

# Summary of

## Efficient Identification of Web Communities

**Mining the World Wide Web** is an interesting research area in these years.

This paper defines a community on the web. A **community** is a set of websites, which has more links to the members inside the community than the links to the members outside. Members of such a community can be efficiently identified in a maximum flow minimum cut framework. A **focused crawler** that crawls to a fixed depth can approximate community membership by augmenting the graph induced by the crawl with links to a virtual sink node.

### Motivation:

The rapid growth of the World Wide Web has created a dilemma for search engine designers while it ushered a boon to most levels of society. They have to balance a number of conflicting goals:

- Valid result and Huge indexable web pages
  - Indexable web should be broadly enough to insure result are valid
  - No search engine covers more than about 16% of the web.
- The complexity of Web Pages is far greater than that of any traditional text document collection.
- There often exists a balance between precision and recall of query result.
  - High Recall & Low Precision: Broad queries to general search engines return thousands of results
  - Low Recall & High Precision: Organize a small subset of the web into a hierarchic architecture. (Yahoo Approach)

$$precision = \frac{|\{Relevant\} \cap \{Retrieved\}|}{|\{Retrieved\}|} \quad recall = \frac{|\{Relevant\} \cap \{Retrieved\}|}{|\{Relevant\}|}$$

### Contribution:

- Introduces a definition of a web community that may enable web crawlers to effectively focus on narrow but topically related subsets of the web. And

proves that a community can be identified in a max flow-min cut framework.

- Provides a focused-crawl algorithm to approximate community membership, which can increase the precision and recall of search results.

### **Relevant Prior work:**

This paper's work is based on three different fields:

- Link Analysis (Google's Approach)
  - Hubs and Authorities:  
An authority is one web page that is given wide endorsement by different authors on the web. Ex. Globus for grid computing.  
A hub is one page that provides collections of links to authorities. Ex. personal web page.  
Hub links to many authorities, Authority links to many hubs
  - How to find authorities and hubs?  
Using HITS algorithms, it will be used to find "seeds" for authors' own algorithm.  
Their limitations are:
    - An interesting community may be overshadowed by a more dominant community.
    - It may be limited use in identifying communities that form web rings and networks without dominating members.
- Graph cuts and partitions  
Identifying the community is equivalent to partition a graph such that the edge weight between the partitions is minimized while maintaining partitions of a minimal size (balanced minimal cut). Its limitation is
  - The problem is NP complete.
- Maximum Flow and Minimum Cuts

Maximum Flow: Given a directed graph  $G=(V,E)$ , with edge capacities  $c(u,v)$  belongs to  $Z^+$ , and two vertices,  $s,t$  belongs to  $V$ , find the maximum flow that can be routed from the source,  $s$ , to the sink,  $t$ , that obeys all capacity constraints.

Minimum Cut: A "Cut" is a line (or a Collection of lines) that separates the *source* node to the *sink*(end) node. A minimum cut is a cut, its  $\{cut(S, T)-f(S,T)\}$  is minimum.

Max flow-min cut theorem of Ford Fulkerson proves that maximum flow of the network is identical to the minimum cut that separates  $s$  and  $t$ .

- Computationally tractable and many polynomial time algorithms exist for solving this.
- Shortest Augmentation path algorithm (extension of the above) can meet the memory space limitation problems imposed by the vast size of web

### Authors' Work:

- What is a Web Community?  
Community: A COMMUNITY is a vertex  $C$  is a subset of  $V$ , such that for all vertices  $v$  belongs to  $C$ ,  $v$  has at least as many edges connecting in  $C$  as in does to vertices in  $(V-C)$ .
- How to identify ideal communities?  
➤ **Theorem 1:** *A Community  $C$  can be identified by calculating the  $s$ - $t$  minimum cut of  $G$  with  $s$  and  $t$  being used as the source and sink, respectively, provided that both  $s^\#$  and  $t^\#$  exceed the cut set size. After the cut, vertices that are reachable from  $s$  are in the community.*

Source Link Count,  $s^\#$ , refers to the number of edges between  $s$  and all vertices in  $(C-s)$ .

Sink Link Count,  $t^\#$ , refers to the number of edges between  $t$  and all vertices in  $(V-C-t)$ .

