

An Exhaustive and Edge-removal Algorithm to Find Cores in Implicit Communities

Nan Yang, Songxiang Lin, Qiang Gao

The School of Information, Renmin University of China,
No 59, Zhongguancun Street, Beijing, China
8601-82500902
yangnan@ruc.edu.cn

Abstract. Web community is intensely studied in web resource discovery. Many literatures use core as the signature of a community. A core is a complete bipartite graphs, denoted as $C_{i,j}$. But discovery of all possible $C_{i,j}$ in the web is a challenging job. This work has been investigated by trawling [1][2]. Trawling employs repeated elimination/generation procedure until the graph is pruned to a satisfied state and then enumerate all possible $C_{i,j}$. We proposed a new method that uses exhaustive and edge removal method. Our algorithm avoids scanning dataset many times. Also, we improve crawling method by only recording potential fans to save disk space. The experiment result show that the new algorithm works properly and many new $C_{i,j}$ can be found by our method.

Keywords: Web communities, Link analysis, Complete Bipartite Graph

1 Introduction

Web is a huge information resource and increases dramatically. Although the growth of web seems chaos, but in fact web shows a great deal of self-organization [3]. Web communities are very important structure in web. There are well-known, explicitly-defined communities, for example, users interested in mercedes-benz cars or in java development. Most of them manifest themselves as newsgroups, webrings, or as resource list in directories such as Yahoo! and Infoseek [1]. Definitely, the web communities are set of pages which created by a group or people with common interest. There many literatures had got involved in web communities, for example, HITS [4][5], Companion[6][7], max flow/min cut[8] and trawling algorithm[1][2]. But there are many communities are implicit and their number overcomes that of explicit ones. Trawling algorithm mainly focused on implicit communities. In this paper, we have analyzed some forms of structure which are not considered by trawling. Then by borrowing edge-removal idea of Newman [9][10][11], we introduced a new extraction algorithm. After a subgraph collected from web graph, we then check the possible cores in it. At each process some edges are removed from web graph. We repeat this process until the web graph is empty. Our background is same as trawling method. But our method is different from trawling in several ways. First, we improve the crawling by only recording the potential fans so that we can

save a lot of disk space. Second, we extract cores based on edge removal which reduce the times of scanning dataset. Like Kumar [2], we also use term frequency to evaluate whether or not the potential cores can organize communities. The outline of the paper is as follow. In section 2, we introduce some basic related knowledge. In section 3, we describe the preparing procedure of dataset and link database. In section 4, we introduce new algorithm. In section 5, we describe topics of communities. In section 6, we arrange the dataset and experiment and some result examples. Conclusions and future works are shown in section 7.

2 Preliminaries

Web can be abstracted to a large directed graph $G=(V, E)$. V is the set of nodes, E is the set of edges. A pair of nodes $(u, v) \in E$ means that there is a hyperlink between u and v . A bipartite graph is a graph whose vertex set can be partitioned into two sets, which we denote F and C . Every directed edge in the graph is directed from a vertex u in F to a vertex v in C , depicted in Fig.1(a). A bipartite graph is dense if many of the possible edges between F and C are present. The trawling algorithm is based on the hypothesis: the dense bipartite graphs that are signatures of web communities contain at least one core. A core is a complete bipartite graph with at least i vertices in F and at least j vertices in C . Thus, the core is a small (i, j) -sized complete bipartite graph, denoted as $C_{i,j}$. We will find a community by finding cores, and then use the cores to find the rest of the community. According to Rajagopalan [12], the data mining graph is bipartite with left hand side and right hand side, denoted as LHS and RHS respectively.

