Methods for Mining Web Communities: Bibliometric, Spectral, and Flow

Gary William Flake[1], Kostas Tsioutsiouliklis[2], and Leonid Zhukov[1]

[1] Overture Services, Pasadena, CA 91105, USA
   `gary.flake@overture.com`, `leonid.zhukov@overture.com`
[2] NEC Labs, Princeton, NJ, 08540, USA
   `kt@nec-labs.com`

Summary. In this chapter, we examine the problem of Web community identification expressed in terms of the graph or network structure induced by the Web. While the task of community identification is obviously related to the more fundamental problems of graph partitioning and clustering, the basic task is differentiated from other problems by being within the Web domain. This single difference has many implications for how effective methods work, both in theory and in practice. In order of presentation, we will examine bibliometric similarity measures, bipartite community cores, the HITS algorithm, PageRank, and maximum flow-based Web communities. Interestingly, each of these topics relate to one-another in a non-trivial manner.

1 Introduction

A *Web community* can be loosely defined as a collection of Web pages that are focused on a particular topic or theme. Viewed from the framework of traditional information retrieval, the problem of community identification would be expressed in terms of document content or explicit relations between documents. However, given the hyperlinked structure of the Web, the community identification problem can be reformulated so as to exploit the implicit relations between documents that are formed by hyperlinks.

To leverage the existence of hyperlinks, we model the Web as a graph where vertices are Web pages and hyperlinks are edges. While Web pages may be similar in terms of textual or multimedia content, a hyperlink is usually an explicit indicator that one Web page author believes that another's page is related or relevant. By examining the structure of hyperlinks on the Web, one can identify communities of Web pages that are more tightly coupled to each other than they are to pages outside of the community. Using hyperlinks in this manner—independently or in addition to using text—allows for communities to be identified in a manner that is less sensitive to the subtleties of language. Moreover, by keeping text and hyperlinks separate, one can use the two to co-validate each other.

Mining Web communities is an interesting problem viewed from mathematical, scientific, and engineering viewpoints. Mathematically, the problem is interesting because any effective solution must make a compromise between the quality of the solution and the runtime resources of the algorithm. How these two constraints are traded-off touches on deep questions related to fundamental clustering algorithms. Scientifically, identified Web communities can be used to infer relationships between Web communities, the pages that form the communities, and the users that author and visit the community members. Hence, Web community identification can be used to dissect the Web so as to make it a tractable subject of scientific investigation. Finally, from the engineering point of view, Web community identification can be used to build many applications, including portal regeneration, niche search engines, content filters, and personalized search.

So how does community identification differ from mining relations within a database? The Web differs from most databases in the following aspects:

- The Web is completely decentralized in organization and in evolution. Hyperlinks can be strong or weak indicators of relevance, or completely counter-informative in the case of spam; hence, a lack of centralized authorship makes the Web's hyperlinks not only disorganized when viewed on a large scale, but inconsistent when viewed on a small scale (unlike the regularity of relations within a database).
- The Web is enormous [Lawrence and Giles, 1999]. As of 2003, typical search engine index size is approximately two billion documents, which represents a mere fraction of the whole of the Web. Web pages may not be indexed for a variety of reasons, including (1) prohibition due to robot exclusion rules, (2) falling behind corporate or personal firewalls, (3) being reachable only through a search form, or (4) simply not being popular enough (in in-bound link cardinality) for indexing. Assuming a conservative average document size of 10KB, the indexable Web is at least 20TB in size, making it larger than most other databases excepting offline data warehouses.
- The Web is distributed unlike anything else in the world. Not only is the Web distributed over multiple computers, it spans organizations, Internet domains, and continents.

That's the bad news. Worse yet, being related to clustering and partitioning, community identification is NP-complete even with mild assumptions. Superficially, the preceding facts suggest that mining the Web for communities is hopeless. However, there is considerable good news to counter-balance the bad:

- The Web is self-organized and contains a considerable amount of exploitable structure [Flake et al., 2003]. Hyperlink distributions obey power-law relationships [Barabasi and Albert, 1999]; Many hyperlink patterns (Web rings, hierarchical portals, and hubs and authorities) are found re-

peatedly on the Web; Text and hyperlinks are closely correlated; and simple generative network models explain a considerable amount of the Web's structure [Pennock et al., 2002].

- The Web graph is exceedingly sparse. Being read and written (mostly) by humans, typical Web pages have a relative small number of hyperlinks (only dozens) when compared to the numbers possible if the Web graph were dense (billions). As a result, the graph structure of the Web can be compactly represented. Moreover, this sparseness make algorithms much more well-behaved than the worst-case scenario.
- Finally, being structured and sparse, the Web can be successfully mined by approximate techniques that, while technically sub-optimal in solution quality, empirically yield very good results.

The bulk of this chapter is devoted to this last point. We begin with a brief introduction to clustering and to bibliographic similarity metrics, which is followed by an introduction to bipartite community cores. Next, we examine the PageRank and HITS algorithms and see how they relate to spectral properties of the Web graph. We follow with a detailed section on maximum flow methods for community identification, which includes theoretical and experimental results. Finally, we conclude with a discussion on future work in this area. Throughout this chapter, we will see how each of the methods for community identification relate to one another in such a way that each is easier to understand and appreciate when given in the context of the other methods.

2 Background

The goal of clustering is to group "similar" objects together while keeping "dissimilar" objects apart, with "similarity" being a function of the attributes of the objects being clustered. More formally, a clustering of n objects is a partitioning into k different sets so as to minimize some cost of assigning objects into sets. Regardless of the algorithm and the data source, the quality of a clustering can be measured in multiple ways [Fasulo, 1999]. No single measure for cluster quality is perfect [Kleinberg, 2003]; instead, all measures for cluster quality actually reflect a trade-off between how mistakes and sub-optimalities within a clustering are weighted relative to one another.

The task of community identification can be considered a slight simplification of clustering in that one is only required to identify a grouping of items that are similar to some seed set of elements. In other words, community identification is akin to asking "What are the members of the cluster that contains S?" where S is a seed set. Note that a community identification algorithm can easily be turned into a clustering algorithm (and vice-versa), so the distinction between the two is not very significant theoretically. However, in practice, finding a Web community is vastly simpler than finding all Web clusters because of the size of the Web.

It is beyond the scope of this chapter to survey all clustering methods. Instead, we refer the reader to this survey [Murtagh, 1983] and make connections between Web methods and more classical clustering approaches when appropriate.

2.1 Notation

Throughout this chapter we will represent the Web (or a subset of the Web) as a directed graph $G = (V, E)$, with $n = |V|$ vertices in set V, and $m = |E|$ edges in set E. Each vertex corresponds to a Web page, and each directed edge corresponds to a hyperlink. When appropriate, we will refer to an edge by a single symbol, such as $e \in E$ or as a node pair $(u, v) \in E$ with $u, v \in V$. In the latter case, (u, v) is the directed edge that starts at u and ends (points to) v. It may also be necessary to associate a real valued function with an edge, such as a flow function, $f(e) = f(u, v)$, or a capacity function, $c(e) = c(u, v)$.