From theorem1 and the famous “max flow-min cut” theorem, we can use maximum flow algorithm to extract a closely-knit community from the web.

- Choice of “seeds”: Using HITS to discover a group of authorities and hubs on index-based web pages.
- Choice of “sink”: Using a small collection of web portals as a virtual sink because they should be very close to the center of the web graph.

- Approximate Communities:

One problem with calculating ideal communities is that it requires rapid access to the inbound and outbound links for many web sites. We cannot store so much information in a single computer. So our focused crawler crawls only to a fixed depth to get an approximate community. See Figure 2. We use virtual sink instead of a true graph center and justify that the flow methods with the virtual sink yield the same partitioning as with the true graph center under some conditions.

**Theorem 2:** Let  $c$  denote the total capacity of the original cut set and let  $N_s$  and  $N_t$  denote the number of vertices in the community,  $C$ , and not in the community,  $(V-C)$ . If the condition  $1 < k < N_t / c$  holds, then the augmented graph as shown in Figure 3 will produce the same partitioning as the un-augmented graph, except that all edges from  $C$  to the virtual sink will be cut as well.

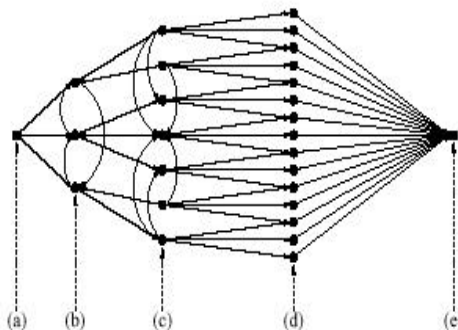


Figure 2: Focused community crawling and the graph induced: (a) The virtual source vertex; (b) vertices of seed web sites; (c) vertices of web sites one link away from any seed site; (d) references to sites not in (b) or (c); and (e) the virtual sink vertex.

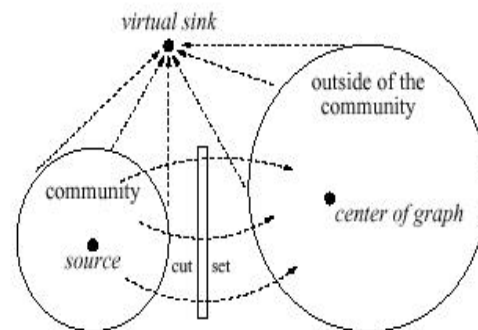


Figure 3: Locating a cut even when a good sink is not available: All edge capacities in the graph are multiplied by a constant factor,  $k$ . Unit capacity edges are added from every vertex to a new virtual sink.

- Algorithm of crawl procedure:

With a small number of seeds, only a relatively small subset of a community can be identified because the combined out degree of the seeds has to be larger than the cut size (the number of edges removed). Thus, we augment our procedure with a method for identifying new seeds. See the following procedure. We typically run the procedure for four iterations, we also give the Ford-Fulkerson algorithm to show how to identify a community. In Ford-Fulkerson algorithm, selection of augmentation path is important. If it is chosen poor, it may not converge to the maximum flow value. We use shortest augmentation path to avoid it. And we use BFS to search the shortest path because it only uses portions of the graph.

Procedure **Focused-Crawl** (graph:  $G=(V,E)$ ; vertex:  $s, t$ , belongs to  $V$ )

- (1) while number of iterations < desired do
  - (2) Set  $k$  = number of vertices in the seed set.
  - (3) Perform maximum flow analysis of  $G$ , yielding community,  $C$ .
  - (4) Identify non-seed vertex,  $v^*$  belongs to  $C$ , with the highest in-degree relative to  $G$ .
  - (5) For all  $c$  belongs to  $C$  such that in-degree of  $v$  equals to  $v^*$ 
    - Add  $v$  to seed set.
    - Add edge( $s,v$ ) to  $E$  with infinite capacity.
  - end for
  - (6) Identify non-seed vertex,  $u^*$ , with the highest out-degree relative to  $G$ .
  - (7) For all  $u$  belongs to  $C$  such that out-degree of  $u$  equals to  $u^*$ 
    - Add  $u$  to seed set.
    - Add edge( $s, u$ ) to  $E$  with infinite capacity.
  - End for
  - (8) Re-crawl so that  $G$  uses all seeds.
  - (9) Let  $G$  reflect new information from the crawl.
- End while
- end procedure.

