, 8:75–263, 1987.

[74] S. Grumbach and Z. Wu. Logical Locality Entails Frugal Distributed Computation over Graphs (Extended Abstract). In *Proceedings of 35th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, LNCS 5911, Springer, pp. 154–165, 2009.

[75] J.-L. Guillaume and M. Latapy. Bipartite graphs as models of complex networks. *Physica A: Statistical and Theoretical Physics*, 371(2):795–813, 2006.

[76] A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 534–543, 2003.

[77] S. L. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. *Journal of the Society for Industrial and Applied Mathematics*, 10(3):496–506, 1962.

[78] V. Havel. A remark on the existence of finite graphs. *Casopis Pest. Mat*, 80:477–480, 1955.

[79] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

[80] Z. Jiang and W. Zhou. Statistical significance of the rich-club phenomenon in complex networks. *New Journal of Physics*, 10:043002, 2008.

[81] C. Jin, Q. Chen, and S. Jamin. Inet: Internet topology generator. *Options*, (November 1997):1–18, 2000.

[82] C. Jin, Q. Chen, and S. Jamin. Inet technical report addendum. Technical report, 2002.

[83] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[84] Y. Kobayashi and K.-i. Kawarabayashi. Algorithms for finding an induced cycle in planar graphs and bounded genus graphs. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1146–1155, 2009.

[85] P. Krapivsky and S. Redner. Organization of growing random networks. *Physical Review E*, 63(6):66123, 2001.

[86] P. L. Krapivsky, S. Redner, and F. Leyvraz. Connectivity of growing random networks. *Physical Review Letters*, 85(4629), 2000.

[87] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally! In *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 300–309, 2004.

[88] R. Lambiotte, J.-C. Delvenne, and M. Barahona. Dynamics and modular structure in networks. *arXiv:0812.1770*, 2010.

[89] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78:46110, 2008.

[90] M. Latapy. Main-memory Triangle Computations for Very Large (Sparse (Power-Law)) Graphs. *Theoretical Computer Science*, 407(1-3):458-473, 2008.

[91] M. Latapy, C. Magnien, and N. Vecchio. Basic notions for the analysis of large twomode networks. *Social Networks*, 30(1):31–48, 2008.

[92] M. Latapy and P. Pons. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10(2):191–218, 2006.

[93] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research*, 11:985–1042, 2010.

[94] N. Linial. Locality in Distributed Graph Algorithms. *SIAM Journal of Computing*, 21(1):193–201, 1992.

[95] W. S. Lovejoy and C. H. Loch. Minimal and maximal characteristic path lengths in connected sociomatrices. *Social Networks*, 25(4):333 – 347, 2003.

[96] C. Magnien, M. Latapy, and M. Habib. Fast computation of empirically tight bounds for the diameter of massive graphs. *Journal of Experimental Algorithmics*, 13:10:1.10–10:1.9, 2009.

[97] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, et al. The Internet AS-level topology: three data sources and one definitive metric. *ACM SIGCOMM Computer Communication Review*, 36(1):17–26, 2006.

[98] B. Mans and L. Mathieson. On the treewidth of dynamic graphs. *arXiv:1112.2795*, 2011.

[99] J. Martin Hernandez, T. Kleiberg, H. Wang, and P. Van Mieghem. A comparison of topology generators with power law behavior. In *Proceedings of the 2007 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, pp. 484–493, 2007.

[100] A. Medina, I. Matta, and J. Byers. On the origin of power laws in Internet topologies. *ACM SIGCOMM Computer Communication Review*, 30(2):18–28,2000.

[101] J. Mendes, S. Dorogovtsev, and A. Ioffe. *Evolution of Networks: From Biological Nets to the Internet and WWW*. Oxford University Press, Oxford, 2003.

[102] R. Milo, N. Kashtan, S. Itzkovitz, M. E. J. Newman, and U. Alon. On the uniform generation of random graphs with prescribed degree sequences. *arXiv:cond-mat/0312028*, 2003.