Graphs can be equivalently represented as an $n \times n$ adjacency matrix, $\mathbf{A}$, with $A_{uv} = 1$ if $(u, v) \in E$ (i.e., page u links to page v) and $A_{uv} = 0$ if no direct hyperlink exist from u to v. Note that all matrices will be written in bold uppercase Roman and vectors in bold lowercase Roman.

The *degree* of a vertex is the number of edges incident on the vertex. The *in-degree* (respectively *out-degree*) of a vertex is the number of in-bound (respectively out-bound) edges incident on the vertex. We denote the in-degree and out-degree of v by d_v^{in} and d_v^{out}, respectively. Other functions or attributes of vertices will be similarly represented by a symbol with a vertex subscript.

3 Bibliographic Metrics & Bipartite Cores

Identification of a Web community can be done in several ways. One of the key distinguishing features of the algorithms we will consider have to do with the degree of locality used for assessing whether or not a page should be considered a community member. On the one extreme are purely local methods which consider only the properties of the local neighborhood around two vertices to decide if the two are in the same community. Global methods operate at the other extreme, and essentially demand that every edge in a Web graph be considered in order to decide if two vertices are members of the same community. We begin with two related local methods.

3.1 Bibliographic Metrics

We started this chapter by noting that a hyperlink between two pages can be an explicit indicator that two pages are related to one another. However, we should also note that not all related pages are linked. In fact, for many Web pages, what they link to and what links to them may be more informative if considered in the aggregate.
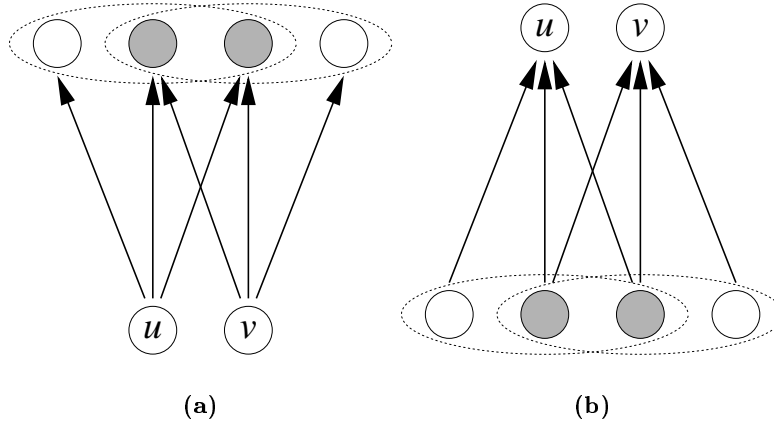
(a)   **(b)**

Fig. 1. Graphic portrayal of bibliographic metrics, **(a)** bibliographic coupling, and **(b)** co-citation coupling. For vertices u and v, the similarity metric is shown as the amount of overlap that vertices have in the set of in-bound neighbors or out-bound neighbors.

Figure 1 illustrates two complementary metrics known as *bibliographic coupling* and *co-citation coupling*. In the figure, we see that the two metrics count the raw number of out-bound or in-bound references, respectively, shared by two pages u and v. Both metrics were originally formulated to capture similarity between scientific literature [Kessler, 1963, Small, 1973] by comparing the amount of overlap between the bibliographies or referrers for two different documents [Garfield, 1979, White and McCain, 1989].

Considering the Web as graph with adjacency matrix $\mathbf{A}$, the product $\mathbf{A}\mathbf{A}^T$ captures the bibliographic coupling between all page pairs such that $(\mathbf{A}\mathbf{A}^T)_{uv}$ equals the bibliographic coupling between pages u and v. Similarly, the product $(\mathbf{A}^T\mathbf{A})_{uv}$ equals the co-citation coupling between u and v.

While bibliographic metrics are simple, intuitive, and elegantly represented by linear algebra, they have several practical shortcomings. First, both matrices $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ will usually be far more dense than $\mathbf{A}$, making storage an issue. Second, some pages may link to or be linked by thousands of other pages, making them have a large amount of link overlap with a disproportionate number of pages.

While the first issue is not easily solved, the second can be easily addressed with normalization. We explain the normalization step in terms of out-bound hyperlinks (i.e., a normalized form of bibliographic coupling) but the basic method can be generalized to hyperlinks in either direction. Let U and V denote sets of vertices that are out-bound from pages u and v, respectively. The quantities:

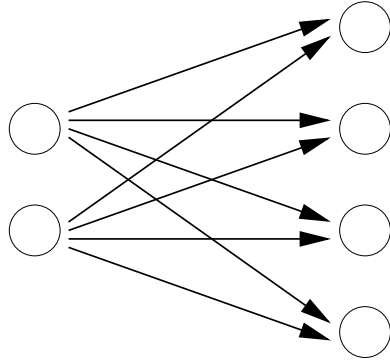$$\frac{2\,|U \cap V|}{|U| + |V|} \text{ or } \frac{|U \cap V|}{|U \cup V|} \tag{1}$$

Fig. 2. Graphic portrayal of a complete bi-partite graph, G_{24}, with each vertex on the left (set L) linking to each vertex on the right (set R). As a bipartite core, G_{24}, could be embedded within a larger graph that has edges that obscure the bipartite core, for example, linking from vertices in R to L, R to R, L to L, or $V - (L \cup R)$ to $(L \cup R)$.

range between 0 and 1 and represent the normalized number out-bound links shared between u and v. One can also generalize normalization methods from text processing, such as $TF \times IDF$ to work for hyperlinks [Giles et al., 1998], or even go so far as to use the Pearson correlation between two rows in $\mathbf{A}$.[3]

3.2 Bipartite Cores

Bibliographic metrics (especially when normalized) are effective for characterizing the degree of similarity between two pages in terms of what they link to and what links to them. What is missing in this framework is the notion that a collection of pages can be related to each other in an aggregate sense.

A *complete bipartite graph* is a directed graph with vertices that can be divided into two sets, L and R (for *left* and *right*) with $L \cup R = V$ and $L \cap R = \emptyset$, such that each vertex in L has an edge to each vertex in R. We use the notation, G_{lr} to denote a complete bipartite graph with $l = |L|$ and $r = |R|$. A *bipartite core* is a subgraph that when considered by itself, forms a complete bipartite graph. Note that this definition for a bipartite core does not preclude vertices (in either of L or R) from linking to—or be being linked by—other vertices outside of L and R, nor does it prohibit edges flowing from R to L. The definition states that being fully connected from L to R is both necessary and sufficient. Figure 2 shows an example bipartite subgraph, G_{24}.

Bipartite subgraphs are relevant to Web communities for at least two reasons that subtly relate to one another. First, a bipartite core, G_{lr}, has the properties that all vertices in L have a bibliographic coupling value lower-bounded by r and all vertices in R have a co-citation coupling value lower-bounded by l. Thus, bipartite subgraphs consist of vertices that have a minimal degree of similarity in terms of raw bibliographic metrics.

