The Web as a graph: measurements, models, and methods

Jon M. Kleinberg[1], Ravi Kumar[2], Prabhakar Raghavan[2], Sridhar Rajagopalan[2], and Andrew S. Tomkins[2]

[1] Department of Computer Science, Cornell University, Ithaca, NY 14853.
[2] IBM Almaden Research Center K53/B1, 650 Harry Road, San Jose CA 95120.

Abstract. The pages and hyperlinks of the World-Wide Web may be viewed as nodes and edges in a directed graph. This graph is a fascinating object of study: it has several hundred million nodes today, over a billion links, and appears to grow exponentially with time. There are many reasons — mathematical, sociological, and commercial — for studying the evolution of this graph. In this paper we begin by describing two algorithms that operate on the Web graph, addressing problems from Web search and automatic community discovery. We then report a number of measurements and properties of this graph that manifested themselves as we ran these algorithms on the Web. Finally, we observe that traditional random graph models do not explain these observations, and we propose a new family of random graph models. These models point to a rich new sub-field of the study of random graphs, and raise questions about the analysis of graph algorithms on the Web.

1 Overview

Few events in the history of computing have wrought as profound an influence on society as the advent and growth of the World-Wide Web. For the first time, millions — soon to be billions — of individuals are creating, annotating and exploiting hyperlinked content in a distributed fashion. A particular Web page might be authored in any language, dialect, or style by an individual with any background, culture, motivation, interest, and education; might range from a few characters to a few hundred thousand; might contain truth, falsehood, lies, propaganda, wisdom, or sheer nonsense; and might point to none, few, or several other Web pages. The hyperlinks of the Web endow it with additional structure; and the network of these links is rich in latent information content. Our focus in this paper is on the directed graph induced by the hyperlinks between Web pages; we refer to this as the *Web graph.*

For our purposes, nodes represent static html pages and hyperlinks represent directed edges. Recent estimates [4] suggest that there are several hundred million nodes in the Web graph; this quantity is growing by a few percent a month. The average node has roughly seven hyperlinks (directed edges) to other pages, making for a total of several billion hyperlinks in all.

There are several reasons for studying the Web graph. The structure of this graph has already led to improved Web search [6, 8, 21, 29], more accurate topic-classification algorithms [11] and has inspired algorithms for enumerating emergent cyber-communities [23]. The hyperlinks themselves represent a fecund source of sociological information. Beyond the intrinsic interest of the structure of the Web graph, measurements of the graph and of the behavior of users as they traverse the graph, are of growing commercial interest.

1.1 Guided tour of this paper

In Section 2 we review two algorithms that have been applied to the Web graph: Kleinberg's HITS method [21] and the enumeration of certain bipartite cliques [23]. We have chosen these algorithms here because they are both driven by the presence of certain structures in the Web graph. These structures (which we will detail below) appear to be a fundamental by-product of the manner in which Web content is created.

In Section 3 we summarize a number of measurements we have made on the entire Web graph, and on particular local subgraphs of interest. We show, for instance, that the in- and out-degrees of nodes follow Zipfian (inverse polynomial) distributions [12, 17, 26, 31]. This and other measurements of the frequency of occurrence of certain structures suggest that traditional random graph models such as $\mathcal{G}_{n,p}$ [7] are likely to do a poor job of modeling the Web graph.

In Section 4 we lay down a framework for a class of random graph models, and give evidence that at least some of our observations about the Web (for instance, the degree distributions) can be established in these models. A notable aspect of these models is that they embody some version of a *copying* process: we add links to a node v by picking a random (other) node u in the graph, and copying some links from u to v (i.e., we add edges from v to some of the nodes that u points to). Such copying operations seem to be fundamental both to the process of content-creation on the Web and to the explanation of the statistics we have observed. One consequence is that the mathematical analysis of these graph models promises to be far harder than in traditional graph models in which the edges emanating from a node are drawn independently.

We conclude in Section 5 with a number of directions for further work.

1.2 Related work

Analysis of the structure of the Web graph has been used to enhance the quality of Web search [5, 6, 8, 9, 21, 29]. The topics of pages pointed to by a Web page can be used to improve the accuracy of determining the (unknown) topic of this page in the setting of supervised classification [11].

Statistical analysis of the structure of the academic citation graph has been the subject of much work in the Sociometrics community. As we discuss below, Zipf distributions seem to characterize Web citation frequency. Interestingly, the same distributions have also been observed for citations in the academic literature. This fact, known as *Lotka's law*, was demonstrated by Lotka in 1926 [26].

Gilbert [17] presents a probabilistic model explaining Lotka's law, which is similar in spirit to our proposal, though different in details and application. The field of bibliometrics [14, 16] has studied phenomena in citation; some of these insights have been applied to the Web as well [25].

A view of the Web as a semi-structured database has been advanced by many authors. In particular, LORE [1] and WebSQL [27] use graph-theoretic and relational views of the Web respectively. These views support structured query interfaces to the Web (Lorel [1] and WebSQL [27]) that are evocative of and similar to SQL. An advantage of this approach is that many interesting queries can be expressed as simple expressions in the very powerful SQL syntax. The corresponding disadvantage is that the generality comes with an associated computational cost which can be prohibitive until we develop query optimizers for Web graph computations that similar to those available for relational data. LORE and WebSQL are but two examples of projects in this space. Some other examples are W3QS [22], WebQuery [8], Weblog [24], and ParaSite/Squeal [29].