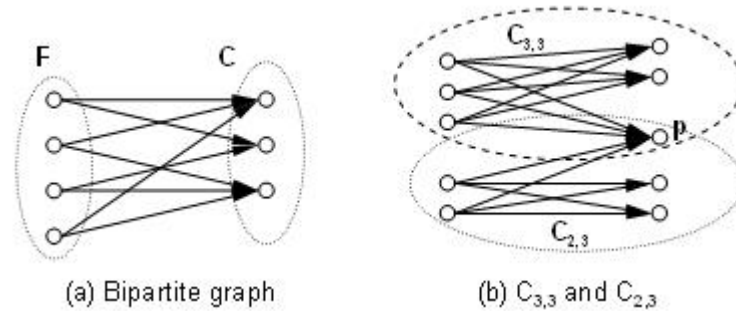


Fig. 1. (a) Bipartite Graph. (b) $C_{2,3}$ and $C_{3,3}$.

3 Dataset Preparation

The web pages are collected by a web crawler [13]. We only extract urls information from html text. We extract static links which begin with “http://” and are included within “”. The length of url is limited to below 100 characters and we don’t repeatedly extract the links belong to same domain. For instance, while

www.edu.cn/xxx and www.edu.cn/yyy arrive, we only reserve www.edu.cn/xxx. An edge have a form of pair <source url, destination url> and set of edges are stored into edge file.

We use 128bit MD5 hash fingerprint as id to record an url. Every edge will occupy 32bytes. To get in-degree and out-degree of a page, we use two datasets to present web graph. One is the set of edges ordered by source id, denoted as DSL and other by destination id, denoted as DSR. Due to a set of edges stored contiguously in DSL(DSR), it is easy to get out-degree(in-degree). We employ BerkeleyDB to manage dataset and use its BTREE access mode with sorted key. This mode maintain key in sorted order automatically.

Because of mirror web sites and duplicated pages, there are many pages are derived from same resources [14]. Many of them are highly inter-connected and tend to form community structure. These structures are value resources but help little to find implicit communities. Hence, deletion of these pages before core extraction is necessary. There are many algorithms to delete mirror or near-duplicated pages. We chose a method as in [9]. Pages with out-links above 8 are taken into account and their common out-links is exceed to 85 percent, to say the two pages are mirror.

Many researches have showed that the distribution of in-degree obeys power-law [3]. This law was used in many community discovery algorithms to prune the dataset. The pages with very low in-links and high in-links are pruned. Too low in-links means that page is less important. And too high in-links means that page belong to popular web sites, such as www.google.com, www.yahoo.com etc.

4 Algorithm on Edge Removal

4.1 Defects of Trawling Algorithm

The complete directed bipartite graph is a metaphor of community. We call complete bipartite graph as a core, denoted as $C_{i,j}$, where i and j are nodes in fans and centers. How to find all possible cores is one of the important problems. In [1][2], Kumar employs the criterion of a core. Consider 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.

The trawling algorithm has three defects. First, some $C_{i,j}$ s will be missing in certain subgraph. Let's look subgraph in Fig. 1(b), node p belongs to both $C_{3,3}$ and $C_{2,3}$. Unfortunately, node p has in-degree 5, according to trawling criterion, it would be pruned when find $C_{3,3}$ and $C_{2,3}$ because p does not have in-degree 3. If node p is pruned, the core $C_{3,3}$ and $C_{2,3}$ will be missing. Second, the removal of nodes will destroy the structure of other cores. For example, in Fig. 1(b), when we find $C_{3,3}$ all nodes related to $C_{3,3}$ are removed. Node p is also removed and the structure of $C_{2,3}$ is destroyed. Third, the enumeration by combining i and j is at high cost because every scan of dataset only fans with out-degree j and centers with in-degree i are taken into account. The next subsection is the new algorithm which is proposed to overcome these shortcomings.

4.2 Exhaustive Algorithm and Removal of Edges

Derived from [4][5][6], they use node removal method to find communities in a graph. We propose a new algorithm to find all possible $C_{i,j}$ s from a subgraph. First, we employ an exhaustive idea to find all possible cores in one scan of dataset. Second, we delete edges instead of nodes. Every time scan, we will construct a bipartite subgraph from a chosen node p . Then an exhaustive algorithm will extract all $C_{i,j}$ s and then related edges are removed. Our method avoids scanning whole dataset whenever a set of edges are deleted. Any node will not be deleted unless all edges associated are deleted.