[103] S. Narayanan. *The Betweenness Centrality of Biological Networks*. PhD thesis, Virginia Polytechnic Institute and State University, 2005.

[104] M. Newman. Assortative mixing in networks. *Physical Review Letters*, 89(20):208701, 2002.

[105] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167, 2003.

[106] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3), 2006.

[107] M. E. J. Newman. Random graphs with clustering. *Physical Review Letters*, 103(5):058701,1–4, 2009.

[108] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), 2004.

[109] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(2):1–17, 2001.

[110] M. E. J. Newman, D. J. Watts, and S. H. Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Sciences*, pp. 2566–2572, 2002.

[111] M. J. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27:39–54, 2005.

[112] N. Nisse, I. Rapaport, and K. Suchan. Distributed computing of efficient routing schemes in generalized chordal graphs. In *Proceeding of the 16th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, LNCS 5869, Springer, pp. 252–265, 2009.

[113] J. D. Noh and H. Rieger. Rethinking centrality: Methods and examples. *Social Networks*, 11:1–37, 1989.

[114] J. D. Noh and H. Rieger. Random walks on complex networks. *Physical Review Letters*, 92(11):118701, 2004.

[115] D. Peleg. Distributed computing: a locality-sensitive approach. *SIAM Monographs on Discrete Mathematics and Applications*, 2000.

[116] A. Pusch, S. Weber, and M. Porto. Generating random networks with given degreedegree correlations and degree-dependent clustering. *Physical Review E*, 77(1), 2008.

[117] B. Quoitin, V. Van den Schrieck, P. Franois, and O. Bonaventure. IGen: Generation of Router-level Internet Topologies through Network Design Heuristics. In *Proceedings of the 21st International Teletraffic Conference (ITC)*, 2009.

[118] E. Ravasz and A.-L. Barabasi. Hierarchical organization in complex networks. *Physical Review E*, 67(2):1–7, 2003.

[119] J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74:016110, 2006.

[120] N. Robertson and P. D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

[121] M. Rosvall and C. T. Bergstrom. Fast unfolding of communites in large networks. *Proceedings of the National Academy of Sciences*, 105:1118–1123, 2008.

[122] G. Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966.

[123] P. Salinger and P. Tvrdík. All-to-all scatter in de Bruijn and Kautz networks. *Computers and Artificial Intelligence*, 20(4), 2001.

[124] T. Schank and D. Wagner. Approximating clustering-coefficient and transitivity. *Journal of Graph Algorithms and Applications*, 9(2):265–275, 2005.

[125] J. Scott. *Social Network Analysis: A Handbook*. 2000.

[126] M. A. Serrano and M. Boguna. Tuning clustering in random networks with arbitrary degree distributions. *Physical Review E*, 72(3), 2005.

[127] P. D. Seymour and R. Thomas. Graph searching and a min-max theorem for tree-width. *Journal Combinatorial Theory, Series B*, 58(1):22–33, 1993.

[128] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[129] A. Slivkins. Distance estimation and object location via rings of neighbors. *Distributed Computing*, 19(4):313–333, 2007.

[130] R. E. Tarjan, Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 1(2):146-160, 1972.

[131] R. Taylor. Constrained switchings in graphs. *Combinatorial Mathematics*, 8:314–336, 1980.

[132] J. Tomasik and M.-A. Weisser. Internet topology on AS-level: model, generation methods and tool. In *Proceedings of the 29th IEEE International Performance Computing and Communications Conference (IPCCC)*, pp. 263–270, 2010.

[133] S. van Dongen. A cluster algorithm for graphs. RFC INS-R001, CWI, Netherlands, 2000.

[134] F. Viger and M. Latapy. Random generation of large connected simple graphs with prescribed degree distribution. In *Proceedings of the 11th International Conference on Computing and Combinatorics (COCOON)*, LNCS 3595, Springer, pp. 440–449, 2005.