Traditional data mining research (see for instance Agrawal and Srikant [2]) focuses largely on algorithms for finding association rules and related statistical correlation measures in a given dataset. However, efficient methods such as *a priori* [2] or even more general methodologies such as *query flocks* [30], do not scale to the numbers of "items" (pages) in the Web graph. This number is already two to three orders of magnitude more than the number of items in a typical market basket analysis.

The work of Mendelzon and Wood [28] is an instance of structural methods in mining. They argue that the traditional SQL query interface to databases is inadequate in its power to specify several structural queries that are interesting in the context of the Web. They provide the example of path connectivity between nodes subject to some constraints on the sequence of edges on the path (expressible in their case as a regular expression). They describe G^+, a language with greater expressibility than SQL for graph queries.

2 Algorithms

We have observed the following recurrent phenomenon on the Web. For any particular topic, there tend to be a set of "authoritative" pages focused on the topic, and a set of "hub" pages, each containing links to useful, relevant pages on the topic. This observation motivated the development of two algorithms which we describe below. The first is a search algorithm that distils high-quality pages for a topic query (Section 2.1), and the second enumerates all topics that are represented on the Web in terms of suitably dense sets of such hub/authority pages (Section 2.2).

2.1 The HITS algorithm

Beginning with a search topic, specified by one or more query terms, Kleinberg's HITS algorithm [21] applies two main steps: a *sampling* step, which constructs a focused collection of several thousand Web pages likely to be rich in relevant authorities; and a *weight-propagation* step, which determines numerical estimates of hub and authority scores by an iterative procedure. The pages with the highest scores are returned as hubs and authorities for the search topic.

Any subset S of nodes induces a *subgraph* containing all edges that connect two nodes in S. The first step of the HITS algorithm constructs a subgraph expected to be rich in relevant, authoritative pages, in which it will search for hubs and authorities. To construct this subgraph, the algorithm first uses keyword queries to collect a *root set* of, say, 200 pages from a traditional index-based search engine. This set does not necessarily contain authoritative pages; however, since many of these pages are presumably relevant to the search topic, one can expect some to contain links to prominent authorities, and others to be linked to by prominent hubs. The root set is therefore expanded into a *base set* by including all pages that are linked to by pages in the root set, and all pages that link to a page in the root set (up to a designated size cut-off). This follows the intuition that the prominence of authoritative pages is typically due to the endorsements of many relevant pages that are not, in themselves, prominent. We restrict our attention to this base set for the remainder of the algorithm; this set typically contains roughly 1000–3000 pages, and that (hidden) among these are a large number of pages that one would subjectively view as authoritative for the search topic.

We begin with one modification to the subgraph induced by the base set. Links between two pages on the same Web site very often serve a purely navigational function, and typically do not represent conferral of authority. We therefore delete all such links from the subgraph induced by the base set, and apply the remainder of the algorithm to this modified subgraph.

Good hubs and authorities can be extracted from the base set by giving a concrete numerical definition to the intuitive notions of hub and authority from the beginning of this section. The algorithm associates a non-negative *authority weight* x_p and a non-negative *hub weight* y_p with each page $p \in V$. We will only be interested in the *relative* values of these weights, not their actual magnitudes; so in the manipulation of the weights, we apply a normalization to prevent the weights from overflowing. (The actual choice of normalization does not affect the results; we maintain the invariant that the squares of all weights sum to 1.) A page p with a large weight x_p (resp. y_p) will be viewed as a "better" authority (resp. hub). Since we do not impose any *a priori* estimates, all x- and y-values are set to a uniform constant initially. As will be seen later, however, the final results are essentially unaffected by this initialization.

The authority and hub weights are updated as follows. If a page is pointed to by many good hubs, we would like to increase its authority weight; thus for a page p, the value of x_p is updated to be to be the sum of y_q over all pages q that link to p:

$$x_p = \sum_{q \text{ such that } q \to p} y_q, \tag{1}$$

where the notation $q \to p$ indicates that q links to p. In a strictly dual fashion, if a page points to many good authorities, its hub weight is increased via

$$y_p = \sum_{q \text{ such that } p \to q} x_q. \tag{2}$$

There is a more compact way to write these updates, and it turns out to shed more light on the mathematical process. Let us number the pages $\{1, 2, \ldots, n\}$ and define their *adjacency matrix* A to be the $n \times n$ matrix whose (i, j)th entry is equal to 1 if page i links to page j, and is 0 otherwise. Let us also write the set of all x-values as a vector $x = (x_1, x_2, \ldots, x_n)$, and similarly define $y = (y_1, y_2, \ldots, y_n)$. Then the update rule for x can be written as $x \leftarrow A^T y$ and the update rule for y can be written as $y \leftarrow Ax$. Unwinding these one step further, we have

$$x \leftarrow A^T y \leftarrow A^T Ax = (A^T A)x \tag{3}$$

and

$$y \leftarrow Ax \leftarrow AA^T y = (AA^T)y. \tag{4}$$

Thus the vector x after multiple iterations is precisely the result of applying the *power iteration* technique to $A^T A$ — we multiply our initial iterate by larger and larger powers of $A^T A$ — and a standard result in linear algebra tells us that this sequence of iterates, when normalized, converges to the principal eigenvector of $A^T A$. Similarly, the sequence of values for the normalized vector y converges to the principal eigenvector of AA^T. (See the book by Golub and Van Loan [18] for background on eigenvectors and power iteration.)