The second reason why bipartite subgraphs are relevant to Web communities is because they empirically appear to be a signature structure of the core of a Web community [Kumar et al., 1999]. Intuitively, authors of Web sites

---

[3] Make each row (or column) zero mean, unit variance, and take their dot product

on the same topic may explicitly chose not to link to one another if the sites are competitive with one another (commercial sites in the same industry) or if they idealistically opposed (pro-X versus anti-X sites). Nevertheless, other Web site authors may chose to link to both competitive sites, thus making the two competitors have a non-zero co-citation coupling. In a similar manner, a reference Web site—say one that contains links to auto companies—will have a link pattern that resembles other auto reference sites. Two general auto references will probably not link to one another, but they will often have a high degree of bibliographic coupling. In this way, the structure of a bipartite core captures the dual nature of Web pages, namely, that general pages link to specific pages, and that pages about the same general topic or specific topic will often have similar references or referrers.

Kumar et al. [Kumar et al., 1999] used all of these empirical observations to devise an efficient procedure to identify bipartite cores from a Web corpus. In a subset of the Web (approximately 200 million Web pages), Kumar et al. found over 100,000 bipartite community cores, some being as large as G_{69}. Interestingly, even the smallest identified cores (G_{33} and G_{35}) were topically focused on an identifiable theme in 96% of the samples examples. Examples topics included:

- Japanese elementary schools
- Hotels in Costa Rica
- Turkish student associations

Hence, the identified community cores were usually topically focused and so specific that they were often not part of any preexisting portal hierarchy. This last point is important because it means that community cores are "natural" in the sense that they are self-organized, and not an artifact of a single individual.

4 Spectral Methods

The previous section focused on local methods where the affinity between two pages could be determined by examining the local region about one or two pages. In this section we focus on global methods that essentially consider all links in the Web graph (or subgraph) to answer the question "do these two pages belong with each other." We begin with a brief survey of linear algebra and eigenvectors.

4.1 Linear Algebra and Eigenvectors

Any real-valued semi-positive $n \times n$ matrix, $\mathbf{M}$, can be equivalently represented as a summation of vector outer-products:

$$\mathbf{M} = \sum_{i=1}^{k} \lambda_i \mathbf{l}_r \mathbf{r}_l^T \tag{2}$$

where l_i and r_i are respectively the ith left and right eigenvectors of M, λ_i is the ith eigenvalue of M, and M has all of the following properties with respect to its eigenvectors and eigenvalues:

$$\lambda_i l_i = M r_i, \tag{3}$$

$$\lambda_i r_i = M^T l_i, \tag{4}$$

$$l_i^T l_i = r_i^T r_i = r_i^T l_i = 1, \text{ for all } i, \tag{5}$$

$$l_i^T r_j = 0, \text{ for } i \neq j, \text{ and} \tag{6}$$

$$\lambda_i \geq \lambda_{i+1} \text{ for all } i. \tag{7}$$

The eigenvalues and eigenvectors form the *spectrum* of a matrix; hence, the reason why algorithms that make use of eigenvectors and eigenvalues are often referred to as *spectral methods*. If the spectrum of a matrix is full (i.e., it contains n distinct eigenvectors), then either the left or right eigenvectors can be used as a basis to express any n-dimensional vector. If M is symmetric, then the left and right eigenvectors of M are identical; however, for an asymmetric matrix, the left and right eigenvectors form a *contravariant basis* with respect to each other.

The key intuition behind the eigen-decomposition of a matrix is that it yields a procedure for compressing a matrix into $k < n$ outer-products, and for expressing matrix-vector products as a summation of inner products.

4.2 HITS

Kleinberg's HITS [Kleinberg, 1998] algorithm (which stands for *hyperlink-induced topic search*) takes a subset of the Web graph and generates two weights for each page in the subset. The weights are usually referred to as the *hub* and *authority* score, respectively, and they intimately relate to the spectral properties of the portion of the Web graph for which the algorithm is being used.

Conceptually, a *hub* is a Web page that links to many authorities, and an *authority* is a Web page that is linked by many hubs. The two scores for each page characterize to what degree a page obeys the respective property. Referring back to Figure 2, we can see that pages on the left side of a dense bipartite core are hubs, and pages on the right are authorities.

HITS is actually performed in two parts. The first is a preprocessing step used to select the subset of the Web graph to be used, while the second part is an iterative numerical procedure. The first part usually proceeds as follows:

1. Send a query of interest to the search engine of your choice.
2. Take the top 200 or so results from the search engine.
3. Also identify all Web pages that are one or two links away (in either direction) from the results gathered in the previous step.

All told, the first part generates a *base set* of Web pages that either contain the original query of interest, or are two links away from a page that does. Of

course, other heuristic constraints need to be used, such as limiting the total number of pages, only considering inter-domain hyperlinks, and changing the size of the initial result set and/or the diameter of the base set.

With the base set of pages being generated, let $G = (V, E)$ refer to this subset of the Web graph (with intra-domain hyperlinks removed)[4] and let $\mathbf{A}$ be this graph's adjacency matrix.

The iterative numerical part of HITS updates two $|V| \times 1$ dimensional vectors, $\mathbf{h}$ and $\mathbf{a}$, as follows, with the initial values of both vectors being set to unity:

$$\mathbf{a} = \mathbf{A}^T \mathbf{h} \tag{8}$$

$$\mathbf{h} = \mathbf{A}\mathbf{a} \tag{9}$$

$$\mathbf{a} = \mathbf{a}/||\mathbf{a}||^2 \tag{10}$$

$$\mathbf{h} = \mathbf{h}/||\mathbf{h}||^2 \tag{11}$$

In plain English, Equation 8 says "let a page's authority score be equal to the sum of the hub scores of the pages that link to it," while Equation 9 says "let a page's hub score be equal to the sum of the authority scores that it links to." The remaining equations enforce $\mathbf{h}$ and $\mathbf{a}$ to maintain unit length.

After iterating the equations, we select the authority pages to be those with the largest corresponding value in $\mathbf{a}$, and the hub pages to be the ones with the largest corresponding value in $\mathbf{h}$.

HITS is a close cousin to the *power method* [Strang, 1980] for calculating the eigenvector of a matrix with the largest eigenvalue (the maximal eigenvector). Both procedures converge to a stable solution with a very small number of iterations. With minimal substitution, we can see that $\mathbf{h}$ and $\mathbf{a}$ converge to the maximal eigenvectors of $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$. Thus, HITS produces a rank one approximation to the raw bibliographic and co-citation coupling matrices.

4.3 PageRank

The PageRank algorithm [Brin and Page, 1998] is motivated by a random walk model of the Web. At any given time step, a random walker exist on a Web page and can do one of two things: (1) with probability ϵ it can "teleport" to random Web page (chosen uniformly from all pages), or (2) with probability $(1-\epsilon)$ it can move from its current location to a random page that it currently points to. If we repeat these steps, each Web page will ultimately have a stable and computable probability of being visited by the random walker; this probability is equivalent to the PageRank value of a page.

We can explicitly calculate the PageRank, r_v, of page v with the following iterative procedure:

$$r_v^{t+1} = \frac{\epsilon}{n} + (1-\epsilon) \sum_{u:(u,v)\in E} \frac{r_u^t}{d_u^{\text{out}}}, \tag{12}$$

---

[4] This step reduces the impact of nepotism.

where ϵ is typically set to something small (say 0.1) and superscripts on r_v are used only to indicate time dependencies and that all values should be simultaneously updated. Additionally, after each iteration, $\mathbf{r}$ must be normalized[5] so that $\sum r_v = 1$.