[135] D. J. Watts and S. H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393(6684):440–442, 1998.

[136] J. Winick and S. Jamin. Inet-3.0: Internet topology generator. Technical report, Technical Report CSE-TR-456-02, University of Michigan, 2002.

[137] Y. Wu and C. Zhang. Hyperbolicity and chordality of a graph. *The Electronic Journal of Combinatorics*, 18(1), 2011, #43.

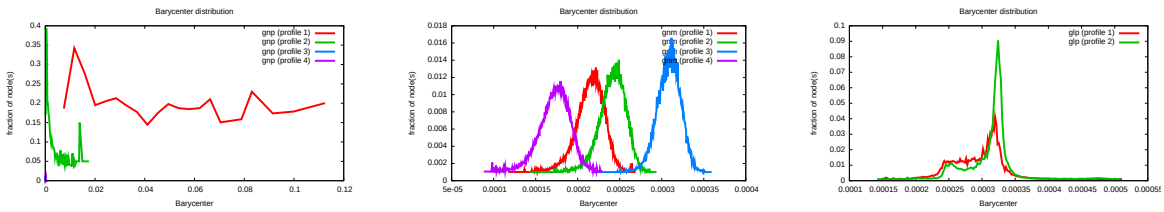
[138] S. Zhou and R. Mondragón. Accurately modeling the Internet topology. *Physical Review E*, 70(6):66108, 2004.

Appendix A

Observed distributions of graph properties

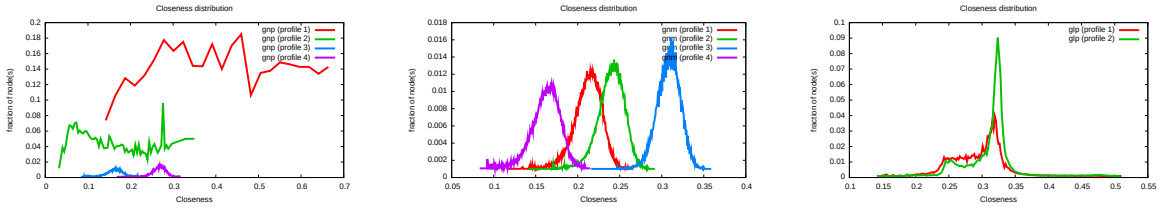
A.1 Random graphs

The following figures represent our experimental results on the generated graphs.



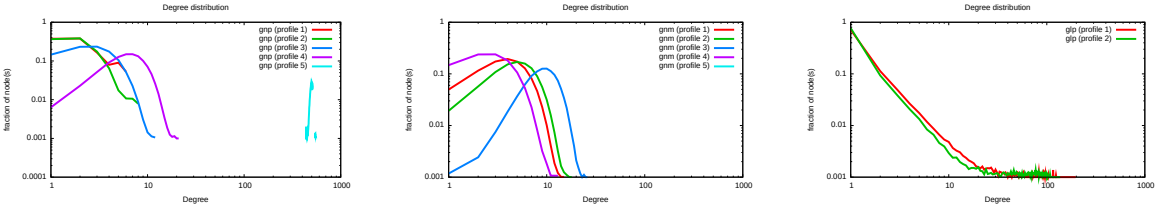
(a) gnp (b) gnm (c) glp

Figure A.1: Closeness centrality distribution (C_C).



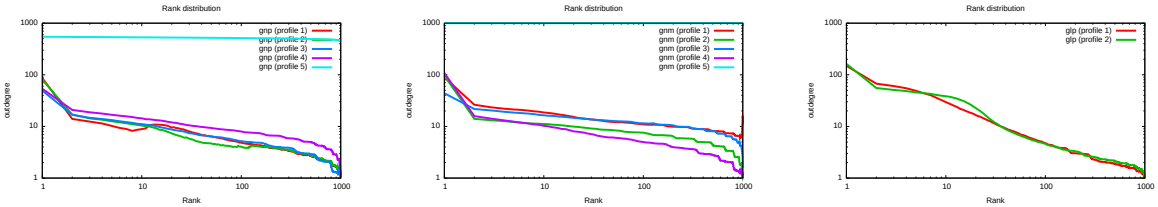
(a) gnp (b) gnm (c) glp

Figure A.2: Closeness centrality distribution, (C'_C).