In fact, power iteration will converge to the principal eigenvector for any "non-degenerate" choice of initial vector — in our case, for example, for any vector all of whose entries are positive. This says that the hub and authority weights computed are truly an *intrinsic* feature of the collection of linked pages, not an artifact of the choice of initial weights or the tuning of arbitrary parameters. Intuitively, the pages with large weights represent a very "dense" pattern of linkage, from pages of large hub weight to pages of large authority weight. This type of structure — a densely linked *community* of thematically related hubs and authorities — will be the motivation underlying Section 2.2 below.

Finally, the output of HITS algorithm for the given search topic is a short list consisting of the pages with the largest hub weights and the pages with the largest authority weights. Thus the algorithm has the following interesting feature: after using the query terms to collect the root set, the algorithm completely ignores textual content thereafter. In other words, HITS is a purely link-based computation once the root set has been assembled, with no further regard to the query terms. Nevertheless, HITS provides surprisingly good search results for a wide range of queries. For instance, when tested on the sample query "search engines", the top authorities returned by HITS were Yahoo!, Excite, Magellan, Lycos, and AltaVista — even though none of these pages (at the time of the experiment) contained the phrase "search engines."

In subsequent work [5, 9, 10], the HITS algorithm has been generalized by modifying the entries of A so that they are no longer boolean. These modifications take into account the content of the pages in the base set, the internet domains in which they reside, and so on. Nevertheless, most of these modifications retain the basic power iteration process and the interpretation of hub and authority scores as components of a principal eigenvector, as above.

2.2 Trawling the Web for cyber-communities

In this section we turn to a second algorithm developed for the Web graph. In contrast to HITS, which is a search algorithm designed to find high-quality pages about a fixed topic, the *trawling* algorithm described below seeks to enumerate *all* topics (under a certain definition), and therefore processes the entire Web graph.

We begin with a more concrete definition of the types of topic we wish to enumerate. Recall that a complete bipartite clique $K_{i,j}$ is a graph in which every one of i nodes has an edge directed to each of j nodes (in the following treatment it is simplest to think of the first i nodes as being distinct from the second j; in fact this is not essential to our algorithms or models). We further define a *bipartite core* $C_{i,j}$ to be a graph on $i + j$ nodes that contains at least one $K_{i,j}$ as a subgraph. The intuition motivating this notion is the following: on any sufficiently well represented topic on the Web, there will (for some appropriate values of i and j) be a bipartite core in the Web graph. Figure 1 illustrates an instance of a $C_{4,3}$ in which the four nodes on the left have hyperlinks to the home pages of three major commercial aircraft manufacturers. Such a subgraph

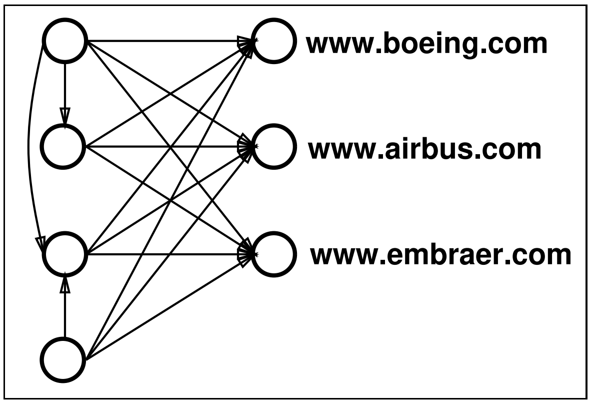

Fig. 1. A bipartite core.

of the Web graph would be suggestive of a "cyber-community" of aficionados of commercial aircraft manufacturers who create hub-like pages like the four on the left side of Figure 1. These pages co-cite the authoritative pages on the right.

Loosely speaking, such a community emerges in the Web graph when many (hub) pages link to many of the same (authority) pages. In most cases, the hub pages in such communities may not co-cite *all* the authoritative pages for that topic. Nevertheless, it is tempting to subscribe to the following weaker hypothesis: every such community will contain a bipartite core $C_{i,j}$ for non-trivial values of i and j. Turning this around, we could attempt to identify a large fraction of cyber-communities by enumerating all the bipartite cores in the Web for, say $i = j = 3$; we call this process *trawling.* Why these choices of i and j? Might it not be that for such small values of i and j, we discover a number of coincidental co-citations, which do not truly correspond to communities?

In fact, in our experiment [23] we enumerated $C_{i,j}$'s for values of i ranging from 3 to 9, for j ranging from 3 to 20. The results suggest that (1) the Web graph has several hundred thousand such cores, and (2) it appears that only a minuscule fraction of these are coincidences — the vast majority do in fact correspond to communities with a definite topic focus. Below we give a short description of this experiment, followed by some of the principal findings.

From an algorithmic perspective, the naive "search" algorithm for enumeration suffers from two fatal problems. First, the size of the search space is far too large — using the naive algorithm to enumerate all bipartite cores with two Web pages pointing to three pages would require examining approximately 10^{40} possibilities on a graph with 10^8 nodes. A theoretical question (open as far as we know): does the work on fixed-parameter intractability [13] imply that we cannot – in the worst case – improve on naive enumeration for bipartite cores? Such a result would argue that algorithms that are provably efficient on the Web graph must exploit some feature that distinguishes it from the "bad" inputs for fixed-parameter intractability. Second, and more practically, the algorithm requires random access to edges in the graph, which implies that a large fraction of the graph must effectively reside in main memory to avoid the overhead of seeking a disk on every edge access.