Intuitively, PageRank enforces the recursive idea that pages are important if important pages link to them. PageRank is used as a means to effectively boost the ranking of a Web pages that all match the same query.

PageRank relates to the spectral properties of $\mathbf{A}$ in the following way. Let $\mathbf{M}$ be the equal to $\mathbf{A}$ except that $\mathbf{M}$ has its rows normalized so that they all sum to 1. Let $\mathbf{U}$ be the $n \times n$ matrix with all components equal to $\frac{1}{n}$. The PageRank vector, $\mathbf{r}$ is the maximal eigenvector of:

$$(\epsilon \mathbf{U} + (1 - \epsilon)\mathbf{M})^T \tag{13}$$

provided that G is ergodic, which means that in the limit (as time goes to infinity), the random walker has a non-zero probability of revisiting every page. A Web graph with pages that have no out-bound links or sink regions— regions that once entered cannot be escaped through forward links—break this constraint. Nonetheless, it is helpful to see that PageRank converges to the maximal eigenvector of something like a "well-behaved" version of $\mathbf{A}$ (where "well-behaved" means ergodic and out-bound weight normalized).

4.4 Discussion

Interestingly, HITS has a random-walk interpretation similar to that of PageRank [Lempel and Moran, 2000]. A HITS random walker does not need the "teleport" step, but it does need to alternate between two different types of moves done relative to its current location: (1) follow a random out-bound link, (2) follow a random in-bound link (i.e., as if to reverse the direction). If these moves are repeatedly done in order (move (1) followed by move (2)), then after each pair of moves, the random walker will be at a page that has some non-zero amount of bibliographic coupling with the page that it was in the previous step. The greater the bibliographic coupling, the greater the probability that the random walker will end up at a page. If we were to run this procedure forever, then the probability of visiting a page would be proportional to the hub score of a page (with scaling differences due to normalizing to sum to 1 instead of sum of squares to 1).

If we re-order the moves and do (2) followed by (1), then the random walker will have behavior that is characterized by the authority score vector. In this scenario, the probability of moving from one page to another is related to the amount of raw co-citation coupling between two pages.

In either case, it is interesting to see that both HITS and PageRank can be described in terms of the spectral properties of an adjacency matrix as well the long-term behavior of a random walker. However, at first glance, neither HITS

---

[5] The normalization step is required only when G is not ergodic.

nor PageRank appear to be methods for community identification. HITS uses an initial query to limit the neighborhood of the Web graph from which to score pages, and PageRank is usually coupled to a secondary text retrieval step that is used to filter results. Hence, in the usual implementation, each method has a crucial dependency on text. However, both HITS and PageRank can be adapted to the problem of community identification.

HITS Communities

Recall, from Equation 2, that a matrix can be rewritten as a summation of outer products. Because both of the matrix products $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$ are symmetric and positive definite, each will have the property that the left and right eigenvectors will be identical (because of symmetry) and that the first eigenvector will have all positive components (with positive eigenvalue).

All other eigenvectors for these matrices can be heterogeneous in that their elements can have mixed signs. These subsequent eigenvectors can be used to separate pages into different communities in a manner related to more classical spectral graph partitioning [Chung, 1996], or in a manner that is related to principal component analysis [Jolliffe, 1986].

Kleinberg [Kleinberg, 1999] and his collaborators [Gibson et al., 1998] have found that the non-maximal eigenvectors can be used to split pages from a base set into multiple communities that contain similar text but are dissimilar in meaning. Example include:

- *Abortion:* with communities split along pro-choice, and pro-life camps;
- *Jaguar:* with communities split among the various meanings (football team, animal, car, operating system, video game, etc.);

In this manner, HITS, can be adapted for community identification. The main caveat to spectral approaches is that as the size of the communities get smaller, the less significant eigenvectors can be dominated by noise and confused by paths of longer length. Nevertheless, the approach has considerable power and spectral methods offer a very elegant mathematical derivation.

PageRank Communities

PageRank can generalized into a form known as *topic sensitive PageRank* [Haveliwala, 2002] that replaces the uniform "teleportation" term in Equation 12 with a non-uniform term:

$$r_v^{t+1} = \epsilon p_v + (1 - \epsilon) \sum_{u:(u,v)\in E} \frac{r_u^t}{d_u^{\text{out}}}, \tag{14}$$

with the constraint that $\sum_v p_v = 1$. The value of p_v (for all v) is chosen so that it reflects the probability that the random walker should land on a page, v, during one of its "teleportation" steps.

The intuition behind topic sensitive PageRank is that the random walker moves about as before with the exception that on "teleportation" the walker moves to a page that is focused on some particular topic. For all known pages, v, on the topic, we can set p_v to some non-zero value. In this way, the random walker is similar to a Web surfer that browses as normal, but periodically restarts to a smaller set of favorite bookmarked pages.

This basic procedure can be used to identify a community of pages by using the topic sensitive score to filter pages. Pages that belong to the community are those that have a probability of being visited greater than some threshold; everything else is outside of the community.

5 Maximum Flow Communities

So far, we have characterized community identification methods as being either local or global. In this section we will consider the *Community Algorithm* [Flake et al., 2000, Flake et al., 2002], which has both local and global properties. It can operate over the entire Web graph or a sub-graph; it has worst-case time complexity that's a function of the whole of the Web, yet in practice it is fast because its runtime is often dominated by the size of the community that it finds (and not the whole graph). Moreover, it yields communities that have strong theoretical guarantees on their local and global properties.

The Community Algorithm achieves these properties by recasting the problem into a maximum flow framework. We begin with a subsection describing s-t maximum flows, minimum cuts, and a simple procedure for solving the s-t maximum flow problem. Next, we describe the Community Algorithm along with a few variations of it. We follow with analysis, less formal discussion, and conclude this section with experimental results.

5.1 Max Flows & Min Cuts

While flows and cuts are well-defined for both directed and undirected graphs, we will restrict the domain to undirected graphs to simplify some definitions and to clarify later analysis. Note that any directed graph can be converted into an undirected graph by treating each edge as being undirected. Thus, let $G = (V, E)$ be henceforth understood as an undirected graph, and for all edges $(u, v) \in E$ let $c(u, v)$ denote the capacity of an edge. By convention, we say that $c(u, v) = 0$ if (u, v) does not exist.

Given two vertices, s and t, the s-t maximum flow problem is to find the maximum flow that can be routed from s to t while obeying all capacity constraints of $c(\cdot)$ with respect to G. Intuitively, if edges are water pipes and vertices are pipe junctions, then the maximum flow problem tells you how much water you can move from one point to another.

Ford and Fulkerson's [Ford Jr. and Fulkerson, 1956] "max flow-min cut" theorem proves that the s-t maximum flow of a graph is identical to the

Table 1. The augmenting path maximum flow algorithm.

```
1    procedure MAX-FLOW(graph: G = (V, E); vertex: s, t)
2       set R ⟵ residual network of G
3       while R contains a directed path from s to t do
4          identify shortest augmenting path, P, from s to t
5          set δ = min (r(u, v) : (u, v) ∈ P)
6          for all (u, v) ∈ P  do
7             set r(u, v) ⟵ r(u, v) − δ
8             set r(v, u) ⟵ r(v, u) + δ
9          end for
10      end while
11      return R
12   end procedure
```

minimum cut that separates s and t. The intuition behind the theorem is that flows are limited by bottlenecks, and by removing the same bottlenecks, one can separate two points in a network. Many polynomial time algorithms exist for solving the s-t maximum flow problem and the curious reader should definitely consult a dedicated text for a more thorough discussion of cuts and flows [Ahuja et al., 1993]. flow algorithm to better convey some intuition behind the problem and the algorithm.

