# 11 Network Statistics

*Michael Brinkmeier and Thomas Schank*

Owing to the sheer size of large and complex networks, it is necessary to reduce the information to describe essential properties of vertices and edges, regions, or the whole graph. Usually this is done via *network statistics*, i.e., a single number, or a series of numbers, catching the relevant and needed information. In this chapter we will give a list of statistics which are not covered in other chapters of this book, like distance-based and clustering statistics. Based on this collection we are going to classify statistics used in the literature by their basic types, and describe ways of converting the different types into each other.

Up to this point *network statistic* has been a purely abstract concept. But one has quite good ideas what a statistic should do:

**A network statistic ...**

**... should describe essential properties of the network.**
This is the main task of network statistics. A certain property should be described in a compact and handy form. We would like to forget the exact structure of the underlying graph and concentrate on a restricted set of statistics.

**... should differentiate between certain classes of networks.**
A quite common question in network analysis regards the type of the 'measured' network and how to generate models for it. This requires the decision whether a generated or measured graph is similar to another one. In many situations this may be done by identifying several statistics, which are invariant in the class of networks of interest. Using these statistics an arbitrary graph can be tested for membership in a specific class, by determining its statistics and comparing them with some references.

**... may be useful in algorithms and applications.**
Some network statistics may be used for algorithms or calculations on the graph. Or they might indicate which graph elements have certain properties regarding the application.

To which degree a certain statistic fulfills one or more of these tasks obviously depends on the application and the network. Therefore we will not go into detail about the interpretation, and restrict ourselves to the description of types of statistics, common constructions, and several examples.

U. Brandes and T. Erlebach (Eds.): Network Analysis, LNCS 3418, pp. 293–317, 2005.

## 11.1 Degree Statistics

The most common and computationally easy statistic is the vertex degree. Depending on the underlying network and its application, it may be a simple measure for the strength of connection of a specific vertex to the graph, or – as in the case of indegrees – a measure for the relevance. But usually, instead of using this statistic directly, the main interest lies in the absolute number or the fraction of vertices of a given in-, out-, or total degree. It has been discovered that the distribution of degrees in many naturally occurring graphs significantly differs from that of classical random graphs.

In a classical undirected random graph $G_{n,p}$ the fraction of vertices of degree $k$ is expected to be

$$\binom{n-1}{k} p^k (1-p)^{n-1-k} \qquad \text{(Binomial distribution)}$$

if the number of vertices $n$ is small, or approximately

$$\frac{(np)^k}{k!} e^{-np} \qquad \text{(Poisson distribution)}$$

if $n$ is large. But in many natural graphs the degrees seem to follow a *power law*, i.e.

$$c\, k^{-\gamma} \qquad \text{with } \gamma > 0 \text{ and } c > 0.$$

The power law is a good example for a parameter elimination

by means of a functional description, as described in Section 11.7.1. Since the degree distribution can be described as $ck^{-\gamma}$, it suffices to determine the constant exponent $\gamma$. This can easily be done with linear regression of the log-log-plot of the distribution. The scaling constant $c$ is then dictated by the fact that the sum over all $k$ is either the number of edges (in the absolute case) or 1 (in the relative case). Of course, the exponent is only meaningful if the degree distribution has the appropriate form.

Examples for graphs and networks whose degree distribution seems to follow a power law include

- the actor collaboration graph ($\gamma \approx 2.3$) [40],
- the World Wide Web ($\gamma_{\text{in}} \approx 2.1, \gamma_{\text{out}} \approx 2.45/2.72$)[1][16, 40, 102],
- the power grid of the United States of America ($\gamma \approx 4$) [40],
- the Internet (router and autonomous systems) ($\gamma \approx 2.2$) [197].

More details about the power law, and models for generating graphs satisfying it, can be found in Chapter 13.

Experiments and measurements indicate that the power law and the resulting exponents are good statistics for the classification of graphs. They are especially useful for the decision whether a specific model generates graphs which are close to the naturally occurring ones.

[1] In the examples shown in Figures 11.4 and 11.5 near the end of this chapter, we have $\gamma_{\text{in}} \approx 2$ and $\gamma_{\text{out}} \approx 3.25$.

## 11.2 Distance Statistics

Another basic, but computationally more complex statistic is the distance between two vertices, defined as $d(u,v) = \min\{|P| \mid P \text{ is a path from } u \text{ to } v\}$. Arranging the distances leads to a $V \times V$-matrix $D$, whose columns and rows are indexed by the vertices of the graph, with

$$D = (d(u,v))_{u,v \in V}$$

containing the distance $d(u,v)$ in row $u$ and column $v$.

For arbitrary edge weights $w\colon E \to \mathbb{R}$ the problem of finding a shortest path is $\mathcal{NP}$-hard (see [240]). But for more special (but also more common) cases the distance can be calculated in polynomial time by solving the all-pairs shortest-path problem (APSP). Before we go into detail about the algorithmic aspects in Section 11.2.5 (see also Section 4.1.1), we will describe related statistics that are commonly used in the literature. As we will see, often not the distances themselves but more 'condensed' statistics are used.

### 11.2.1 Average or Characteristic Distance

The *average* or *characteristic distance* $\bar{d}$ is the arithmetic mean of all distances in the graph, i.e.,

$$\bar{d} := \frac{1}{|V|^2 - |V|} \sum_{u \neq v \in V} d(u,v).$$

For disconnected graphs we obviously have $\bar{d} = \infty$. Hence it might be useful to restrict ourselves to all connected pairs, leading to the *average connected distance*

$$\bar{d} := \frac{1}{k} \sum_{\substack{u \neq v \in V \\ 0 < d(u,v) < \infty}} d(u,v)$$

where $k$ is the number of connected pairs $u \neq v$. In abuse of notation we denote both statistics with $\bar{d}$, since usually only the second is meaningful.

If the distance-matrix $D$ is known, the average distance can be calculated in $\mathcal{O}(n^2)$ time.

### 11.2.2 Radius, Diameter and Eccentricity

By fixing one argument of the distance, the number of parameters of this statistic is reduced, leading to the *eccentricity* $\varepsilon(u)$ of a vertex $u$. This is the maximal distance of another node from $u$, i.e.,

$$\varepsilon(u) := \max\{d(u,v) \mid v \in V\}.$$

The *radius* $\operatorname{rad}(G)$ is the minimal eccentricity of all vertices, i.e.

$$\operatorname{rad}(G) := \min\{\varepsilon(u) \mid u \in V\}.$$

Maximizing over both arguments of the distance, one obtains the *diameter* $\operatorname{diam}(G)$ of a graph $G$ as the maximal distance between two arbitrary (connected) vertices, i.e.,

$$\operatorname{diam}(G) := \max\{d(u,v) \mid u,v \in V\}.$$

As with the average distances, the diameter and the eccentricity can be calculated from the distance matrix in $\mathcal{O}(n^2)$ time. The eccentricity of a single vertex may also be calculated via the single-source shortest-paths problem (SSSP), see Sections 4.1.1 and 11.2.5.

### 11.2.3 Neighborhoods

The *h-neighborhood* $\operatorname{Neigh}_h(v)$ *of a vertex* $v$ is the set of all vertices $u$ with distance less than or equal to $h$ from $v$, i.e.,

$$\operatorname{Neigh}_h(v) := \{u \in V \mid d(v,u) \le h\}.$$

The sizes of the $h$-neighborhoods form a parameterized statistic

$$N(v,h) := |\operatorname{Neigh}_h(v)|.$$

The *(absolute) hop plot* $P(h)$ eliminates the dependence on the vertex by assigning the number of pairs $(u,v)$ with $d(u,v) \le h$ to each parameter $h$, i.e.,

$$P(h) := \left|\left\{(u,v) \in V^2 \mid d(u,v) \le h\right\}\right| = \sum_{v \in V} N(v,h).$$

The *(relative) hop plot* $p(h)$ is the fraction of pairs with a distance less than or equal to $h$, i.e.,

$$p(h) := \frac{P(h)}{n^2} = \frac{1}{n^2} \sum_{v \in V} N(v,h).$$

The *average h-neighborhood size* $\overline{\operatorname{Neigh}}(h)$ is defined as

$$\overline{\operatorname{Neigh}}(h) := \frac{1}{n} \sum_{v \in V} N(v,h) = \frac{P(h)}{n} = np(h).$$

Again the absolute and relative hop plots, and the neighborhood sizes $N(v,h)$, can be calculated from the distance matrix in $\mathcal{O}(n^2)$ time and space.