We call our algorithmic methodology the *elimination-generation* paradigm. An algorithm in the elimination/generation paradigm performs a number of sequential passes over the Web graph, stored as a binary relation. During each pass, the algorithm writes a modified version of the dataset to disk for the next pass. It also collects some metadata which resides in main memory and serves as state during the next pass. Passes over the data are interleaved with sort operations, which change the order in which the data is scanned, and constitute the bulk of the processing cost. We view the sort operations as alternately ordering directed edges by source and by destination, allowing us alternately to consider out-edges and in-edges at each node. During each pass over the data, we interleave *elimination* operations and *generation* operations, which we now detail.

Elimination. There are often easy necessary (though not sufficient) conditions that have to be satisfied in order for a node to participate in a subgraph of interest to us. Consider the example of $C_{4,4}$'s. Any node with in-degree 3 or smaller cannot participate on the right side of a $C_{4,4}$. Thus, edges that are directed into such nodes can be pruned from the graph. Likewise, nodes with out-degree 3 or smaller cannot participate on the left side of a $C_{4,4}$. We refer to these necessary conditions as *elimination filters.*

Generation. Generation is a counterpoint to elimination. Nodes that barely qualify for potential membership in an interesting subgraph can easily be verified to either belong in such a subgraph or not. Consider again the example of a $C_{4,4}$. Let u be a node of in-degree exactly 4. Then, u can belong to a $C_{4,4}$ if and only if the 4 nodes that point to it have a neighborhood intersection of size at least 4. It is possible to test this property relatively cheaply, even if we allow the in-degree to be slightly more than 4. We define a *generation filter* to be a procedure that identifies barely-qualifying nodes, and for all such nodes, either outputs a core or proves that such a core cannot exist. If the test embodied in the generation filter is successful, we have identified a core. Further, regardless of the outcome, the node can be pruned since all potential interesting cores containing it have already been enumerated.

Note that if edges appear in an arbitrary order, it is not clear that the elimination filter can be easily applied. If, however, the edges are sorted by source (resp. destination), it is clear that the outlink (resp. inlink) filter can be applied in a single scan. Details of how this can be implemented with few passes over the data (most of which is resident on disk, and must be streamed through main memory for processing) may be found in [23].

After an elimination/generation pass, the remaining nodes have fewer neighbors than before in the residual graph, which may present new opportunities during the next pass. We can continue to iterate until we do not make significant progress. Depending on the filters, one of two things could happen: (1) we repeatedly remove nodes from the graph until nothing is left, or (2) after several passes, the benefits of elimination/generation "tail off" as fewer and fewer nodes are eliminated at each phase. In our trawling experiments, the latter phenomenon dominates.

Why should such algorithms run fast? We make a number of observations about their behavior:

1. The in/out-degree of every node drops monotonically during each elimination/generation phase.
2. During each generation test, we either eliminate a node u from further consideration (by developing a proof that it can belong to no core), or we output a subgraph that contains u. Thus, the total work in generation is linear in the size of the Web graph plus the number of cores enumerated, assuming that each generation test runs in constant time.
3. In practice, elimination phases rapidly eliminate most nodes in the Web graph. A complete mathematical analysis of iterated elimination is beyond the scope of this paper, and requires a detailed understanding of the kinds of random graph models we propose in Section 4.

Kumar *et al.* [23] report trawling a copy of the Web graph derived from a Web crawl obtained from Alexa, inc. This experiment generated well over 100,000 bipartite cores $C_{3,3}$. Note that since these cores are the result of enumeration (rather than querying), they lack any form of context or topic with which one can tag them. Indeed, as noted in [23], the only certain way of determining whether a core is coincidental or real is manual inspection. The results of [23] suggest that over 90% of the cores enumerated in this experiment are not coincidental, but in fact bear definite themes.

We conclude this section by remarking that bipartite cores are not necessarily the only subgraph enumeration problems that are interesting in the setting of the Web graph. The subgraphs corresponding to Webrings look like bidirectional stars, in which there is a central page with links to and from a number of "spoke" pages. Cliques, and directed trees, are other interesting structures for enumeration. Devising general paradigms for such enumeration problems appears to be difficult, unless one understands and exploits the peculiarities of the Web graph. The next two sections address this issue.

3 Measurements

In the course of experiments with the algorithms of Section 2, we were able to study many of the local properties of the Web graph. In this section we survey these observations, and point out that traditional random graph models like $\mathcal{G}_{n,p}$ would do a poor job of explaining them.

3.1 Degree distributions

We begin with the in- and out-degrees of nodes in the Web graph. Figure 2 is a log-log plot (with the x-axis negated) of the in-degree distribution. The plot suggests that the probability that a node has degree i is proportional to $1/i^{\alpha}$, where α is approximately 2. Such a Zipfian distribution [31] cannot arise in a model such as $\mathcal{G}_{n,p}$, where (due to the superposition of Bernoulli trials) in-degrees exhibit either a Poisson or a binomial distribution. Consider next the