Table 1 gives pseudo-code for the augmenting path maximum flow algorithm, which is the simplest maximum flow algorithm known. The procedure operates on a *residual network* which is a data structure used to keep track of edge capacities, both used and available. The residual network, $R = (V, E')$, of G has two directed edges for every undirected edge in E; hence, for $(u, v) \in E$, E' will have both (u, v) and (v, u). The residual capacities in R are initialized by $r(u, v) = r(v, u) = c(u, v)$ for all $(u, v) \in E$.

We say that R has an augmenting path from s to t if there exists a path connecting the two vertices such that each directed edge along the path has non-zero residual capacity. At line 5 of the procedure, we identify the smallest capacity value along the path, P. Lines 6–9 remove the available capacity from the residual network along the path; if $r(u, v)$ becomes zero, we treat that edge as no longer being available. In this way, the procedure simply forces flow from s to t until no more flow can be passed. Finally, at line 11, when there are no more paths from s to t, we return R, which contains sufficient information to easily find the s-t minimum cut or maximum flow of G—but more importantly—R can be used to find a connected component that contains s.
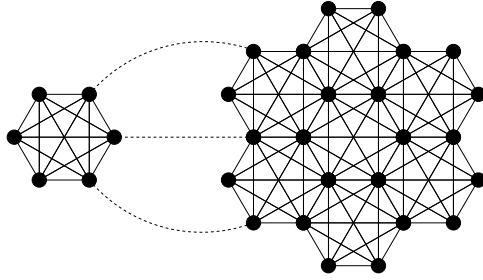
Fig. 3. Maximum flow methods will separate the two subgraphs with any choice of s and t that has s on the left subgraph and t on the right subgraph, removing the three dashed links.

5.2 The Community Algorithm

We now define communities in terms of an undirected graph where each edge has unit capacity. The definition can be generalized for non-unit capacity edges, but cannot be easily generalized for the directed case.

Definition 1. *A* COMMUNITY *is a vertex subset $X \subseteq V$, such that for all vertices $v \in X$, v has at least as many edges connecting to vertices in X as it does to vertices in $(V - X)$.*

Note that this definition is slightly recursive in that it leads to statements of the form "a Pokémon Web site predominately links more Pokémon sites than non-Pokémon sites." Figure 3 shows an example of a community (on the left) being separate from the rest of the graph (on the right).

Interestingly, the questions "Does some graph contain a community?" is NP-Complete [Garey and Johnson, 1979]. However, we will show that there is a polynomial time algorithm that can identify many communities (but not all). This isn't as much of a sacrifice as it would seem because the procedure's limitation is that it can only find communities that have a high degree of intra-community weight and a low degree of inter-community weight, that is, the communities that it is limited to finding are in fact "strong" communities.

The COMMUNITY procedure is shown in Table 2. Its input is a graph G, a set of "seed" Web sites S, and a single parameter, α, which will be explained in greater detail in the next subsection. The procedure creates a new graph, G_α, that has two artificial vertices, s and t. The source vertex, s, is connected with infinite capacity to all pages in the seed set, S. The sink vertex, t, is connected to all original vertices with a small capacity specified by α.

After constructing, G_α, the procedure calls MAX-FLOW as a subroutine, and uses the resulting residual graph to return to portion of R that remains connected to s. This connected component is guaranteed to be a community as defined by Definition 1, provided that the algorithm has not termited with the trivial cut of simply disconnecting all v in S from the rest of the graph.

Table 3 and Table 4 contain two related algorithms, APPROXIMATE-COMMUNITY and COMMUNITY-CLUSTER, respectively (the latter referred to as "cut clustering in [Missing,]), both of which use COMMUNITY as a subroutine. Procedure COMMUNITY is appropriate when the entire graph can be

Table 2. The Community Algorithm.

```
1   procedure COMMUNITY(graph: G = (V, E); set: S; real: α)
2       set V_α ⟵ V ∪ {s, t}
3       set E_α ⟵ E ∪ {(v, t) : v ∈ V} ∪ {(s, u) : u ∈ S}
4       set c(v, t) ⟵ α, ∀v ∈ V
5       set c(s, u) ⟵ ∞, ∀u ∈ S
6       set G_α ⟵ (V_α, E_α)
7       set R ⟵ MAX-FLOW(G_α, s, t)
8       set X ⟵ members of smallest connected component about s in R
9       return X − {s}
10  end procedure
```

Table 3. The Approximate Community Algorithm.

```
1   procedure APPROXIMATE-COMMUNITY(set: S: integer: k)
2       while number of iterations is less than desired do
3           set G to a crawl from S of depth k
4           set α ⟵ |S|
5           set X ⟵ COMMUNITY(G, S, α)
6           rank all v ∈ X by number of edges in X
7           add highest ranked non-seed vertices in X to S
8       end while
9       return X
10  end procedure
```

Table 4. The Community Clustering (or Cut Clustering) Algorithm.

```
1   procedure COMMUNITY-CLUSTER(graph: G = (V, E); real: α)
2       set S ⟵ V
3       while ∃s ∈ S do
4           set X ⟵ COMMUNITY(G, {s}, α)
5           for all v ∈ X do
6               set cluster(v) ⟵ s
7           end for
8           set S ⟵ S − X
9       end while
10      return cluster labels of v ∈ V
11  end procedure
```
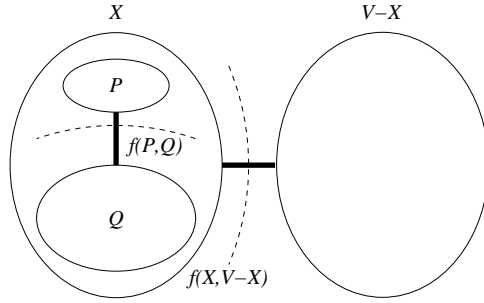
Fig. 4. Inter-community and intra-community cut bounds: $X \subset V$ is any community and $P \cup Q = X$ is any possible partitioning of X. $f(P, Q)$ is an intra-community cut value, and $f(X, V - X)$ serves as an upper-bound on inter-community cuts.

contained in memory and only a single community is required. Procedure APPROXIMATE-COMMUNITY is appropriate when only a small portion of the graph can be contained in memory. It uses a fixed depth crawl to calculate an approximate community, then uses the "strongest" members in the community to be new seeds for a subsequent iteration. Procedure COMMUNITY-CLUSTER can be used to find all communities in a graph. Hence, the last procedure is only appropriate when all communities are desired and the entire graph can be maintained in memory.

5.3 Analysis

Having formally defined the community algorithm, we can now make some rigorous statements regarding the quality of maximum flow communities relative to inter- and intra-community weight. But first, some definitions.