The absolute hop plot of the Internet is examined by Faloutsos et al. in [197]. They observe that it follows a power law with an exponent around 4.7.

The $h$-neighborhoods that are necessary for the hop plot can be approximated using the ANF-algorithm of Palmer et al. presented in [462]. We will go into detail about that in Section 11.2.6.

### 11.2.4 Effective Eccentricity and Diameter

The *effective eccentricity* $\varepsilon_{\text{eff}}$ and the *effective diameter* $\text{diam}_{\text{eff}}$ are obtained via an interesting construction from the eccentricity and the diameter. The former measures the minimal distance at which a specified fraction $r$ of all nodes lies from a specific source $v$, i.e.,

$$\varepsilon_{\text{eff}}(v, r) := \min \{h \mid N(v, h) \geq rn\} .$$

The latter does the same without specifying a concrete source, i.e.,

$$\text{diam}_{\text{eff}}(r) := \min \left\{h \mid P(h) \geq rn^2\right\} = \min \{h \mid p(h) \geq r\} .$$

If $N$ and $P$ are known, both statistics can be calculated in $\mathcal{O}(\log \text{diam}(G))$ using binary search on $N(v, h)$ and $P(h)$.

The effective diameter and the effective eccentricity occur in [462] for a fixed value of $r = 0.9$.

### 11.2.5 Algorithmic Aspects

As mentioned at the beginning of this section, the problem of finding a shortest path between two vertices in a network with arbitrary edge weights $w\colon E \to \mathbb{R}$ is $\mathcal{NP}$-hard (see [240]). But if we restrict ourselves to networks without cycles of negative weight, the problem can be solved in polynomial time by well-known algorithms described in the following.

**The Path Algebra.** Assume that $G = (V, E)$ is a weighted graph (directed/undirected) without cycles of negative weight. Then the most general approach, which may be adapted to several other problems, for the calculation of the distance matrix is given by matrix multiplication over the *path algebra*.

Let $d_i(u, v)$ be the weight of a shortest path (i.e. a path of minimal weight) from $u$ to $v$ using at most $i$ edges. This implies

$$d_0(u, v) = \begin{cases} 0 & \text{if } u = v \\ \infty & \text{otherwise.} \end{cases}$$

Since a path with at most $i + 1$ edges either has at most $i$ edges or consists of a path of length $\leq i$ to a predecessor $v'$ of $v$ and the edge $(v', v)$, we have

$$d_{i+1}(u, v) = \min_{v' \in V} \left(d_i(u, v), d_i(u, v') + w(v', v)\right) .$$

Here it is important to keep in mind that $G$ does not contain cycles of negative weight.

The distance matrix may be calculated via the summation and multiplication of an adapted adjacency matrix $A$ over the *commutative semi-ring* $(\bar{\mathbb{R}}, \min, +)$, with $\bar{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$. The entries of $A$ are given by

$$a_{vu} = \begin{cases} 0 & \text{if } u = v \\ w(u,v) & \text{if there exists an edge from } u \text{ to } v \\ \infty & \text{otherwise} \end{cases}.$$

Explicitly, this means that summation is replaced by the minimum and multiplication by the addition. Then the distance matrix $D$ is $A^{\mathrm{diam}(G)}$, or, written as an iteration, $D$ is the limit of the iteration $D_{i+1} = D_i \cdot A$ and $D_0 = A$.

Fortunately it suffices to iterate $\mathrm{diam}(G)$-times, where the diameter is taken on the graph without weights. This is due to the fact that each simple path has at most length $\mathrm{diam}(G)$.

If $T(n)$ is the time needed for a multiplication of two $n \times n$-matrices, the iteration leads to a running time of $\mathcal{O}(T(n)\mathrm{diam}(G))$. Using the iteration $D_{2i} = D_i \cdot D_i$ instead, one obtains a running time of $\mathcal{O}(T(n)\log(\mathrm{diam}(G)))$.

Hence, the time needed for the calculation of the distance matrix via matrix multiplication is dominated by the time $T(n)$ needed for the multiplication of two $n \times n$-matrices. For the naive matrix multiplication we have $T(n) = \mathcal{O}(n^3)$. In [595] Zwick described an algorithm with $T(n) = \mathcal{O}(n^{2.575})$.

**Single-Source Shortest-Paths.** The distance matrix may be calculated by solving the single-source shortest-paths problem (SSSP) for all sources. Depending on the type of weight function, several algorithms are known.

- If the network is unweighted, i.e., $w(e) = 1$ for all edges $e \in E$, then SSSP can be solved via *breadth-first-search* in $\mathcal{O}(m)$ (see [133]).
- If $G$ has only edges of nonnegative weights then the algorithm of *Dijkstra* solves SSSP in $\mathcal{O}(n \log n + m)$ time when using Fibonacci Heaps (see [133] or Algorithm 4 in Section 4.1.1. ).
- If $G$ contains no cycles of negative weight, then the *Bellman-Ford* algorithm solves SSSP in $\mathcal{O}(nm)$ (see [133]).

The runtime of the algorithm of Dijkstra may be improved if more sophisticated data structures and strategies are used. In [547] Thorup describes an algorithm which calculates the SSSP for a given source on an undirected graph with positive integers as weights in $\mathcal{O}(m)$ time and space, leading to $\mathcal{O}(nm)$ time for the distance matrix. He uses an alternative strategy for the choice of the next node in the Dijkstra algorithm, following [152], and realizes the priority queue using buckets.

For further results about shortest paths, see [224, 225, 546, 548, 487, 12].

**All-Pairs Shortest-Paths.** An alternative to the use of matrix multiplications over the path algebra is the *Floyd-Warshall* algorithm for the solution of the all-pairs shortest-paths problem (APSP). Again the algorithm can only be applied to networks without cycles of negative weight. It requires $\mathcal{O}(n^3)$ time and can be found in textbooks (e.g., [133]) or in this volume (Algorithm 5 in Section 4.1.2).

Another approach to solving the APSP consists of running an SSSP algorithm for each of the $n$ vertices of the given graph. This leads to the following runtimes for APSP:

1. unweighted: $\mathcal{O}(nm)$
2. non-negative weights: $\mathcal{O}(nm + n^2 \log n)$
3. no cycles of negative weight: $\mathcal{O}(n^2 m)$

An improved solution to the APSP on a weighted graph without cycles of negative weight is achieved by Johnson's algorithm (details can be found in [133]). It first calculates the distances from an artifical source to all vertices in the graph using the Bellman-Ford algorithm. If the graph does not contain cycles of negative weight, it recalculates the edge weights using the results of the Bellman-Ford algorithm and then determines the distances of all pairs by calling the Dijkstra algorithm $n$ times. If the graph contains cycles of negative length it simply terminates. In total this leads to a runtime of $\mathcal{O}(n^2 \log n + nm)$.

### 11.2.6 ANF – The Approximate Neighborhood-Function

In [461] Palmer et al. described an algorithm for the estimation of the hop plot of an unweighted graph $G = (V, E)$ using approximative counting as introduced by Flajolet and Martin in [215]. The basis of the algorithm is the observation that the set $\text{Neigh}_h(u) = \{v \mid d(u, v) \leq h\}$ of nodes $v$ with distance at most $h$ from $u$ can be described as

$$\text{Neigh}_h(u) = \text{Neigh}_{h-1}(u) \cup \bigcup_{(u,v)\in E} \text{Neigh}_{h-1}(v).$$

Therefore it suffices to approximately count the number of elements in an iteratively increasing set. The main idea is to represent the sets $\text{Neigh}_h(u)$ by bitmasks, such that each bit represents a subset of vertices. Then the union corresponds to a logical `or` of the bitmasks. The resulting algorithm is shown as Algorithm 22.