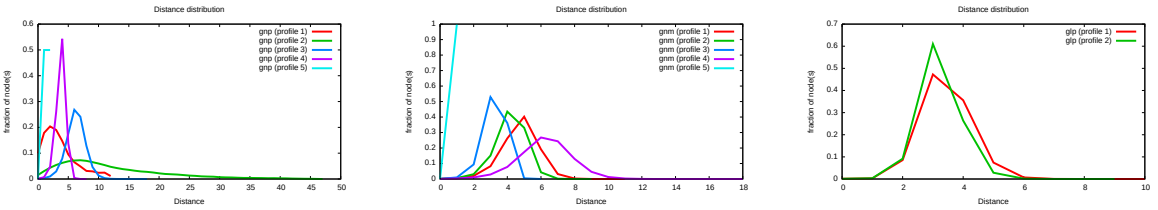
(a) gnp (b) gnm (c) glp

Figure A.3: Degree distribution (loglog scale).



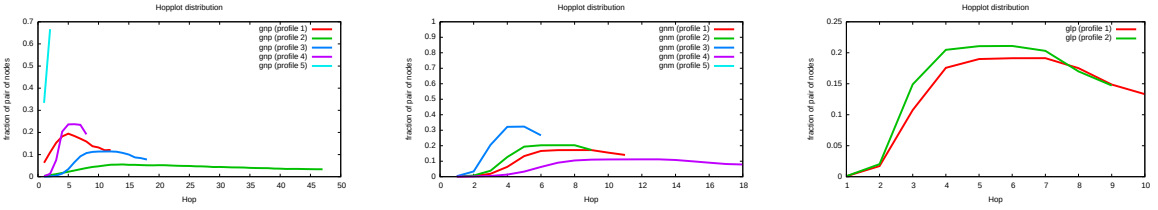
(a) gnp (b) gnm (c) glp

Figure A.4: Rank distribution (loglog scale).



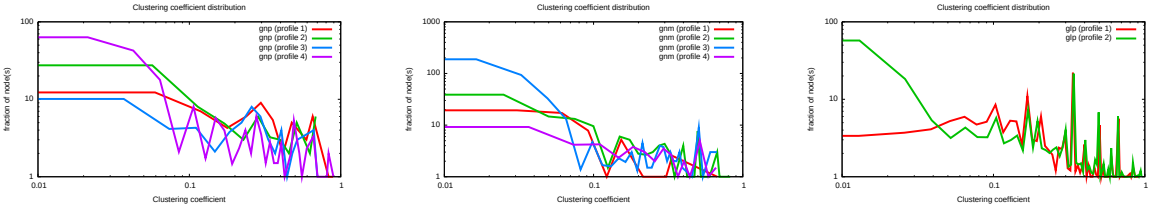
(a) gnp (b) gnm (c) glp

Figure A.5: Distance distribution.



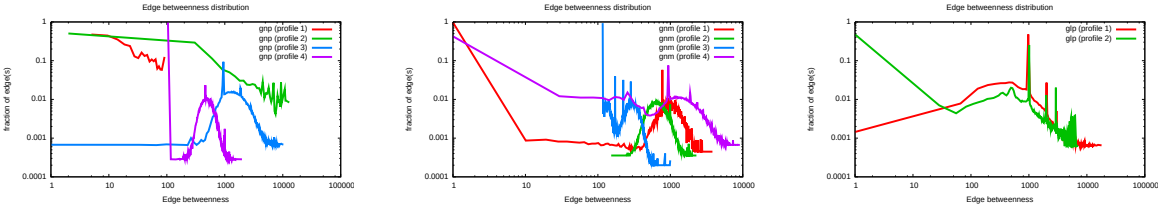
(a) gnp (b) gnm (c) glp

Figure A.6: Hotplot distribution.