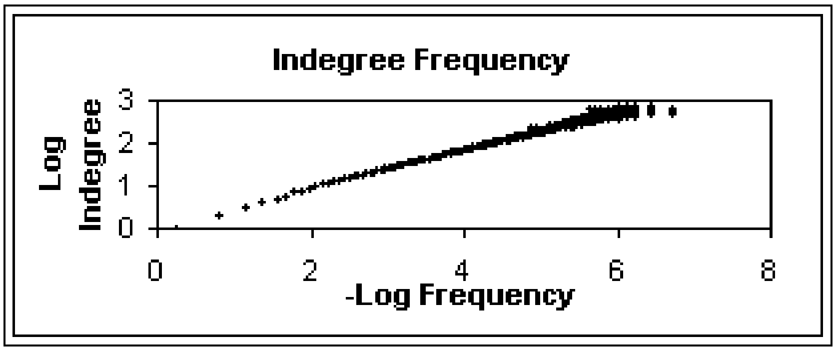

Fig. 2. In-degree distribution.

out-degree distribution (Figure 3). Again the distribution looks faintly Zipfian, although here the variations seem larger. The average out-degree we observed is about 7.2. A natural question now arises: if $\mathcal{G}_{n,p}$ will not result in such Zipfian

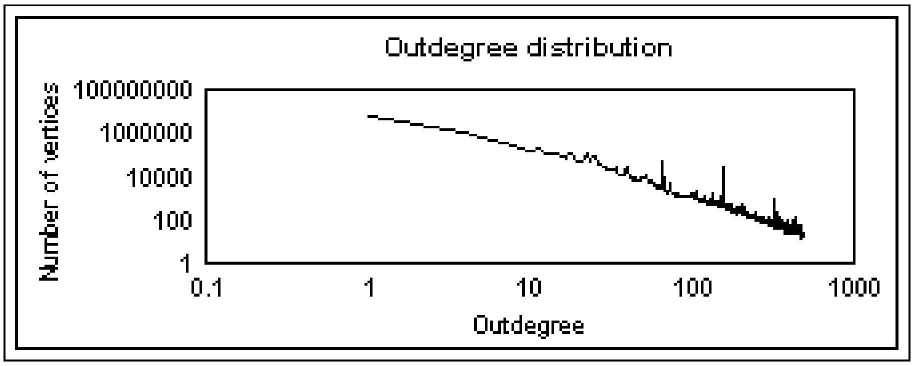

Fig. 3. Out-degree distribution.

distributions, what is a natural stochastic process that will? We provide a partial answer in Section 4.

3.2 Number of bipartite cores

We turn next to the distribution of cores $C_{i,j}$, based on the numbers discovered in the trawling experiment [23]. Figures 4 and 5 depict these distributions as

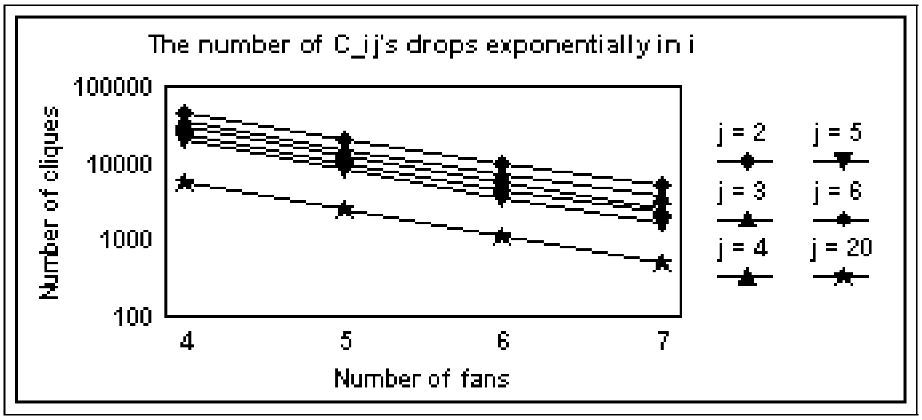

Fig. 4. Core distribution by left side.

functions of i and j; these quantities are from a crawl of the Web that is roughly two years old, obtained from Alexa, inc. The number of Web pages in this crawl is roughly 100 million. How does this compare with the numbers one might observe in a graph generated using $\mathcal{G}_{n,p}$, say for $np = 7.2$ (our observed out-degree)? A simple calculation yields that the expected number of $C_{i,j}$'s is

$$\binom{n}{i}\binom{n}{j}\left(\frac{7.2}{n}\right)^{ij},$$

which is negligible for $ij > i + j$. Clearly, one cannot explain the multitude of $C_{i,j}$'s in the Web graph using $\mathcal{G}_{n,p}$; once again, we hope that the models we propose in Section 4 can explain these observations.

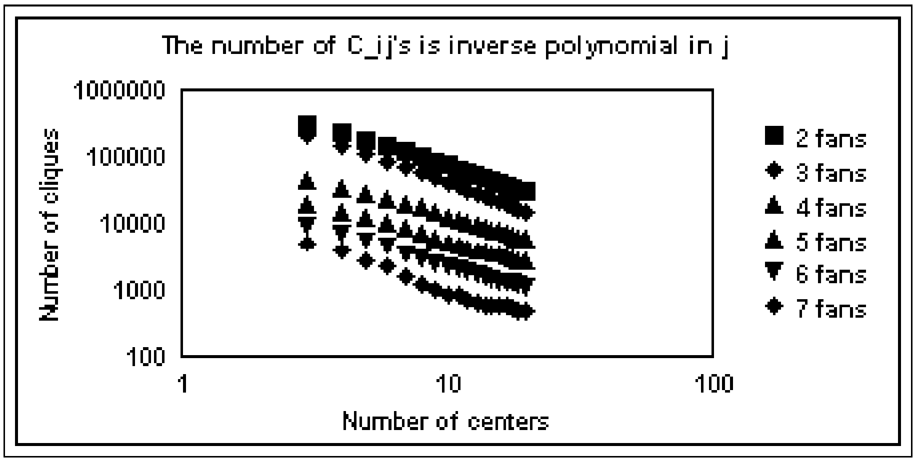

Fig. 5. Core distribution by right side.