In the last loop the estimate for $N(u, h)$ is calculated from the lowest position $R[u, h]$ of a 0-bit in the bitmask $B[u, h]$ starting at 0, i.e. $B[u, h]$ is of the form $\{0, 1\}^* 01^{R[u,h]}$. Following [215] the expected value of $R[u, h]$ is $\log(\varphi N(u, h))$ with $\varphi \approx 0.77351$.

If this procedure is repeated $z$ times with different position assignments $k[u]$, one can use the average $\bar{R}[u, h]$ over all lowest 0-bit positions to obtain an even better estimate with

$$N(u, h) = \frac{2^{\bar{R}[u,h]}}{0.77351\,(1 + 0.31/z)},$$

where the additional factor $(1 + 0.31/z)$ is caused by a bias of the expected value of $N[u, h]$ in this situation. Details can be found in [215].

Experiments conducted by Palmer, Gibbons and Faloutsos in [461] indicate a high accuracy of the estimations (less than 10% error for $z = 64$ trials) while

**Algorithm 22**: The ANF-Algorithm

**Input**: A graph $G = (V, E)$ and a natural number $r$
**Data**: Bitmasks $B[u, h]$ for each $u \in V$, $h = 1, \ldots, n$ of length $\log n + r$
**Output**: Estimations $N(u, h)$ for the number of nodes within distance $h$ of $u$ for each vertex $u$

**begin**
  $l \leftarrow \log n + r$
  **foreach** $v \in V$ **do**
    Set $k[v] = i$ with probability $\frac{1}{2^{i+1}}$ for $0 \leq i < l - 1$ and probability $\frac{1}{2^{l-1}}$ for $i = l - 1$
  **end**
  **foreach** $v \in V$ **do**
    Set position $k[v]$ of $B[v, 0]$ to 1
  **end**
  **for** $h \in \{1, \ldots, n\}$ **do**
    **foreach** $u \in V$ **do**
      $B[u, h] \leftarrow B[u, h - 1]$
    **end**
    **foreach** $(u, v) \in E$ **do**
      $B[u, h] \leftarrow B[u, h] \vee B[v, h - 1]$
    **end**
  **end**
  **foreach** $u \in V$ *and* $h \in \{0, \ldots, n\}$ **do**
    Let $R[u, h]$ be the lowest position of a 0-bit in $B[u, h]$
    $N(u, h) \leftarrow 2^{R[u,h]}/0.77351$
  **end**
**end**

providing higher speed and lower space requirements than other approximation methods based on random intervals and sampling. More exact results involving the standard deviation of the estimate can be found in [215].

## 11.3 The Number of Shortest Paths

A statistic closely related to the distance is the *number* $c(u, v)$ *of distinct shortest paths* between two vertices

$$c(u, v) := |\{P \mid P \text{ is a shortest path from } u \text{ to } v\}| .$$

Since the general APSP is $\mathcal{NP}$-hard, the same obviously holds for the calculation of the number of distinct shortest paths. But again it becomes polynomially solvable if the network does not contain cycles of negative or zero length. The $c(u, v)$ can be calculated by a modified Floyd-Warshall-algorithm (see Algorithm 23). In a similar way the Dijkstra algorithm may be modified if all edge weights are positive.

Closely related to $c(u, v)$ is the number of *disjoint* shortest paths. But this problem is known to be $\mathcal{NP}$-hard [346].

Alternatively, one can calculate the number of distinct shortest paths between two arbitrary vertices, using a specific edge, i.e.,

**Algorithm 23**: Counting the number of distinct shortest paths

**Input**: A graph $G = (V, E)$ with edge weights $w$ and without cycles of negative or zero length.

**Output**: The distances $d(u, v)$ and the number of distinct shortest paths between each pair $u, v$ of vertices

**begin**
  **foreach** $v \in V$ **do**
    **foreach** $u \in V$ **do**
      $d(u,v) \leftarrow \begin{cases} 0 & \text{if } u = v \\ w(u,v) & \text{if } (u,v) \in E \\ \infty & \text{otherwise} \end{cases}$
      $c(u,v) \leftarrow \begin{cases} 1 & \text{if } (u,v) \in E \\ 0 & \text{otherwise} \end{cases}$
  **foreach** $v' \in V$ **do**
    **foreach** $u \in V$ **do**
      **foreach** $v \in V$ **do**
        **if** $d(u,v') + d(v',v) = d(u,v)$ **then**
          $c(u,v) \leftarrow c(u,v) + c(u,v')c(v',v)$
        **else**
          **if** $d(u,v') + d(v',v) < d(u,v)$ **then**
            $d(u,v) \leftarrow d(u,v') + d(v',v)$
            $c(u,v) \leftarrow c(u,v')c(v',v)$
**end**

$$c(e) := |\{P \mid P \text{ is a shortest path including } e\}|.$$

Since each shortest path passing through the edge $e = (u, v)$ from $u'$ to $v'$ consists of a shortest path from $u'$ to $u$, the edge $e$, and a shortest path from $v$ to $v'$, the statistic $c(e)$ can be calculated from the $c(u, v)$:

$$c(e) = \sum_{d(u',v')=d(u',u)+1+d(v,v')} c(u',u)c(v,v').$$

## 11.4 Distortion and Routing Costs

Consider a spanning tree $T$ of $G = (V, E)$, i.e., an acyclic, connected subgraph of $G$ containing all vertices $v \in V$. Then the *distortion* $D(T)$ of $T$ is the average length of paths in $T$ between two adjacent vertices in $G$. More precisely

$$D(T) := \frac{1}{|E|} \sum_{\{u,v\} \in E} d_T(u, v)$$

where $d_T(u, v)$ is the distance from $u$ to $v$ in $T$.

The global *distortion* $D(G)$ *of* $G$ is the minimum distortion over all spanning trees of $G$, i.e.,

$$D(G) := \min \{D(T) \mid T \text{ is a spanning tree of } G\}.$$

The distortion is closely related to the *communication costs* of a spanning tree $T$ of a network, as introduced by Hu in [317]. In this setting a requirement $r_{u,v}$ is given for each pair of vertices. The communication cost of $T$ is given as the sum of all distances between vertices in $T$, multiplied by the requirement values. By setting $r_{u,v} = \frac{1}{m}$ if $\{u, v\} \in E$ and 0 otherwise, one obtains the distortion. In [333] Johnson et al. proved that the calculation of the minimal communication costs of a network with $r_{u,v} = 1$ for all pairs of vertices is $\mathcal{NP}$-hard.

## 11.5 Clustering Coefficient and Transitivity

The clustering coefficient introduced by Watts and Strogatz [573] in the year 1998 has become a frequently used tool in network analysis. For a node $v$ the clustering coefficient $c(v)$ is supposed to represent the likeliness that two neighbors of $v$ are connected. The clustering coefficient $C(G)$ of a graph is the average of $c(v)$ taken over all nodes. The latter seems to be a very popular index in network analysis.

In 2000 Barrat and Weigt [41] used an 'alternative formulation' for this average, claiming that their redefinition does not alter the 'physical significance' for certain generated small world networks. In the year 2002 the so-called transitivity was introduced by Newman, Strogatz and Watts [447], again as an alternative formulation of the clustering coefficient of a graph. It really turned out to be equivalent to the formulation of Barrat and Weigt, but not at all equivalent to the original clustering coefficient.