Let the maximum flow value between s and t be represented as $f(s, t)$. We denote the edge cut set that separates s and t with total weight $f(s, t)$ by $C(s, t) \subseteq E$. Removing the cut set $C(s, t)$ from E will always leave at least two connected components: one that contains s and the other that contains t. The maximum flow always has the following relationship to the cut set:

$$f(s, t) = \sum_{(u,v) \in C(s,t)} c(u, v). \tag{15}$$

Finally, we can generalize the meaning of $C(\cdot)$ and $f(\cdot)$ so that their arguments range over sets of vertices. In this case, $C(S, T)$ will be the edge cut set of minimal capacity that separates all vertices in S from all vertices in T, and $f(S, T)$ is the maximum flow or minimum cut value between the two vertex sets.

Recall that the COMMUNITY procedure uses a single parameter, α, which serves as the capacity between all vertices and the artificial vertex, t, added to G. The main theoretical result [Missing,] for the Community Algorithm follows and is made with reference to Figure 4.

Theorem 1. *Let X be a community found by procedure* COMMUNITY *with value α. For any P and Q such that $P \cup Q = X$ and $P \cap Q = \emptyset$, the following bounds will always hold:*

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

Proof of the theorem is beyond the scope of this chapter, but the complete proof can be found in [Missing,].

In a nutshell, Theorem 1 shows that α serves as an upper-bound for the inter-community edge capacity, and a lower-bound for the intra-community edge capacity. Thus, the community algorithm simultaneously guarantees that community members are relatively densely connected to one another but relatively sparsely connected to non-community members.

Also notice that these bounds show how α can be used to tune the size and number of identified communities. A small choice for α, say close to zero, can yield just one community that comprises the entire graph. A large value for α, say $\alpha = 1 + \sum_{(u,v) \in E} c(u, v)$, will yield n singleton communities.

Ironically, the strength of the bounds of the community algorithm is in some sense its main weakness. It is possible that there exists a community that obeys Definition 1 but cannot be found because it fails to obey the bounds in Theorem 1. Nevertheless, this price basically means that the algorithm will only find the best communities, where "best" is in terms of the bounds.

Looking back to Table 4, where the COMMUNITY-CLUSTER procedure is defined, we can make a few more interesting statement about how communities relate to one another. For any two identified communities, X_u and X_v, found by using a seed set of one vertex in each case (u and v), either $X_u \cap X_v = \emptyset$, $X_u \subseteq X_v$, or $X_v \subseteq X_u$ must be true. In other words, communities must nest with one another, forming a hierarchy. This nesting property has been used as a heuristic to speed up the COMMUNITY-CLUSTER [Missing,].

5.4 Discussion

The real trick behind the COMMUNITY procedure is the transformation of G that is performed in lines 2–6 in Table 2. The new graph, G_α, has flow and cut properties that are similar to G. However, the artificial sink, t, is nearly identical to the centroid of G's minimum cut tree (a cut tree is a data structure that preserves all cut and flow properties of a graph; see [Gomory and Hu, 1961] for more details). This means that using t as a sink for an s-t maximum flow calculation is similar to extracting the subtree of G's cut tree that contains s.

Recall that the "teleportation" step in PageRank is required to keep the dynamics of the iterative system well-behaved; without it, excess rank could accumulate to pages that have no out-bound links. Almost the exact same correction could be achieved by introducing an artificial vertex which is connected to every other vertex with some small weight or capacity (and in both directions). Thus, a random walker at any vertex would have some small probability of jumping to the artificial vertex. Similarly, a random walker at the artificial vertex could jump to any other vertex with equal probability. In this
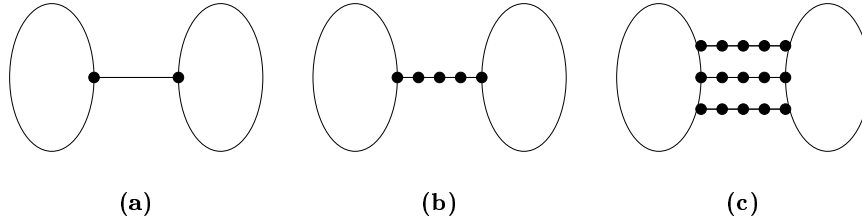
Fig. 5. Three similar graphs with different cut and spectral properties: **(a)** two halves separated by a single short path, **(b)** two halves separated by a single long path, and **(c)** two halves separated by multiples long paths.

way, PageRank is similar to the flow-based Community Algorithm in that both use artificial transitions (via "teleportations" or the artificial edges) to stabilize a calculation.

A significant difference between the flow and spectral methods is in how they treat distances in graphs. HITS and PageRank are sensitive to the distance of the paths that join different parts of a graph. The flow-based Community Algorithm has no sensitivity to path distance whatsoever. For both spectral methods, a page is a community member if the probability of a random walker visiting the page is above some threshold. For the flow method, a page is a community member if it has more unique paths to community members than to non-community members.

These differences are best visualized with the help of Figure 5. To a flow algorithm, (a) and (b) look identical because the two halves can be disconnected by removing one edge (and the flow is also the same). But to a random walker, (a) and (b) look very different because moving between the two halves is much harder in (b) because a long path must be traversed.

By way of comparison, (a) and (c) look very different to a flow algorithm because both the cut and the flow between the two halves are very different. Yet to a random walker, (a) and (c) are similar because the redundant paths in (c) make up for the fact that the individual paths are long.

The spectral and flow methods also differ in that the spectral methods apply a threshold to page scores after the main portion of the algorithm to decide community membership, while the Community Algorithm has its equivalent parameter accounted for at the start of the algorithm. This difference may have implications for both runtime and quality of results.

In the end, each approach has implicit assumptions builtin that make each more practical than many classical clustering or partitioning algorithms; however, the implicit assumptions have implications for how and when each approach makes a mistake.

Table 5. Members of the 9/11 community that contain the word "Howstuffworks".

- Howstuffworks "How Airport Security Works"
- Howstuffworks "How Biological and Chemical Warfare Works"
- Howstuffworks "How Black Boxes Work"
- Howstuffworks "How Building Implosions Work"
- Howstuffworks "How Cell Phones Work"
- Howstuffworks "How Cipro Works"
- Howstuffworks "How Cruise Missiles Work"
- Howstuffworks "How Emergency Rooms Work"
- Howstuffworks "How Machine Guns Work"
- Howstuffworks "How NATO Works"
- Howstuffworks "How Nostradamus Works"
- Howstuffworks "How Nuclear Bombs Work"
- Howstuffworks "How Skyscrapers Work"
- Howstuffworks "How Stun Guns Work"
- Howstuffworks "How the U.S. Draft Works"
- Howstuffworks "How Viruses Work"

Table 6. Members of the 9/11 community that contain the word "Who".