3.3 Connectivity of local subgraphs

We now consider some relating to the connectivity of local subgraphs of the Web graph. We begin by fixing our set of local subgraphs to be the base sets arising in the HITS algorithm of Section 2. To recapitulate, we begin with c nodes obtained by issuing a keyword query to a text-based search engine. We then expand this root set to the base set by adding any page linked to by a page in the root set, and any page linking to a page in the root set (up to a cut-off of d pages per element of the root set). In experiments, we have applied this with c, the size of the root set, equal to 200; and d, the cut-off, equal to 50. As above, the resulting base set typically consists of roughly 1000 to 3000 pages, and since we wish to focus on cross-site links we discard links between two pages within the same site.

We now ask how well-connected these local subgraphs are. We will view G primarily as a directed graph, but we also define the undirected version G_u obtained by ignoring the direction of all edges.

A collection of graphs constructed in this way for an ensemble of fifty sample query terms reveals some consistently occurring structural properties; we will discuss several of these here at a qualitative level.

A range of connectivity relations. A first observation is that the graphs G_u are not, in general, connected. This is intuitively very natural: the initial root set typically induces very few edges; and while the expansion to the base set serves to connect many of these nodes, others remain in small isolated components.

A graph G_u of this form, however, does typically contain a "giant component" — a connected component that contains a significant constant fraction of all the nodes. In all the cases considered, there was only a single giant component of G_u.

Turning to a stronger measure, we consider *biconnectivity*: we say that two nodes u and v are *biconnected* if there is no third node w so that w lies on all u-v paths. We will call the equivalence classes of nodes under this relation the *biconnected components.* The graph G_u typically has a "giant biconnected component," again unique. Thus, we can intuitively picture the structure of G_u as consisting of a central biconnected "nucleus," with small pieces connected to this nucleus by cut-nodes, and with other small pieces not connected to this nucleus at all.

The biconnected nucleus contains much of the interesting structure in G; for example, it generally contains all the top hubs and authorities computed by HITS. We will use H to denote the subgraph of G induced on this biconnected nucleus; H is thus a directed graph, and we use H_u to denote its undirected version.

We can try to further refine the structure of H by looking at *strongly connected components.* In this we make crucial use of the orientation of edges: u and v are related under strong connectivity if each can reach the other by a directed path. For this relation, however, we discover the following: the subgraphs H do not contain "giant strongly connected components." Indeed, for many of the graphs we considered, the largest strongly connected component had size less than 20.

Thus, while G is not connected when viewed in its entirety, it can be viewed as having a large subgraph that is biconnected as an undirected graph. G, however, does not generally contain any large strongly connected subgraphs.

Alternating connectivity. Biconnectivity yielded a giant component; strong connectivity pulverized the graph into tiny components. It is natural to ask whether there is a natural connectivity measure that takes some account of the orientation of edges and still results in large "components." We now describe one such measure, *alternating connectivity*, and observe a sense in which it is difficult to find the right definition of "component" under this measure.

If u and v are nodes, we say that a sequence P of edges in G is an *alternating path* from u to v if (1) P is a path in G_u with endpoints u and v, and (2) the orientations of the edges of P strictly alternate in G. Thus, P leads from u to v via a sequence of link traversals that alternate forward and backward directions. This definition corresponds closely to the HITS algorithm. Intuitively, the strict alternation of edge orientations parallels the way in which hub and authority weight flows between nodes; and indeed, the hub and authority weights effectively compute the relative growth rates of alternating paths from each node. Furthermore, two steps in an alternating path connect two nodes that either cite the same page, or are cited by the same page — this notion of co-citation has been used as a similarity measure among documents [14] and among web pages (e.g. [11, 21, 25]).

Suppose we write $u \sim v$ if there is an alternating path between u and v. It is clear immediately from the definition that the relation $\sim$ is symmetric; however, it is easy to construct examples with nodes u, v, w so that $u \sim v$ and $v \sim w$, but $u \not\sim w$. As a result, $\sim$ is not transitive and hence not an equivalence relation.

However, we can show a sense in which $\sim$ is "nearly" an equivalence relation: we can prove that if a node u is related to three nodes v_1, v_2, v_3, then at least one pair among $\{v_1, v_2, v_3\}$ is also related. We can call such a relation *claw-free* — no node is related to three nodes that are themselves mutually unrelated.

In tests on the subgraphs H, we find that a large fraction of all pairs of nodes $u, v \in H$ are related by alternating connectivity. Among pairs that are related, we can define their *undirected distance* as the length of the shortest u-v path in H_u; and we can define their *alternating distance* as the length of the shortest alternating u-v path in H. We find that the average alternating distance between related pairs is generally at most a factor of two more than the average undirected distance between them; this indicates that the biconnected nuclei H are rich in short, alternating paths.

4 Model

In this section we lay the foundation for a class of plausible random graph models in which we can hope to establish many of our observations about the local structure of the Web graph. There are a number of reasons for developing such a model:

1. It allows us to model various structural properties of the Web graph — node degrees, neighborhood structures, etc. Of particular interest to us is the distribution of Web structures such as $C_{i,j}$'s which are signatures of Web communities or other Web phenomena of interest.
2. It allows us to predict the behavior of algorithms on the Web; this is is of particular interest when these algorithms are doomed to perform poorly on worst-case graphs.
3. It suggests structural properties of today's Web that we might be able to verify and then exploit.