### 11.5.1 Definitions

We will define both indices in terms of triangles and triples of a node and a graph, respectively. Let $G$ be a simple and undirected graph. A *triangle* $\triangle = \{V_\triangle, E_\triangle\}$ is a complete subgraph of $G$ with exactly three nodes, see Figure 11.1. We use

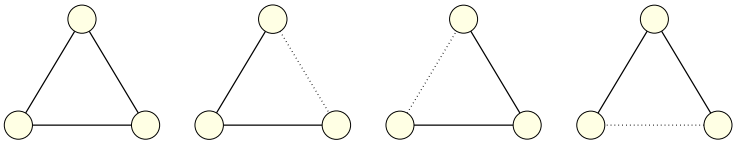

**Fig. 11.1.** A triangle and its three triples

$\lambda(G)$ for the number of triangles of a graph $G$. Accordingly we define $\lambda(v) = |\{\triangle \mid v \in V_\triangle\}|$ as the number of triangles of a node. Note that $\lambda(G) = 1/3 \sum_{v \in V} \lambda(v)$.

A *triple* is a subgraph of $G$ (not necessarily an induced subgraph) with three nodes and two edges. A triple is a *triple at node* $v$ if $v$ is incident with both edges of the triple. The number of triples at a node $v$ can be formulated in dependence of its degree $d(v)$ as

$$\tau(v) = \binom{\mathrm{d}(v)}{2} = \frac{\mathrm{d}(v)^2 - \mathrm{d}(v)}{2}.$$

The number of triples for the whole graph is $\tau(G) = \sum_{v \in V} \tau(v)$. In the above terms the *clustering coefficient* of a node $v$ with $\tau(v) \neq 0$ is defined as

$$c(v) = \frac{\lambda(v)}{\tau(v)}.$$

With $V' = \{v \in V \mid d(v) \geq 2\}$ we define the clustering coefficient of the whole graph as

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

Note that there is some variation in the literature in how nodes of degree less than two are handled, e.g., $c(v)$ is defined to be zero or one and included in the averaging process.

The *transitivity* for a graph is defined as

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

As there are exactly three triples in each triangle, see Figure 11.1, $3\lambda(G) \leq \tau(G)$ holds and consequently $T(G)$ is a rational number between zero and one.

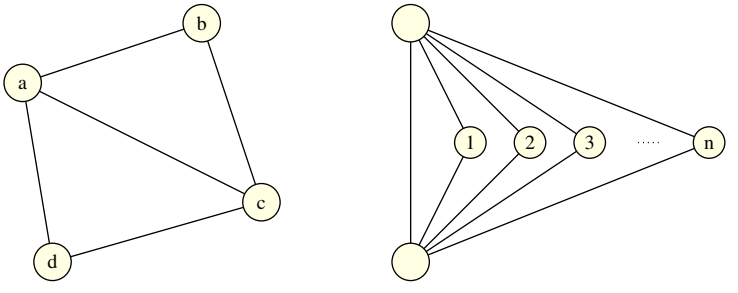


**Fig. 11.2.** On the left: Graph with clustering coefficients: $c(a) = c(c) = 2/3$, $c(b) = c(d) = 1$, $C(G) = \frac{1}{4}(2 + 4/3) \approx 0.83$ and transitivity $T(G) = 3 \cdot 2/8 = 0.75$. On the right: family of graphs where $T(G) \to 0$, $C(G) \to 1$ for $n \to \infty$.

### 11.5.2 Relation Between $C$ and $T$

The transitivity $T(G)$ was first introduced as an alternative formulation for the clustering coefficient $C(G)$ in [447]. The left hand side of Figure 11.2 shows a small graph where the two values differ. The right hand side shows a family of graphs where $T(G)$ approaches zero while $C(G)$ approaches one, for increasing $n$. The equation

$$T(G) = \frac{\sum_{v \in V'} \tau(v) c(v)}{\sum_{v \in V'} \tau(v)}$$

given by Bollobás and Riordan [68] shows a formal relation between the two indices: The transitivity is equal to the triple weighted clustering coefficient. Hence $T(G)$ equals $C(G)$, e.g., if all nodes have the same degree or all clustering coefficients are equal.

### 11.5.3 Computation

To compute the clustering coefficients we need to compute the number of triples $\tau(v)$ and the number of triangles $\lambda(v)$, for each node $v$. It is straight-forward to compute the number of triples $\tau(v)$ in linear time. This leaves us with the task of computing the number of triangles. For the transitivity it suffices to compute $\lambda(G)$ for the whole graph. It is not known whether there is an algorithm that is asymptotically faster in computing the triangles globally vs. locally.

The standard method is to iterate over all nodes and check whether the edge between any two neighbors is present. This algorithm has running time in $\mathcal{O}(nd_{\max}^2)$ where $d_{\max} = \max\{d(v) \mid v \in V\} = \Delta(G)$.

We assumed above that it is possible to test for edge existence in constant time. This could be done with a $n \times n$ matrix by using the 'indirection trick', see e.g. Exercise 2.12 in [4]. A more useful approach, requiring only linear space, is to use hashing. If each node is assigned a random bit vector of appropriate length, and the hash function uses a combination of two of these, we will get a randomized algorithm with expected testing time in $\mathcal{O}(1)$.

The second approach is to use matrix multiplication. Note that if $A$ is the adjacency matrix of graph $G$, then the diagonal elements of $A^3$ contain two times the number of triangles of the corresponding node. This gives an algorithm with running time in $\mathcal{O}(n^\gamma)$, where $\gamma$ is the matrix multiplication coefficient. It is currently known that $\gamma \leq 2.376$ [132].

Itai and Rodeh [321] proposed an algorithm with running time in $\mathcal{O}(m^{3/2})$. This is an improvement for sparse graphs, compared to the mentioned methods.

We will now discuss the algorithm of Alon, Yuster, and Zwick [26]. It improves the running time by using fast matrix multiplication, and still expresses only dependence on $m$ in the running time. If used with standard matrix multiplication ($\gamma = 3$) it will achieve the same bound as the algorithm of Itai and Rodeh.

The pseudocode of the algorithm is listed in Algorithm 24. Informally the algorithm splits the node set into low degree vertices $V_{\text{low}} = \{v \in V : \mathrm{d}(v) \leq \beta\}$

**Algorithm 24**: AYZ triangle algorithm

**Input**: Graph $G$ with adjacency array representation and hashed edge set
matrix multiplication parameter $\gamma$
**Output**: number of triangles $\lambda(v)$ for each node

```
1  β ⟵ m^((γ−1)/(γ+1))
2  for v ∈ V do
       λ(v) ← 0
       if d(v) ≤ β then
           V_low ← V_low ∪ {v}
       else
           V_high ← V_high ∪ {v}
3  for v ∈ V_low do
       for all pairs of neighbors {u, w} of v do
4          if edge between u and w exists then
5              if u, w ∈ V_low then
                   for z ∈ {v, u, w} do
                       λ(z) ← λ(z) + 1/3
6              else if u, w ∈ V_high then
                   for z ∈ {v, u, w} do
                       λ(z) ← λ(z) + 1
7              else
                   for z ∈ {v, u, w} do
                       λ(z) ← λ(z) + 1/2
8  A ← adjacency matrix of node induced subgraph of V_high
9  M ← A^3
10 for v ∈ V_high do
11     λ(v) ← λ(v) + M(i, i)/2 where i is index of v
```

and high degree vertices $V_{\text{high}} = V \setminus V_{\text{low}}$, where $\beta = m^{\gamma-1/\gamma+1}$. It then performs the standard method on the low degree nodes, and uses fast matrix multiplication on the subgraph induced by the high degree nodes.

**Lemma 11.5.1.** *Algorithm 24 computes the triangles $\lambda(v)$ for each node and can be implemented to have running time in $\mathcal{O}(m^{2\gamma/(\gamma+1)})$, or $\mathcal{O}(m^{1.41})$ .*

*Proof.* The correctness of the algorithm can be easily seen by checking the case distinction for different types of triangles consisting of exactly three (line 5), two (line 7), one (line 6), or zero (line 11) low degree nodes.