Before we describe our algorithm, some definitions and notations should be given first. BG is a bipartite graph. We use $L(BG)$ to denote LHS of BG and $R(BG)$ to denote RHS of BG. We use $C(x, BG)$ to denote the nodes in $R(BG)$ pointed by x and $P(x, BG)$ to denote the nodes in $L(BG)$ pointing to x . $S(p, BG)$ denote the nodes in $L(BG)$ that point to the set of nodes in $R(BG)$ which are also pointed by node x . Let BG_w to denote the web graph, BG_s to denote constructed bipartite graph and BG_c denote the bipartite graph of $C_{i,j}$ and contain node p .

Definition BG_s: For any given node $p \in L(BG_w)$, BG_s is a bipartite subgraph constructed from p , where $L(BG_s) = \{p, S(p, BG_w)\}$ and $R(BG_s) = C(p, BG_w)$.

Definition BG_c: For any given $C_{i,j}$ and node p , BG_c is a bipartite subgraph of $C_{i,j}$, where $L(BG_c) = \text{LHS of } C_{i,j}$, $R(BG_c) = \text{RHS of } C_{i,j}$ and $p \in L(BG_c)$.

The **Definition BG_s** explains that procedure of constructing BG_s has two steps. Step 1 is to get the children of p in BG_w to construct LHS of BG_s . Step 2 is to get the siblings of node p , and then use p and its siblings construct RHS of BG_s .

Theorem 1: If BG_s is a constructed bipartite subgraph of node p , then all possible $BG_c \subseteq BG_s$.

Proof: For any $u \in L(BG_c)$, we have $C(u, BG_c) = R(BG_c)$, likewise for any $v \in R(BG_c)$ we have $P(v, BG_c) = L(BG_c)$. Because $p \in L(BG_c)$, $C(p, BG_c) = R(BG_c)$. BG_s is constructed from BG_w and the $L(BG_s)$ is all children of node p in BG_w , therefore $R(BG_c) \subseteq R(BG_s)$. For any $q (q \neq p, q \in L(BG_s))$, we have $C(q, BG_c) = R(BG_c)$, so q is sibling of p . Because $L(BG_s)$ is p and all its siblings derived from BG_w , $L(BG_c) \subseteq L(BG_s)$.

According to **Theorem 1**, two-step construction of subgraph is complete to include all possible $C_{i,j}$ s. Then next job is how to extract all possible BG_c from BG_s . In trawling, Kumar use enumeration by combining of i and j , from intersection of potential fans with j degree or centers with i degree to find $C_{i,j}$. Here we don't use enumeration method by employ a function. We introduce a function $U(x, y, t)$ and $V(z, x, y, t)$, which have following form:

$$U(x, y, t) = \begin{cases} x \cap y; & |x \cap y| \geq t \\ x; & \text{else} \end{cases} \quad (1)$$

$$V(z, x, y, t) = \begin{cases} z \cup y; & |x \cap y| \geq t \\ z; & \text{else} \end{cases} \quad (2)$$

Here x, y and z are sets. t is an integer ($t > 0$) to give a threshold value.

We definition two sets as H, L and $p_1 = p$. Let $S(p_1, BGs) = \{p_2, p_3, \dots, p_n\}$. For a certain t , we have:

$$\begin{aligned} H_1 &= C(p_1, BGs); L_1 = \{p_1\} \\ H_2 &= U(H_1, p_2, t); L_2 = V(L_1, H_1, p_2, t) \\ H_3 &= U(H_2, p_3, t); L_3 = V(L_2, H_2, p_3, t) \\ &\vdots \end{aligned}$$

$$H_n = U(H_{n-1}, p_n, t); L_n = V(L_{n-1}, H_{n-1}, p_n, t)$$

After n iteration, if $H_n \neq \Phi$, the intersection of all nodes in H_n should be great than or equal to t , that is to say, a $C(|L_n|, |H_n|)$ is found. Then we can check if $|L_n| \geq i$ and $|H_n| \geq j$, we output $C(L_n, H_n)$. We choose $C(p_1, BGs)$ as initial value of H , it is because $|C(p_1, BGs)| = n$. So that guarantees to find max size of neighborhood intersection.