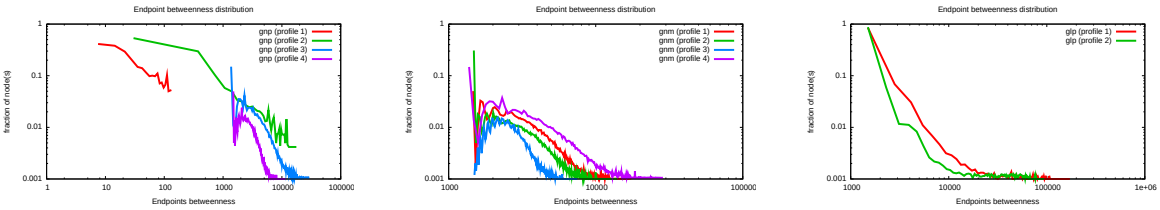
(a) gnp (b) gnm (c) glp

Figure A.7: Clustering coefficient distribution (loglog scale).



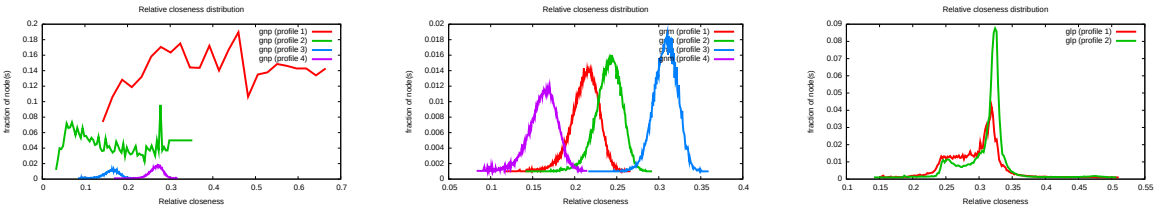
(a) gnp (b) gnm (c) glp

Figure A.8: Edge betweenness distribution (loglog scale).



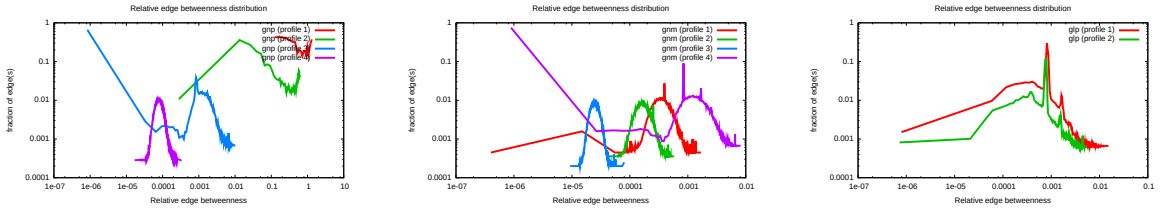
(a) gnp (b) gnm (c) glp

Figure A.9: Endpoint betweenness distribution (loglog scale).



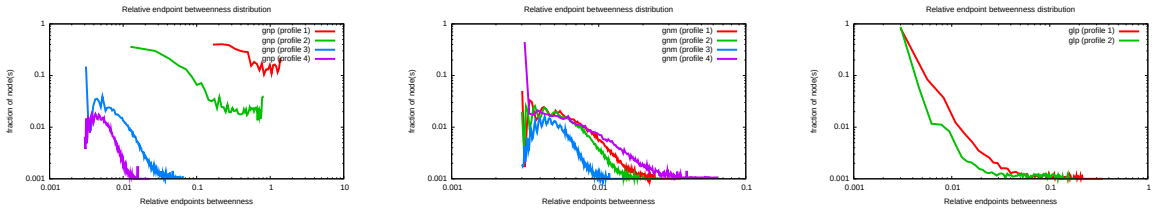
(a) gnp (b) gnm (c) glp

Figure A.10: Relative closeness distribution.