Procedure **Ford-Fulkerson** (graph:  $G=(V,E)$ ; vertex:  $s, t$ , belongs to  $V$ ):

- (1) For each edge( $u, v$ ) belong  $E[G]$ 
  - $f[u, v] = 0$
  - $f[v, u] = 0$
- (2) While there exists a path  $p$  from  $s$  to  $t$  in the residual network  $G_f$ 
  - $C_f(p) = \min \{C_f(u, v): (u, v) \text{ is in } p\}$
  - For each edge( $u, v$ ) in  $p$

$$f[u, v] = f[u, v] + C_f(p)$$

$$f[v, u] = -f[v, u]$$

step(2) usually is called augmenting step.

For examples, please see the slides.

### Analysis of Algorithm:

What is the run time for Focused-Crawl?

It has been proved that

Step(3) (BFS) for shortest path,  $O(VE^2)$ , others:  $O(V^3)$

Step(4), step(6):  $O(V)$

Step(8):  $O(E)$

Total: (m iterations):  $O(VE^2)$ ,  $V = \max\{1 \leq i \leq m \mid V_i\}$ ,  $E = \max\{1 \leq i \leq m \mid E_i\}$

### Experimental Results:

The authors test their algorithm from three different groups of initial web pages.

Each community are closely related to the interested field:

- Support Vector Machine Community:  
Graph Size: 11000 Community Size: 252 Results: strong related to SVM research
- The Internet Archive Community:  
Graph Size: 7000 Community Size: 289 Results: closely related to the mission of the IA
- The “Ronald Rivest” Community:  
Graph Size: 38000 Community Size: 150 Results: closely related to his research

### Our opinion:

We have spent so much time in understanding this paper. It is tough for a beginner. We need to read many other materials to understand some basic concepts in this paper. This paper focused on the research in Web Information

Retrieving, trying to improve the search result both in precision and recall. The discovered communities are very cohesive.

- Advantage:
  - This paper exposes us towards a bunch of advanced technologies and they are applied in authors' work.
  - Authors give us their algorithm to identify a community and also justify why they can do it.
- Limitations:
  - There is no algorithm analysis in this paper.
  - There is no comparison with other search engine, although there is no standard benchmark for it.

Based on the discussion above, we give this paper a rating 8 out of 10 with the consideration of the novelty and logical construction, organization of ideas, the style and presentation of the authors, and authors' willingness to share the reference materials.

Authors wish to improve their algorithm by combine it with other text-based methods. We believe that it is a nice idea.

## **Suggestions for Improvements:-**

Beyond that, We are interested in further investigating the algorithm when we do the following modification: In step 4 and step 6, we want to use the fixed enough value of  $s^\#$  and  $t^\#$  instead of the largest, to see whether it can control the size of community. And in step 1, we can decide the end of iteration dynamically by comparing community under some measurement. We also want to expend the depth of crawler and try to figure out whether the result is improved, and how much is improved.

At the same time, we realize that a successful search of community is related to the initial seeds group, which may be limited by the storage space of a single computer, and also in reality a community may be divided into several small sub-communities, which are more cohesive inside than outside. To run a sequential

algorithm may converge into one sub-group. Useful information in other sub-group community may be missed. So we put forward to a parallel algorithm based on the focused crawler.

### **Parallel Focused-Crawl Algorithm**

Procedure Focused-Crawl(graph:  $G=(v,E)$ ; vertex:  $s,t$ , belongs to  $V$ )

For each processor, par-do

- (1) while number of iterations < desired do
- (2)   Set  $k$  = number of vertices in the seed set.
- (3)   Perform maximum flow analysis of  $G$ ,  
          yielding community,  $C$ .
- (4)   calculate in-degree and out-degree link for each vertex
- (5)   choose several vertex according to measurement to neighbor processor
- (6)   add some vertex from input vertex to seeds group
- (8)   Re-crawl so that  $G$  uses all seeds.
- (9)   Let  $G$  reflect new information from the crawl.  
      End while
- (10) combine community according to other measurement  
end procedure.