To prove the time complexity, we first note that the lines 1, 2, and 10 can clearly be implemented to run in linear time. We prove that the time required for the loop beginning at line 3 is in $\mathcal{O}(m^{2\gamma/(\gamma+1)})$. Line 4 requires a test for edge existence in constant time. We have discussed this in the context of the standard method above. The following tests in line 5, 6 and 7 are clearly in constant time. The bound follows then from $\sum_{v \in V_{\text{low}}} \binom{d(v)}{2} \leq m\beta$. The running time of line 8 is less than that of line 9. To bound line 9 we have to show that

$\mathcal{O}\left(n_{\text{high}}^{\gamma}\right) \subset \mathcal{O}\left(m^{2\gamma/(\gamma+1)}\right)$. Utilizing the hand shaking lemma $\sum_{v \in V} d(v) = 2m$, we get $n_{\text{high}}\beta \leq 2m$, which yields $n_{\text{high}} \leq 2m^{2/(\gamma+1)}$. □

### 11.5.4 Approximation

For processing very large networks, linear or sublinear running time is desired. We outline now how we can achieve approximations with sublinear running time using random sampling. A more detailed description can be found in [502].

Let $X_i$ be independent real random variables bounded by $0 \leq X_i \leq M$ for all $i$. With $k$ denoting the number of samples, and $\epsilon$ some error bound, Hoeffding's bound [299] states:

$$\Pr\left(\left|\frac{1}{k}\sum_{i=1}^{k} X_i - \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right]\right| \geq \epsilon\right) \leq e^{\frac{-2k\epsilon^2}{M^2}} \tag{11.1}$$

We assume that the graph is in an appropriate data structure in working memory. Specifically, we require that testing whether an edge between two nodes exists is in constant time. When giving the running time we regard the error bound $\epsilon$ and probability of correctness to be constants. To approximate the clustering coefficient for a node $v$, we estimate $\lambda(v)$ by checking for the required number (determined by Eq. 11.1) of neighbor pairs whether they are connected. This leads to an $\mathcal{O}(n)$-time algorithm for approximating $c(v)$ for all nodes. The clustering coefficient for the graph $C(G)$ can be approximated in a similar fashion. A short computation shows that it is sufficient to choose a random node $v$ and then two random neighbors for each sample. This gives an algorithm running in constant time. To approximate the transitivity one can proceed in a similar fashion, however, the nodes have to be chosen with weights corresponding to $\tau(v)$. This can be done in time $\mathcal{O}(n)$.

**Lemma 11.5.2.** *Consider the error bound $\epsilon$ and the probability of correctness to be constants. Then there exist algorithms that approximate the clustering coefficients for each node $c(v)$ and the transitivity $T(G)$ in time $\mathcal{O}(n)$. The clustering coefficient $C(G)$ can be approximated in time in $\mathcal{O}(1)$.*

## 11.6 Network Motifs

In molecular biology a functional domain of a molecule is called a *motif*. Thus motifs are building blocks for molecules on a higher level than atoms. Milo et al. [423] introduced the term *motifs* for networks, following the idea of building blocks on a higher level than nodes and edges. They enumerated the occurrences of small subgraphs in a network $G$. A connected subgraph that occurs in $G$ significantly more often than in a random network of same size and degree distribution is called a motif of $G$.

The authors considered several networks from biology (food webs, neuronal networks, gene regulation), engineering (electrical circuits), and also a domain of the World Wide Web. They looked for weakly connected subgraphs up to four nodes, and discovered for each of the networks distinctive motifs.

### 11.6.1 Algorithmic Aspects

A very basic algorithm to enumerate the occurrences of a subgraph with $k$ nodes is to check all $\binom{n}{k}$ possible combinations of $k$ nodes. The comparison itself can be done by permuting the $k$ nodes in $k!$ steps, where in each step all potential $2\binom{k}{2}$ edges are considered. To do this the number of possible automorphisms of the subgraph has to be known, see Section 12.1.

Unfortunately the original work of Milo et al. [423] does not describe the algorithm, and the 'supplementary materials' remain very vague in that respect, too. The 'supplementary materials', some of the considered networks, and an implementation of the algorithm can be downloaded from an author's website.[2] However, it has been reported that the published results produced by that program are incorrect.[3]

Algorithm 24 of Alon, Yuster, and Zwick [26] for enumerating all triangles in an undirected graph can be modified to count directed and connected subgraphs of three nodes without incurring additional costs in the asymptotic running time.

In [363] the basic idea of the triangle counting algorithm of Alon, Yuster, and Zwick was refined by Kloks, Kratsch and Müller to achieve an algorithm for counting all $K_4$'s in a graph in time $\mathcal{O}(m^{\frac{\gamma+1}{2}})$. Here $\gamma \leq 2.376$ [132] is the matrix multiplication coefficient. Further, for the undirected case, they were able to derive an algorithm that counts all subgraphs with at most four nodes in time $\mathcal{O}(n^\gamma + m^{\frac{\gamma+1}{2}})$. It is not known whether this can be generalized to the directed case whilst obeying the same bound in running time.

## 11.7 Types of Network Statistics

Browsing through the literature, and by examination of the previous examples, one can identify four basic types of statistics which can be described by two pairs of exclusive attributes:

**single-valued vs. distribution** and **global vs. local**

Even though the attributes are intuitively clear, we give a formal description of the four resulting types. Let $\mathcal{G}$ be a class of graphs (e.g., (un-)weighted, (un-)directed, (un-)connected etc.), $P$ a set of parameters, and $Y$ a set of *values*. Usually we have, e.g., $P, Y = \mathbb{N}, \mathbb{Z}, \mathbb{R}$ or products of them. Furthermore let $X_G$ be a set of not specified *graph elements* in $G$, which may consist of vertices, edges, subgraphs, paths etc.

**Global (Single-valued) Statistics.** A *global (single-valued) statistic* $\gamma$ assigns a single value $\gamma_G \in Y$ to each graph $G \in \mathcal{G}$.

Examples: Number of vertices/edges, diameter (Section 11.2.2), clustering coefficient of a graph (Section 11.5), edge- and vertex-connectivity (Chapter 7).

[2] `http://www.weizmann.ac.il/mcb/UriAlon/`
[3] Falk Schreiber, personal communication regarding `MFinder 1.1`

**Global Distributions.** A *global distribution* $\Gamma$ assigns a map $\Gamma_G \colon P \to Y$ to each graph $G \in \mathcal{G}$. We usually denote the value by $\Gamma_G(t)$, where $t$ is the parameter. For example for $P = \mathbb{N}$ a global distribution $\Gamma_G$ is a sequence $(\Gamma_G(0), \Gamma_G(1), \dots)$ of values for each graph in $\mathcal{G}$.

Examples: absolute/relative distribution of in/out-degrees (Section 11.1), hop plot (Section 11.2.3).

**Local (Single-valued) Statistics.** A *local (single-valued) statistic* $\lambda_G$ assigns a single value $\lambda_G(x) \in Y$ to a certain graph element $x$ of a given graph $G \in \mathcal{G}$, where $x$ may be a vertex, an edge, a set of vertices or edges, a subgraph, or whatever may be seen as a local graph element.
More formally $\lambda_G \colon X_G \to Y$ is a map from a set $X_G$ of graph elements in $G$ to the set of values $Y$.

Examples: in/out-degree (Section 11.1), weight/capacity/length of edges, distance (Section 11.2), clustering coefficient of a vertex (Section 11.5) , various centralities (Chapters 3 and 5).

**Local Distributions.** A *local distribution* $\Lambda$ assigns a map $\Lambda_G \colon X_G \times P \to Y$ from the cartesian product of the appropriate set of graph elements and a parameter set $P$ to $Y$ to each graph $G \in \mathcal{G}$. As in the global case we write $\Lambda_G(x, t)$ for the value at $x$ with parameter $t$.

Examples: Sizes of neighborhood (Section 11.2.3), effective eccentricity and diameter (Section 11.2.4).

Since we already used the term *distribution* for multiple valued statistics, we will adapt our nomenclature slightly. The term *statistic* refers to a single-valued, and *distribution* to multiple-valued, network statistics. Nonetheless both types are included in the notion *network statistics*.