4. It allows us to make predictions about the shape of the Web graph in the future.

Desiderata for a Web graph model. We begin by reviewing some criteria that are desirable in such a graph model; many of these are motivated by empirical observations on the structure of the Web graph. Next we present our model. Note that we do not seek to model the text or the sizes of the pages; we are only interested here in the interconnection patterns of links between pages. We would like our model to have the following features:

1. It should have a succinct and fairly natural description.
2. It should be rooted in a plausible macro-level process for the creation of content on the Web. We cannot hope to model the detailed behavior of the many users creating Web content. Instead, we only desire that the aggregate formation of Web structure be captured well by our graph model. Thus, while the model is described as a stochastic process for the creation of individual pages, we are really only concerned with the aggregate consequences of these individual actions.
3. It should not require some *a priori* static set of "topics" that are part of the model description — the evolution of interesting topics and communities should instead be an emergent feature of the model.[1] Such a model has several advantages:
   – It is extremely difficult to characterize the set of topics on the Web; thus it would be useful to draw statistical conclusions without such a characterization.
   – The set of topics reflected in Web content has proven to be fairly dynamic. Thus, the shifting landscape of actual topics will need to be addressed in any topic-aware model of time-dependent growth.
4. We would like the model to reflect many of the structural phenomena we have observed in the Web graph.

4.1 A class of random graph models

In our model, we seek to capture the following intuition:

– some page creators on today's Web may link to other sites without regard to the topics that are already represented on the Web, but
– *most* page creators will be drawn to Web pages covering existing topics of interest to them, and will link to pages within some of these existing topics.

We have already observed that the Web graph has many hub pages that contain resource lists focused on a topic. Here is the dominant phenomenon for link-creation in our model: a user encounters a resource list for a topic of interest to him, and includes many links from this list in his/her page.

[1] In particular, we avoid models of the form "Assume each node is some combination of topics, and add an edge from one page to another with probability dependent on some function of their respective combinations."

We reiterate that this process is *not* meant to reflect individual user behavior on the Web; rather, it is a local procedure which in aggregate works well in describing page creation on the Web and which implicitly captures topic creation as follows: first, a few scattered pages begin to appear about the topic. Then, as users interested in the topic reach critical mass, they begin linking these scattered pages together, and other interested users are able to discover and link to the topic more easily. This creates a "locally dense" subgraph around the topic of interest. This intuitive view summarizes the process from a page-creator's standpoint; we now recast this formulation in terms of a random graph model that — again, on aggregate — captures the above intuition.

Indeed, it is our thesis that random copying is a simple, plausible stochastic mechanism for creating Zipfian degree distributions. Below, we state at a high level our model for the evolution of the Web graph. We are unable to provide complete probabilistic analyses of the graphs generated by even the simplest concrete instantiations of such models. Heuristic calculations, however, yield distributions for degree sequences, $C_{i,j}$'s and other local structures that conform remarkably well with our observations.

Our model is characterized by four stochastic processes — *creation* processes $\mathcal{C}_v$ and $\mathcal{C}_e$ for node- and edge-creation, and *deletion* processes $\mathcal{D}_v$ and $\mathcal{D}_e$ for node- and edge-deletion. These processes are discrete-time processes. Each process is a function of the time step, and of the current graph.

A simple node-creation process would be the following: independently at each step, create a node with probability $\alpha_c(t)$. We could have a similar Bernoulli model with probability $\alpha_d(t)$ for node deletion; upon deleting a node, we also delete all its incident edges. Clearly we could tailor these probabilities to reflect the growth rates of the Web, the half-life of pages, etc.

The edge processes are rather more interesting. We begin with the edge-creation process $\mathcal{C}_e$. At each step we sample a probability distribution to determine a node v to add edges out of, and a number of edges k that will be added. With probability β we add k edges from v to nodes chosen independently and uniformly at random. With probability $1-\beta$, we *copy* k edges from a randomly chosen node to v. By this we mean that we choose a node u at random, and create edges from v to nodes w such that (u, w) is an edge. One might reasonably expect that much of the time, u will not have out-degree exactly k; if the out-degree of u exceeds k we pick a random subset of size k. If on the other hand the out-degree of u is less than k we first copy the edges out of u, then pick another random node u' to copy from, and so on until we have enough edges. Such a copying process is not unnatural, and consistent with the qualitative intuition at the beginning of this section.

A simple edge-deletion process $\mathcal{D}_e$ would again be a Bernoulli process in which at each step, with probability δ, we delete a randomly chosen node. The probability that a particular node v is deleted would ideally be non-increasing in its in-degree.

We illustrate these ideas with a very simple special case. Consider a model in which a node is created at every step. Nodes and edges are never deleted, so the graph keeps on growing. Consider the following edge process: for some $\beta \in (0,1)$, at each step the newly-created node points to a node chosen uniformly at random. With probability $1-\beta$, it copies a uniform random edge out of a random node. Simulations (and heuristic calculations) suggest that under this model, the probability that a node has in-degree i converges to $i^{-1/(1-\beta)}$. Similar calculations suggest that the numbers of cores $C_{i,j}$ are significantly larger than random graphs in which edges go to uniform, independent random destinations.