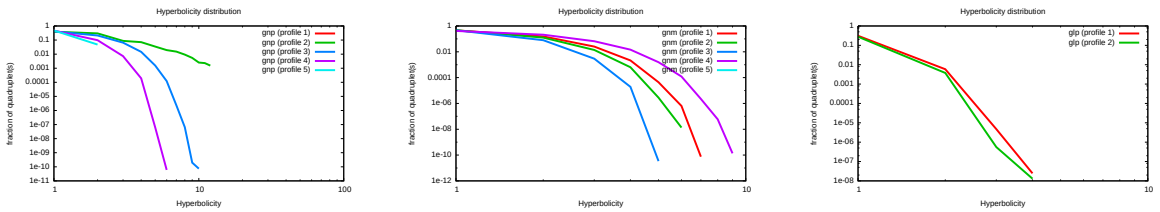
(a) gnp (b) gnm (c) glp

Figure A.11: Relative edge betweenness distribution (loglog scale).



(a) gnp (b) gnm (c) glp

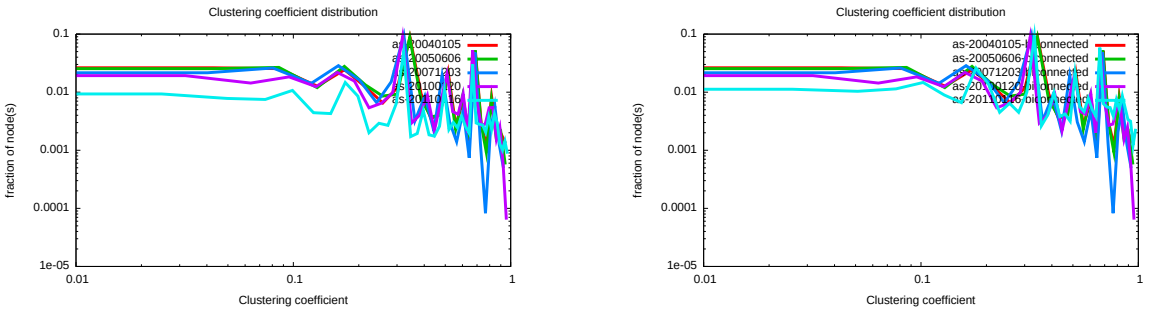
Figure A.12: Relative endpoint betweenness distribution (loglog scale).



(a) gnp (b) gnm (c) glp

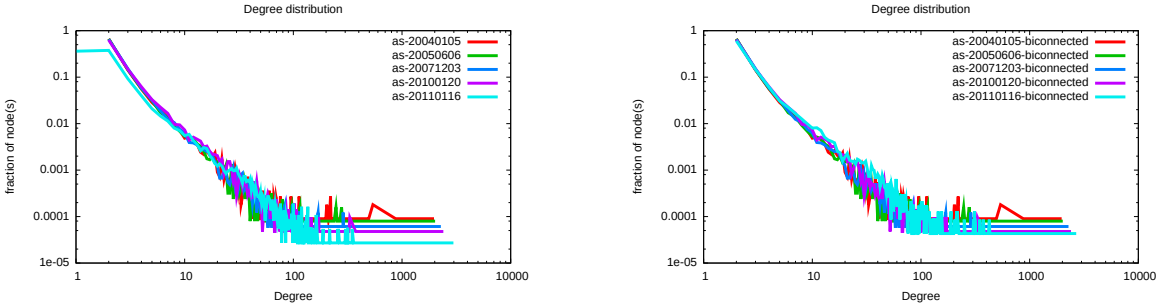
Figure A.13: Hyperbolicity distribution (loglog scale).