### 11.7.1 Transformation of Types of Statistics

The four types are not isolated from each other. In the literature several techniques can be found to transform one into another. In fact, it usually is possible to classify network statistics by their 'underlying' statistic, from which they are deduced in one or more of the constructions we are going to describe. A rough scheme of the possible transformations is presented in Figure 11.3. A more detailed description is given in the following sections.

Of course, all constructions may be *composed*, e.g., after going from a global statistic to a local distribution, one may construct a local statistic from the result. In this section we restrict ourselves to the operations which seem to be more or less independent of each other. In the literature and in the examples below, many network statistics are obtained by the application of two or more of the steps described here.

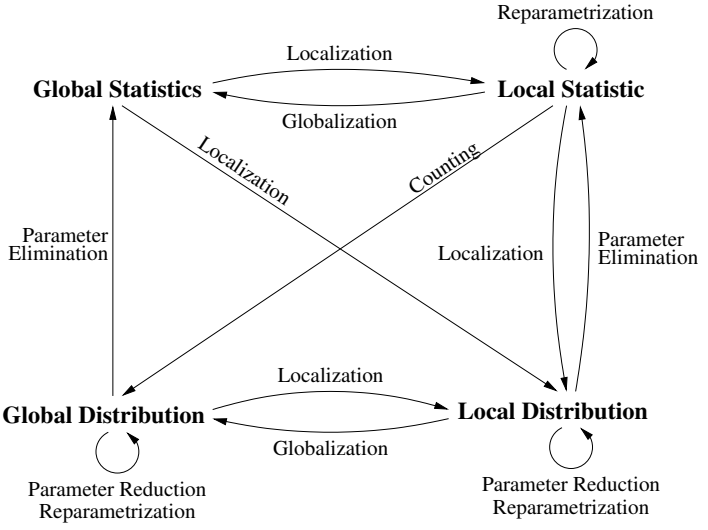


**Fig. 11.3.** Transformations of types of statistics

Some of the operations do not occur in the literature (at least as far as we know). They are mainly presented for completeness, and because they seem to be very natural possibilities. Furthermore we do not discuss the intuition behind the transformations, since this depends on the concrete application and interpretation of the values. We only give a formal description of the techniques.

**Globalization.** As the name says, globalization eliminates the dependence of a local statistic or distribution on graph elements. This is often done by calculating, choosing, or constructing a single value $\gamma_G$ from the values $\lambda_G(x)$ for all graph elements $x \in X_G$. The most common examples are:

– the maximum
$$\gamma_G := \max\{\lambda_G(x) \mid x \in X_G\},$$
– the minimum
$$\gamma_G := \min\{\lambda_G(x) \mid x \in X_G\},$$
– the summation
$$\gamma_G := \sum_{x \in X_G} \lambda_G(x),$$
– and averaging
$$\gamma_G := \frac{1}{|X_G|} \sum_{x \in X_G} \lambda_G(x).$$

For distributions, the same constructions may be applied parameterwise, e.g.,
$$\Gamma_G(t) := \max_{x \in X_G} \{\lambda_G(x,t)\}.$$

Examples are the average (connected) distance (Section 11.2.1), the diameter and the radius (Section 11.2.2), the relative and absolute hop plot (Section 11.2.3), and the global distortion (Section 11.4).

**Counting.** To transform a local statistic $\lambda$ into a global distribution $\Gamma$, one may count the number of graph elements $x$ such that $\lambda_G(x)$ lies in a specific range of values. For discrete statistics (e.g., $\lambda_G(x) \in \mathbb{N}, \mathbb{Z}$), the absolute number of occurrences of the value $t$, i.e.,

$$\Gamma_G(t) := |\{x \in X_G \mid \lambda_G(x) = t\}|,$$

or the relative number of occurrences, i.e.

$$\Gamma_G(t) := \frac{|\{x \in X_G \mid \lambda_G(x) = t\}|}{|X_G|}$$

are very common. Similarly, for continuous statistics (e.g., $\lambda_G(x) \in \mathbb{R}$), the absolute or relative number of elements with $\lambda_G(x) \leq t$, i.e.,

$$\Gamma_G(t) := |\{x \in X_G \mid \lambda_G(x) \leq t\}| \qquad \text{or}$$

$$\Gamma_G(t) := \frac{|\{x \in X_G \mid \lambda_G(x) \leq t\}|}{|X_G|}$$

is widely used.

Examples are the absolute, relative, and complementary cumulative degree distributions (Section 11.1), the absolute and relative hop plot (Section 11.2.3).

**Parameter Elimination and Reduction.** Perhaps the widest class of type changes is the elimination of parameters. In some respect nearly all types of changes can be interpreted in this way. But we restrict ourselves to parameters, i.e., values which are not directly related to the examined network.

Similar to the transformation from local to global statistics and distributions, one may calculate a single value $\lambda_G$ from a sequence/map of values given by a distribution $\Lambda_G$. But instead of using the graph elements as variables, one uses the parameter of the distribution. Examples are

– the maximum

$$\lambda_G(x) := \max\{\Lambda_G(x,t) \mid t \in P\},$$

– the minimum

$$\lambda_G(x) := \min\{\Lambda_G(x,t) \mid t \in P\},$$

– summation

$$\lambda_G(x) := \sum_{t \in P} \Lambda_G(x,t),$$

– averaging (only sensible if $P$ is finite)

$$\lambda_G(x) := \frac{1}{|P|} \sum_{t \in P} \Lambda_G(x,t),$$

– and the projection onto a certain parameter $t_0 \in P$

$$\lambda_G(x) := \Lambda_G(x,t_0).$$

For single-valued statistics one simply has to neglect the argument $x$. The eccentricity (Section 11.2.2) is an example for this transformation.

If a local or global distribution has more than one parameter, i.e., if its set of parameters is the cartesian product $P = P' \times P''$, one may use only one parameter for the calculations, leading to a *parameter reduced* distribution, e.g.,

$$\Lambda'_G(x,t') := \max_{t'' \in P''} \{\Lambda_G(x,t',t'')\}.$$

If the local or global distribution allows the description as a function of the parameters, some may be eliminated using a different technique. Assume that the local distribution $\Lambda_G(x,t)$ can be described by a function $f_r$ with parameter $r(x)$, i.e., $\Lambda_G(x,t) = f_{r(x)}(t)$. Then the parameters $r(x)$ form a local statistic which indirectly describes the local distributions. If the graph element $x$ is omitted, the same technique can be applied for the construction of a global statistic from a global distribution.

An example for this transformation is the power law exponent (Section 11.1). The degree distributions are reduced to a single value.

**Reparametrization.** Instead of eliminating a parameter, one can change it. For a local distribution $\Lambda_G(x,t)$ with parameter $t \in P$, we usually can write the reparameterized distribution $\tilde{\Lambda}_G$ with parameter $r \in P'$ as

$$\tilde{\Lambda}_G(x,r) := f_r(x, (\Lambda_G(x,t))_{t \in P})$$

where $f_r$ is a function with parameter $r$, using all values of $\Lambda_G(x,t)$ for a given $x$.

For example one may chose $P = P' = \mathbb{N}$,

$$f_r(x; y_0, y_1, \dots) = \max\{y_r, y_{r+1}, \dots\}$$

leading to

$$\tilde{\Lambda}_G(x,r) := \max\{\Lambda_G(x,t) \mid t \geq r\}.$$

Another example is $f_r(x; y_0, y_1, \dots) = \sum_{t \geq r} \Lambda_G(x,t)$ leading to

$$\tilde{\Lambda}_G(x,r) := \sum_{t \geq r} \Lambda_G(x,t).$$

Concrete examples are the effective eccentricity and diameter (Section 11.2.4).

Another type of reparametrization affects the locality of local statistics and distributions. In some situations it may useful to change the set of graph elements on which a certain statistic is defined. For example one may change a vertex-based statistic $\lambda_G(v)$ to an edge-based statistic $\lambda'_G(e)$ via the general relation

$$\lambda'_G(e) := f(\lambda_G(u), \lambda_G(v))$$

where $e = \{u, v\}$, and $f$ is an arbitrary function with two arguments (like sum, average, product etc.).