Clearly the processes creating these graphs, as well as the statistics and structures observed, differ significantly from those of traditional random graph models. This is both a feature and a challenge. On the one hand, the relationship between copying and Zipfian distributions is of intrinsic interest for a variety of reasons (given such distributions also arise in a number of settings outside of the Web — in term frequencies, in the genome, etc.). On the other hand, the process of copying also generates a myriad of dependencies between the random variables of interest, so that the study of such graphs calls for a whole new suite of analytical tools.

5 Conclusion

Our work raises a number of areas for further work:

1. How can we annotate and organize the communities discovered by the trawling process of Section 2.2?
2. What extensions and applications can be found for the connectivity measures discussed in Section 3.3?
3. What are the properties and evolution of random graphs generated by specific versions of our models in Section 4? This would be the analogue of the study of traditional random graph models such as $\mathcal{G}_{n,p}$.
4. How do we devise and analyze algorithms that are efficient on such graphs? Again, this study has an analogue with traditional random graph models.
5. What can we infer about the distributed sociological process of creating content on the Web?

We thank Byron Dom and Ron Fagin for their comments. The work of Jon Kleinberg was supported in part by an Alfred P. Sloan Research Fellowship and by NSF Faculty Early Career Development Award CCR-9701399.

References

1. S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Weiner. The Lorel Query language for semistructured data. *Intl. J. on Digital Libraries,* 1(1):68-88, 1997.
2. R. Agrawal and R. Srikanth. Fast algorithms for mining association rules. *Proc. VLDB,* 1994.
3. G. O. Arocena, A. O. Mendelzon, G. A. Mihaila. Applications of a Web query language. *Proc. 6th WWW Conf.*, 1997.

4. K. Bharat and A. Broder. A technique for measuring the relative size and overlap of public Web search engines. *Proc. 7th WWW Conf.,* 1998.
5. K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. *Proc. ACM SIGIR*, 1998.
6. S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Proc. 7th WWW Conf.*, 1998.
7. B. Bollobás. *Random Graphs*, Academic Press, 1985.
8. J. Carrière and R. Kazman. WebQuery: Searching and visualizing the Web through connectivity. *Proc. 6th WWW Conf.*, 1997.
9. S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan and S. Rajagopalan. Automatic resource compilation by analyzing hyperlink structure and associated text. *Proc. 7th WWW Conf.*, 1998.
10. S. Chakrabarti, B. Dom, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Experiments in topic distillation. *SIGIR workshop on hypertext IR*, 1998.
11. S. Chakrabarti and B. Dom and P. Indyk. Enhanced hypertext classification using hyperlinks. *Proc. ACM SIGMOD*, 1998.
12. H. T. Davis. *The Analysis of Economic Time Series.* Principia press, 1941.
13. R. Downey, M. Fellows. Parametrized Computational Feasibility. In *Feasible Mathematics II*, P. Clote and J. Remmel, eds., Birkhauser, 1994.
14. L. Egghe, R. Rousseau, *Introduction to Informetrics*, Elsevier, 1990.
15. D. Florescu, A. Levy and A. Mendelzon. Database techniques for the World Wide Web: A survey. *SIGMOD Record*, 27(3): 59-74, 1998.
16. E. Garfield. Citation analysis as a tool in journal evaluation. *Science*, 178:471–479, 1972.
17. N. Gilbert. A simulation of the structure of academic science. *Sociological Research Online*, 2(2), 1997.
18. G. Golub, C. F. Van Loan. *Matrix Computations*, Johns Hopkins University Press, 1989.
19. M. R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. *AMS-DIMACS series,* special issue on computing on very large datasets, 1998.
20. M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14:10–25, 1963.
21. J. Kleinberg. Authoritative sources in a hyperlinked environment, *J. of the ACM*, 1999, to appear. Also appears as IBM Research Report RJ 10076(91892) May 1997.
22. D. Konopnicki and O. Shmueli. Information gathering on the World Wide Web: the W3QL query language and the W3QS system. *Trans. on Database Systems,* 1998.
23. S. R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins. Trawling emerging cyber-communities automatically. *Proc. 8th WWW Conf.*, 1999.
24. L. V. S. Lakshmanan, F. Sadri, and I. N. Subramanian. A declarative approach to querying and restructuring the World Wide Web. *Post-ICDE Workshop on RIDE*, 1996.
25. R. Larson. Bibliometrics of the World Wide Web: An exploratory analysis of the intellectual structure of cyberspace. *Ann. Meeting of the American Soc. Info. Sci.*, 1996.
26. A. J. Lotka. The frequency distribution of scientific productivity. *J. of the Washington Acad. of Sci.*, 16:317, 1926.
27. A. Mendelzon, G. Mihaila, and T. Milo. Querying the World Wide Web, *J. of Digital Libraries* 1(1):68–88, 1997.
28. A. Mendelzon and P. Wood. Finding regular simple paths in graph databases. *SIAM J. Comp.*, 24(6):1235-1258, 1995.

29. E. Spertus. ParaSite: Mining structural information on the Web. *Proc. 6th WWW Conf.*, 1997.
30. D. Tsur, J. Ullman, S. Abiteboul, C. Clifton, R. Motwani, S. Nestorov, and A. Rosenthal. Query Flocks: A generalization of association rule mining. *Proc. ACM SIGMOD*, 1998.
31. G. K. Zipf. Human behavior and the principle of least effort. *New York: Hafner*, 1949.