- BBC News — SOUTH ASIA — Analysis: Who are the Taleban?
- BBC News — SOUTH ASIA — Who are the Taleban?
- BBC News — SOUTH ASIA — Who is Osama Bin Laden?
- CNN.com - Backgrounder: Who is bin Laden, al Qaeda? - December 12, 2001
- Countries Need To Plan Effectively for "Deliberate Infections" - WHO Leader Urges Health Ministers
- DefenseLINK News: Bush: No Distinction Between Attackers and Those Who Harbor Them
- EIRC is working on a list of Resources to assist school staff respond to the needs of students and their families who may be impacted
- FEMA: Message to All Who Want to Volunteer or Make Donations from FEMA Director Joe M. Allbaugh
- Forbes.com: Who Is Osama Bin Laden?
- frontline: hunting bin laden: who is bin laden?: a biography of osama bin laden
- Guardian Unlimited — The Guardian — Man whose job is to keep America safe
- Scoop: David Miller: Who is Osama bin Laden?
- THE WORLD TRADE CENTER BOMB: Who is Ramzi Yousef? And Why It Matters - The National Interest, Winter, 1995/96
- U.S. to Assist Those Who Seek a Peaceful, Economically Developed Afghanistan
- Who did it? Foreign Report presents an alternative view
- Who is Osama bin Ladin?
- Who's Clueless?
- Who's OK? - WorldTradeAftermath.com

Table 7. Members of the 9/11 community from CNN.com published in 2002.

- CNN.com - Daniel Pearl, 38, reporter, expectant father - February 22, 2002
- CNN.com - Detainees treated humanely, officials say - January 23, 2002
- CNN.com - Guantanamo Bay in U.S. control over 100 years - January 10, 2002
- CNN.com - Police: Tampa pilot voiced support for bin Laden - January 7, 2002
- CNN.com - States eye high-tech drivers' licenses - February 17, 2002
- CNN.com - Student Bureau: 'Suspicion' - January 21, 2002
- CNN.com - U.S. journalist Daniel Pearl is dead, officials confirm - February 22, 2002
- CNN.com - Will anonymous e-mail become a casualty of war? - February 13, 2002

Table 8. The top 150 text features of the 9/11 Web community. Underscores are used to indicate the occurrence of white space between individual words. A prefix of A, F, E, indicates that the term occurred in the anchor text, full text, or extended anchor text of a page.

1–50	51–100	101–150	151–200
A.terrorism	F.al_qaeda	F.pakistan	A.security
F.terrorist_attacks	A.fbi	A.mil	F.on_september
F.bin_laden	A.wtc	F.the_september	F.u_s_department
E.terrorism	A.laden	A.september	F.crisis
F.taliban	A.attack	F.national_security	F.emergency
F.on_terrorism	F.attacks_on	F.bbc	F.americans
F.in_afghanistan	A.terrorism_http_www	F.arab	A.2001
F.osama	F.afghanistan_and	F.of_war	F.11th
F.terrorism_and	F.attack_on_america	F.state_department	F.the_middle_east
F.osama_bin	F.of_terrorist	F.victims_of	F.troops
F.terrorism	A.bin_laden	F.iraq	F.9_11
F.osama_bin_laden	F.trade_center	F.briefing	E.war
F.terrorist	A.afghan	F.victims	F.threat
E.afghanistan	A.afghanistan	F.tragedy	F.cia
F.against_terrorism	F.of_september_11	A.military	F.the_war
F.the_taliban	F.world_trade	F.attack	A.government
E.terrorist	F.the_pentagon	A.war	F.of_u_s
F.to_terrorism	F.war_against	F.events_of	F.s_department_of
A.state_gov	F.war_on	F.cnn	F.united_nations
F.anthrax	A.defense	F.white_house	F.in_new_york
F.terrorist_attack	F.september_11	F.warfare	A.law
F.world_trade_center	F.president_bush	A.politics	F.center_and
F.the_attack	E.september_11	F.iran	F.enforcement
F.terrorists	A.bbc_co_uk	F.of_defense	A.pubs
A.terrorist	A.bbc_co	A.against	A.federal
E.attacks	F.pentagon	F.department_of_defense	F.2001_the
F.afghan	A.bbc	F.bush	F.defense
F.laden	F.september_11_2001	F.saudi	A.united
F.war_on_terrorism	F.attack_on	F.armed	F.military
A.terror	F.bombing	F.disaster	A.radio
F.afghanistan	F.attacks	F.weapons	F.secretary_of
F.homeland	F.aftermath	F.u_s_government	F.s_department
F.the_world_trade	A.www_state	F.s_government	F.of_international
F.on_america	A.http_www_state	F.law_enforcement	F.justice
F.the_terrorist_attacks	F.department_of_state	F.destruction	A.new_york
A.terrorism_http	F.11_2001	F.the_events	F.of_u
F.the_terrorist	F.islam	F.middle_east	F.congressional
A.emergency	F.red_cross	F.of_state	F.crime
F.of_afghanistan	A.response	F.threats	F.relief
F.the_attacks	F.muslim	F.human_rights	F.killed
F.the_september_11	A.http_news	A.u_s	A.archives
F.september_11th	F.islamic	A.middle	F.coalition
F.wtc	F.muslims	A.asia	A.trade
F.of_terrorism	F.fbi	F.civilian	F.york_times
A.attacks	E.attack	F.speeches	F.new_york_times
A.www_state_gov	F.attack_on_the	F.washington_post	F.nations
F.sept_11	F.september_2001	F.violence	E.september
T.terrorism	F.terror	F.america_s	F.the_nation
F.attacks_on_the	F.foreign_policy	A.america	F.and_security
F.qaeda	F.of_september	F.sept	F.nation

5.5 Experimental Results

To illustrate how the flow-based Community Algorithm works in practice, we used APPROXIMATE-COMMUNITY to identify a set of Web sites dealing with the topic of September 11, 2001 (i.e., the 9/11 community). Ten Web pages were used as seed sites, and the algorithm was run for four iterations. After completion, the community consisted 6257 Web sites.

Tables 5, 6, and 7 show three different samples of the community. Table 5 shows pages from the site Howstuffworks. Table 6 shows all pages that have the word "Who" in the title. Table 7 is a list of all pages from CNN.com published in 2002.

While it is impossible to enumerate all members of the 9/11 Web community, it is interesting to see how highly relevant the members appear to be based on these samples. Had we simply enumerated pages which has the word "Bin Laden" in it, there would be no surprise that some results seem relevant. However, these three tables indicate how the content varies when looking at specific sites, extremely general words, and relatively new additions to the community.

As can be seen, the Howstuffworks result are very relevant to 9/11 but in a very subtle and non-trivial manner. The pages with "Who" in the title appear to have a single outlier ("Who's Clueless"), yet this Web site had was topically focused on the events of 9/11 at the time that this experiment was conducted. Finally, the CNN.com pages show that newer pages in the community maintained relevance to the central theme.

Finally, Table 8 shows the top n-grams found in members of the 9/11 Web community, ranked by their ability to disambiguate community members from non-community members. (In this case, we used the simplistic ratio of the probability of occurring in the 9/11 community versus the probability of appearing in a random set of pages.) As can be clearly seen, the n-grams in the table are all highly related to 9/11 with no obvious outliers. These results support the main claim that one can identify large Web communities that are topically and textually focused with methods that use hyperlinks and no text.

More information on the 9/11 community is available at:

$$\texttt{http://webselforganization.com/example.html}$$

which allows one to browse the entire community and to search the community for specific words.