An example for this transformation is the eccentricity (Section 11.2.2), where the notion of locality is transformed from 'a pair of vertices' to 'a single vertex'.

**Localization.** To construct a local distribution $\Lambda$ from a global statistic $\gamma$, one may choose a subgraph $H(x,t) \subseteq G$ for each graph element $x \in X_G$ and each parameter $t \in P$. Then set

$$\Lambda_G(x,t) := \gamma_{H(x,t)}.$$

Of course this requires that the statistic $\gamma$ is defined on the subgraph $H(x,t)$. An example for the choice of $H(x,t)$ is:

- The ball of radius $t$ around $x$, i.e., the subgraph induced by the $t$-neighborhood $\mathrm{Neigh}_t(x)$ (all vertices $v$ with $d(x,v) \leq t$) of $x$ ([486, 540]).
- The largest/smallest subgraph of $G$ containing $x$ and satisfying a certain criterion, possibly depending on $t$, e.g. the largest subgraph among all subgraphs containing $x$ and having edge connectivity $\geq t$.

Similarly, a local statistic $\lambda$ can be derived from a global statistic $\gamma$ by choosing a subgraph $H(x)$ depending only on $x$ and setting

$$\lambda_G(x) := \gamma_{H(x)}.$$

An example for the choice of $H(x)$ is:

- The union of the cliques containing $x$ that have largest size among all cliques containing $x$.

The localization of a global distribution $\Gamma$ can be done in a similar manner to that described above. If the subgraph $H(x)$ is chosen solely based on the graph element $x$, this leads to the local distribution

$$\Lambda_G(x,t) := \Gamma_{H(x)}(t).$$

The last type of localization constructs a local distribution from a local statistic. Again a subgraph $H(x,t)$ is chosen for each graph element $x \in X_G$. But this time it additionally depends on a new parameter $t \in P$. This leads to

$$\Lambda_G(x,t) := \lambda_{H(x,t)}(x).$$

Transformations of these types can be found in [486, 540].

### 11.7.2 Visualization

A single-valued global statistic $\gamma$ is simply a value. Hence its visualization is trivial. It becomes more interesting if several graphs of a class $\mathcal{G}$ are examined. If the graphs depend on a parameter $t$, then one may examine the distribution $\gamma_{G(t)}$ where $G(t)$ is a graph with parameter $t$. This distribution may be treated like the distributions on a single graph, as described in the following.

Global distributions $\Gamma\colon P \to Y$ are maps from the set of parameters to the set of values. Hence they may be visualized as functions in the usual way. Furthermore this interpretation allows the application of techniques like interpolation or regression to obtain a functional description based on a set of parameters. Since we assume that the reader is familiar with these notions and techniques, we do not go into detail about them here.

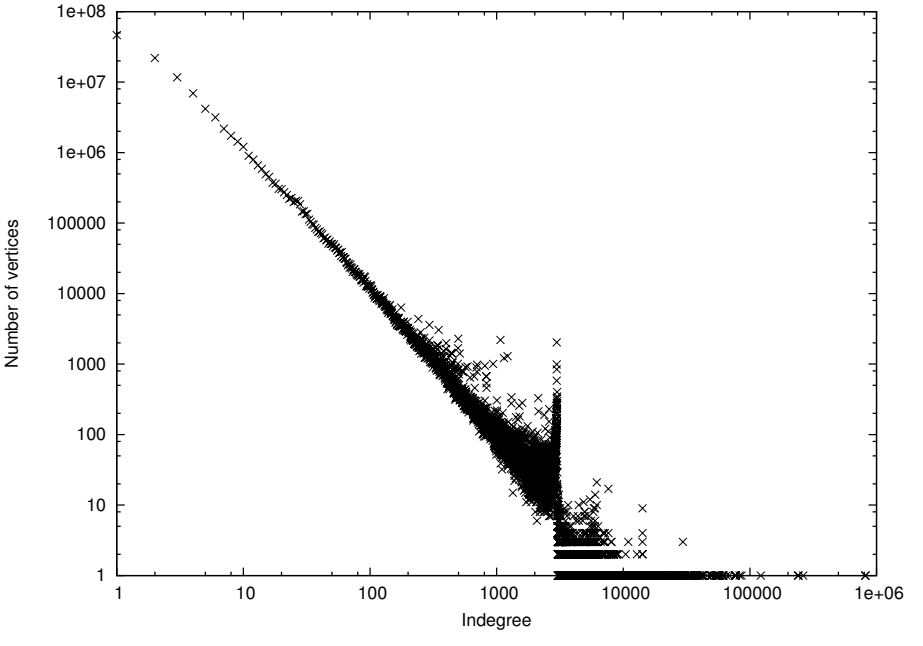


**Fig. 11.4.** The absolute indegree distribution of the 2001 Crawl of WebBase [566] with logarithmic scale

Figures 11.4 and 11.5 show two examples of visualizations of this type. They show the absolute in- and outdegree distributions of the 2001 WebBase Crawl [566]. Both diagrams have logarithmic scale and show a linear relation, indicating a power law (see Section 11.1).

For local statistics and distributions the visualization is more tricky. For a single-valued statistic $\lambda_G(x)$, one could choose an order $x_1, x_2, \ldots$ on the graph

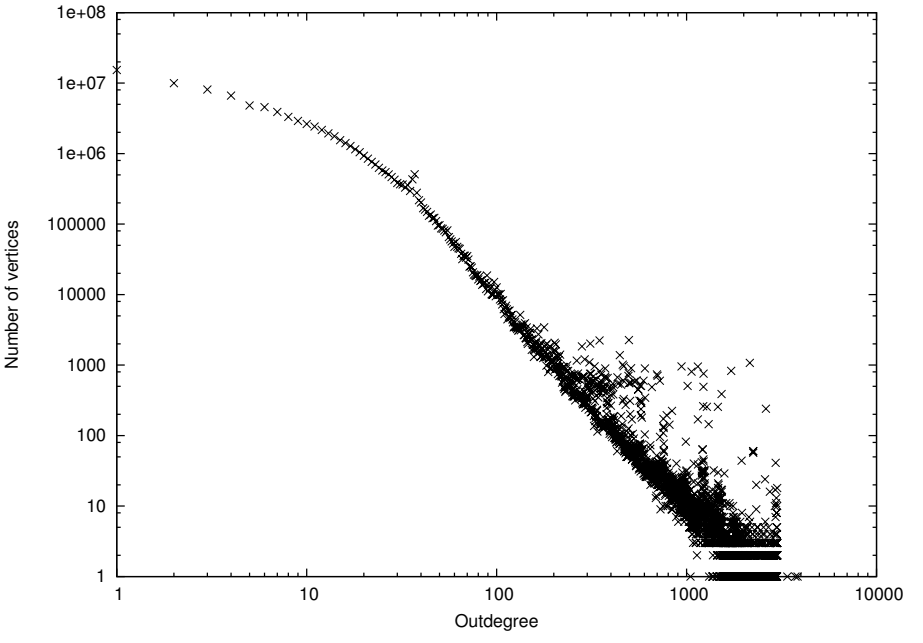


**Fig. 11.5.** The absolute outdegree distribution of the 2001 Crawl of WebBase [566] with logarithmic scale

elements and plot the resulting pairs $(i, \lambda_G(x_i))$. But the resulting diagram heavily depends on the chosen order. A better approach would be counting or averaging, and then using the global distributions or statistics, leading to a global distribution.

Different local statistics of the same graph, based on the same set $X_G$ of graph elements, may be visualized using a *scatter plot*. Let $\lambda_G$ and $\lambda'_G$ be two local statistics over the same set $X_G$ of graph elements. Then the *scatter plot* of $\lambda_G$ and $\lambda'_G$ consists of the pairs $(\lambda_G(x), \lambda'_G(x))$ for all graph elements $x \in X_G$. The scatter plot visualizes the relation between $\lambda$ and $\lambda'$ in the underlying graph $G$. If the resulting diagram is an unstructured cloud, the two statistics seem to be unrelated. If, on the other hand, the resulting diagram allows a functional description (obtained via regression, interpolation or similar techniques), the two statistics seem to be closely related – at least in the given graph.