A.2 CAIDA AS maps



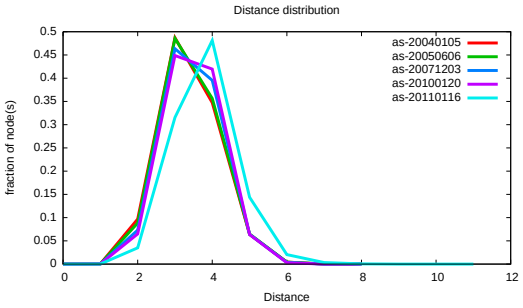
(a) Full maps (b) Largest 2-connected components

Figure A.14: Clustering coefficient distribution C'_C (log log).

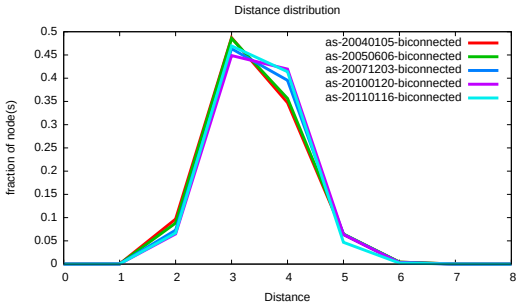


(a) Full maps (b) Largest 2-connected components

Figure A.15: Degree distribution (log log).

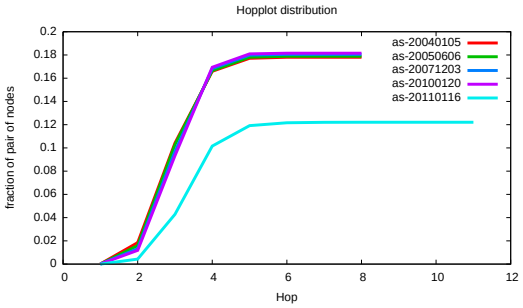


(a) Full maps

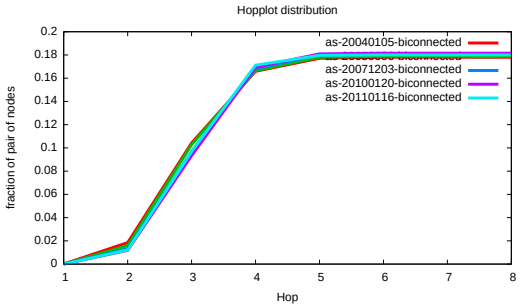


(b) Largest 2-connected components

Figure A.16: Distance distribution.

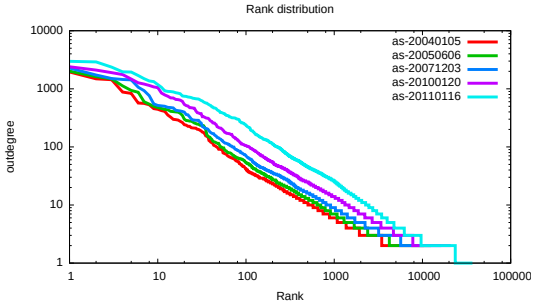


(a) Full maps

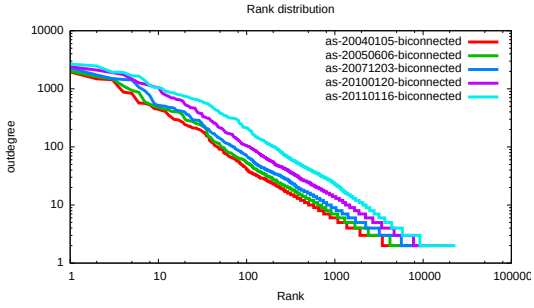


(b) Largest 2-connected components

Figure A.17: Hotplot distribution.



(a) Full maps



(b) Largest 2-connected components

Figure A.18: Rank distribution (log log).