6 Conclusions

We have explored several different methods for identifying communities on the Web. While all of these methods differ from one another, there are interesting connections between many of the methods; for example:

- Bibliometric methods define a notion of similarity for pages that do not directly link to one another.
- Bipartite cores consist of pages that have high bibliographic metrics with respect to each other.
- HITS identifies hubs and authorities, which are pages that often fulfill the definition of a bibliographic core.
- PageRank and HITS both have a spectral and random walk interpretation relative the Web graph.
- PageRank and the flow-based Community Algorithm, both use a nearly identical artificial transition for stabilizing each respective calculation.

We also saw how some of the approaches related to more traditional clustering algorithms and how in some cases tight theoretical bounds could be derived for global and local community properties.

Further Explorations

There are many open areas within the field of Web data mining that are directly related to the task of identifying Web communities:

Combine Text with Hyperlinks

Probabilistic versions of HITS have been used [Cohn and Hofmann, 2001] that treat hyperlinks and text on equal footing. There is an opportunity to do the same for the flow-based Community Algorithm. In particular, it would be interesting to see if text and hyperlinks could be combined in such a way that accounts for varying word frequencies.

Unite PageRank and the Community Algorithm

The Community Algorithm normally operates over unit capacity edges. However, there has been some interesting success in normalizing edge capacity by out-degree (before discarding edge direction) [Missing,]. An extension of this idea would use PageRank to calculate a probability of an edge transition, then use rescaled probabilities for capacity values. It would also be interesting if this procedure yields more satisfactory bounds on community quality because it may be possible to express the bounds in terms of an absolute probability of moving from one community to another (instead of flow or cut values).

Flow Communities of a Bibliographic Matrices

One may also be able to apply the Community Algorithm to $A^T A$, AA^T, or some other related matrices. It is not clear when or if this would be a benefit, nor what the precise theoretical interpretation of the resulting communities. Nonetheless, these communities may be more appropriate for very sparse graphs.

Generalize the Community Algorithm for Arbitrary Cohesiveness

It has been suggested [Flake et al., 2000] that one can generalize the Community Algorithm so that it finds communities that obey the definition that members have x% of their links to other members, instead of the standard 50%. Generalizing the algorithm in this manner would require more extensive use of artificial vertices and edges; however, it may yield a more satisfying results and more interesting bounds.

Use Randomized Algorithms for Improved Speed

While flow algorithms have been steadily improving, there exist approximate and randomized flow algorithms that may improve performance on large-scale problems. In particular, it would be interesting to implement such an algorithm that could operate on a graph with only linear scans of the adjacency matrix.

Automate the Search for α

Because the Community Algorithm's α parameter reflect a trade-off between community size and the number of communities in a graph, there would be considerable value in automating the search for α. There exists newer flow algorithms (on paper) that can perform this search in time proportional to a single s-t maximum flow calculation; however, there is no implementation known to exist at this time.

Reconcile Bounds for Spectral and Flow Partitions

Finally, there are many well-known bounds for spectral clustering which use the so-called *Laplacian* matrix (instead of the adjacency matrix). It would be of value to reconcile the spectral bounds with the flow bounds.

References

[Ahuja et al., 1993] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows : Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ.

[Barabasi and Albert, 1999] Barabasi, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286.

[Brin and Page, 1998] Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. In *Proc. 7th Int. World Wide Web Conf.*

[Chung, 1996] Chung, F. (1996). *Spectral Graph Theory*. CBMS Lecture Notes. Amer. Math. Soc.

[Cohn and Hofmann, 2001] Cohn, D. and Hofmann, T. (2001). The missing link - a probabilistic model of document content and hypertext connectivity. In *Neural Information Processing Systems 13*.

[Fasulo, 1999] Fasulo, D. (1999). An analysis of recent work on clustering algorithms.

[Flake et al., 2000] Flake, G. W., Lawrence, S., and Giles, C. L. (2000). Efficient identification of web communities. In *Proc. 6th Int. Conf. on Knowledge Discovery and Data Mining*, pages 150–160.

[Flake et al., 2002] Flake, G. W., Lawrence, S., Giles, C. L., and Coetzee, F. (2002). Self-organization of the web and identification of communities. *IEEE Computer*, 35(3):66–71.

[Flake et al., 2003] Flake, G. W., Pennock, D. M., and Fain, D. C. (2003). The self-organized web: The yin to the semantic web's yang. *IEEE Intelligent Systems*. To appear.

[Ford Jr. and Fulkerson, 1956] Ford Jr., L. R. and Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian J. Math.*, 8:399–404.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman, New York.

[Garfield, 1979] Garfield, E. (1979). *Citation Indexing: Its Theory and Application in Science*. Wiley, New York.

[Gibson et al., 1998] Gibson, D., Kleinberg, J., and Raghavan, P. (1998). Inferring web communities from link topology. In *Proc. 9th ACM Conf. on Hypertext and Hypermedia*.

[Giles et al., 1998] Giles, C. L., Bollacker, K., and Lawrence, S. (1998). CiteSeer: An automatic citation indexing system. In Witten, I., Akscyn, R., and Shipman III, F. M., editors, *Digital Libraries 98 - The Third ACM Conference on Digital Libraries*, pages 89–98, Pittsburgh, PA. ACM Press.

[Gomory and Hu, 1961] Gomory, R. E. and Hu, T. C. (1961). Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570.

[Haveliwala, 2002] Haveliwala, T. (2002). Topic-sensitive PageRank. In *Proceedings of the Eleventh International World Wide Web Conference*.

[Jolliffe, 1986] Jolliffe, I. (1986). *Principal Component Analysis*. Springer Verlag.

[Kessler, 1963] Kessler, M. (1963). Bibliographic coupling between scientific papers. *American Documentation*, 14:10–25.

[Kleinberg, 2003] Kleinberg, J. (2003). An impossibility theorem for clustering. In *Advances of Neural Information Processing Systems 15*, Cambridge, MA. MIT Press.

[Kleinberg, 1998] Kleinberg, J. M. (1998). Authoritative sources in a hyperlinked environment. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677.

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

[Kumar et al., 1999] Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. (1999). Trawling the web for emerging cyber-communities. In *Proc. 8th Int. World Wide Web Conf.*

[Lawrence and Giles, 1999] Lawrence, S. and Giles, C. L. (1999). Accessibility of information on the web. *Nature*, 400(6740):107–109.

[Lempel and Moran, 2000] Lempel, R. and Moran, S. (2000). The stochastic approach for link-structure analysis (SALSA) and the TKC effect. *Computer Networks (Amsterdam, Netherlands: 1999)*, 33(1–6):387–401.

[Missing,] Missing. Missing Reference to Come.

[Murtagh, 1983] Murtagh, F. (1983). A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4):354–359.

[Pennock et al., 2002] Pennock, D. M., Flake, G. W., Lawrence, S., Glover, E. J., and Giles, C. L. (2002). Winners don't take all: Characterizing the competition for links on the web. *Proceedings of the National Academy of Sciences*, 99(8):5207–5211.

[Small, 1973] Small, H. (1973). Co-citation in the scientific literature: A new measure of the relationship between two documents. *J. Am. Soc. for Inf. Sci.*, 24(4):265–269.

[Strang, 1980] Strang, G. (1980). *Linear algebra and its applications.* Harcourt Brace Jovanovich, San Diego.

[White and McCain, 1989] White, H. and McCain, K. (1989). Bibliometrics. In *Ann. Rev. Info. Sci. and Technology*, pages 119–186. Elsevier.