Figure 11.6 shows the scatter plot of the absolute in- and outdegrees of the 2001 WebBase Crawl in logarithmic scale. Each cross represents one or more vertices having the corresponding combination of in- and outdegree. Since the resulting diagram is a very dense cloud, the two types of degree do not seem to be directly related.

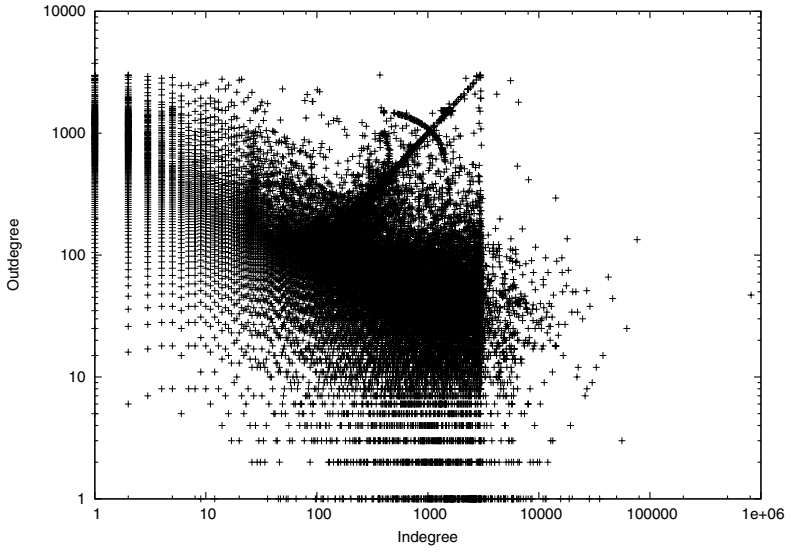


**Fig. 11.6.** The scatter plot of the in- and outdegrees of the 2001 Crawl of WebBase [566] with logarithmic scale

### 11.7.3 Sampling of Local Statistics

As we have seen, many network statistics are computationally expensive on large graphs, even though they require only polynomial space and time. Hence it seems sensible to use approximations.

Assume that $\sigma_G$ is an arbitrary local or global statistic derived from a local statistic $\lambda_G$ based on graph elements $x \in X_G$. Then one very common and accepted technique for the approximation of $\sigma_G$ is *sampling*. Instead of using the values $\lambda_G(x)$ for all graph elements $x \in X_G$ for the calculation of $\sigma$, only the values for some *samples* $x \in X_{\text{samples}} \subset X_G$ are used.

In general, the sampling of a (global) statistic $\sigma_G$ from a local statistic $\lambda_G \colon X_G \to Y$ has the following form:

1. Choose a set $X_{\text{samples}}$ of samples in $X_G$. (This may be done completely at random or using a specific strategy.)
2. Calculate $\lambda_G(x)$ for each sample $x \in X_{\text{samples}}$.
3. Calculate $\sigma_G$ from the set $\{\lambda_G(x) \mid x \in X_{\text{sample}}\}$.

Of course, this description is very rough. Depending on the concrete situation, adaptations might be necessary or useful. For example the strategy for the choice of the samples often determines the quality of the approximation. On the other hand, the calculation of the approximation of $\sigma_G$ from the samples may require

changes in the definition or calculation of $\sigma_G$ to improve the quality or to obtain reasonable values.

In any case the quality of this approximation has to be certified either by experiments or by analysis of the specific graph models. In general it is clear that an increase of the number of samples directly results in a better approximation.

For example the relative hop plot of a graph can be estimated quite easily using sampling (see Algorithm 25). For $l = n$ one obtains the exact values $\bar{P}$. The runtime of this algorithm is $\mathcal{O}(ln^2 \log n)$ and the space requirements are $\mathcal{O}(n)$ (in addition to the graph). Unfortunately, we do not know of any results about the quality of the estimates.

**Algorithm 25**: A simple example for sampling: The relative hop plot

```
Input: A graph G = (V, E).
begin
    Set P(h) = 0 for h = 0, 1, 2, ..., diam(G)
    for i ∈ {1, ..., l} do
        Choose an unused vertex u_i
        Calculate the distances d(u_i, v) for all v ∈ V by solving the SSSP for
        source u_i
        for v ∈ V do
            for h ∈ {d(u_i, v), ..., diam(G)} do
                P(h) ← P(h) + 1;
    p(h) ← P(h) / (ln)
end
```

## 11.8 Chapter Notes

The first articles mentioning the power law distribution of the degrees of large 'naturally' occurring networks are [16, 40, 197]. Following that, more detailed studies were done. During these and the initial studies a power law was found for several distributions. These include

- the hop plot of the Internet (see Section 11.2.3) [197],
- the distribution of the eigenvalues of the Internet (see Chapter 14) [197],
- the size of certain components of the WWW [102].

Other examples of power law distributions can be found in [405, 147].

Recently, the seeming universality of the power law was questioned (e.g. [117]). Furthermore Bu and Towsley observed in [104] that the *complementary cumulative distribution of degrees* of the Internet seems to fit better with the power law than the degree distributions did. More precisely, this means that

$$D(k) := |\{v \mid d(v) \leq k\}| = ck^{\lambda}$$

for some constants $c$ and $\lambda$.

The average (connected) distance of networks is used in [573, 16, 102, 104].

The eccentricity of the Internet on the level of autonomous systems is examined by Magoni and Pansiot in [405]. They observe a mean eccentricity of 7, while the radius is 5.

The eccentricity also appears in [540], where it is called *node diameter*.

In [486, 540] Tangmunarunkit et al. use the average of absolute and relative hop plots over all vertices in the graph. They call their statistic *expansion*.

In [197] Faloutsos et al. give an alternative definition of the effective diameter, which relies on a power law distribution of the hop plot. They observe that the absolute hop plot $P(h)$ satisfies $P(1) = n + 2m$ (distance 1 is given by the edges) and therefore $P(h) = (n+2m)h^{\mathcal{H}}$ for a constant exponent $\mathcal{H}$. Then their effective diameter $\delta_{\text{eff}}$ is the value such that $P(\delta_{\text{eff}}) = (n+2m)\delta_{\text{eff}}^{\mathcal{H}} = n^2$, leading to $\delta_{\text{eff}} = \left(\frac{n^2}{n+2m}\right)^{1/\mathcal{H}}$. Therefore their effective diameter estimates the maximal distance under the assumption that the hop plot satisfies the power law exactly.

The number of distinct shortest paths is used in [405]. There the fraction of all pairs of vertices with a certain number of distinct shortest paths is measured.

Tangmunarunkit et al. apply the distortion in [486, 540] to analyze the network of autonomous systems of the Internet. They sample the distortion over a collection of spanning trees, generated by unspecified heuristics using the number of all-pairs-shortest-paths traversing a specific link. This global statistic is localized with balls of radius $h$ around a vertex $v$, leading to a local distribution. This in turn is globalized by averaging over all center nodes.

The term *clustering coefficient* might be somewhat misleading. Clustering in networks, see Chapter 8, is based on many concepts. There are graphs with high clustering coefficient that are contrary to most of these concepts. A relatively high *clustering coefficient* together with a small diameter is known as the *small world* property of a network [573], see also Section 13.1.2 for details.

Also the term *transitivity* as used in Section 11.5 is somewhat misleading. In a directed graph the edge $(u, w)$ is called transitive if there is a path consisting of the two edges $(u, v)$ and $(v, w)$. In this sense the term *transitivity ratio* was defined by Harary and Kommel in 1979 [280]. The equivalent *index of transitivity* was defined in the context of linguistics even earlier, in the year 1957 [282]. The *transitivity* of Newman, Watts and Strogatz [447] is an undirected version of the above terms. Interestingly the term *triple* also appeared already in its directed version in [280].