Theorem 2: The BGs is constructed bipartite graph from p , $\{BGc\}$ is the set of BGc , $n = |L(BGs)|$ and $m = |R(BGs)|$. If there exist $\{BGc\}$, by apply U and V function to all nodes in $L(BGs)$ with $t = \{m, m-1, \dots, j\}$, and output $C(L, H)$ with $|L_n| \geq i$, $\{BGc\}$ can be found by H_n and L_n .

Proof: H_n is all possible intersections with $|H_n| \geq t$. We iterate the calculation of H_n and L_n with t from $m, m-1, \dots, j$. L_n is the set of node selected from $L(BGs)$ when $|H_n| \geq t$. Therefore, if a BGc exist and $L(BGc) \subseteq L(BGs)$ and $R(BGc) \subseteq R(BGs)$, the BGc can be found by H_n and L_n .

Theorem 1 tell us that the constructed bipartite graph BGs from a node p is complete to include all possible $\{BGc\}$ and **Theorem 2** tell us that by enumerating t from $m, m-1, \dots, j$ to calculate H_n and L_n , where $n = |L(BGs)|$ and $m = |R(BGs)|$, we can find all possible cores. Following is the detail description of our algorithm with specified i and j .

4.3 The Implementation of Algorithm

We need two pair of dataset to store BGw and BGs respectively. Each BG is represented by a pair of dataset. The BGw is stored in dataset DSL and DSR in hard disk. BGs is stored in dataset EOS and EOD in main memory. All operation is based on both DSL and DSR . We chose a page from DSL , usually the first one in dataset DSL . Then we apply following algorithm until both DSL and DSR are empty. Thus,

either a BGc is found or not, there must be a collection of pages are removed from both dataset. Our algorithm consists of following steps and we repeat all steps until web graph empty.

- Step1: Get a node p from dataset DSL, if DSL is empty to the end of algorithm.
- Step2: If $|C(p, BG_w)| \geq i$ then construct BGs.
- Step3: Prune BGs with i and j .
- Step4: Extract BGc by U and V function.
- Step5: Delete edges from BGw.

5 Decision Topics on Communities

After completion of community's extraction, next job is to decide topics of each community. Due to the communities generated from only linkage information, eventually the page content is used to found topics. In the researches mentioned before, this job is fulfilled by human effort. So a mechanized process for dealing with over a hundred thousand communities is necessary. Our intuition is very simple, based on terms frequency. We can count the occurrences of terms in each page and rank the frequencies of terms. From the ranking list, we choose top N terms to make a term list. So the topic of each community will be correspondent to a term list. A stop list is needed. Because many words like 'a, the' have not the meaning, they don't help to find topic. Term in different field should be assigned different weight, for example, the terms appear in the title field will have higher weight than in other field.

6 Experiment and Result

6.1 Preparation Dataset

During crawling procedure, crawler only deals with potential fans and reserve linkage information. Crawling process continue until disk full. Because in this experiment we like to verify the feasibility and effectiveness of our algorithm, so we collect part of web graph that contain about 6.7 million pages and 9.4 million edges. Owing to one source page followed by at least 6 destination pages, the resulting nodes of graph will be great than 6.7 million. We delete mirror or near-duplicated pages. Then we create two dataset DSL and DSR and prune centers by DSR. After mirror deletion and in-degree pruning, the dataset contain 6.9 million edges.

6.2 Cores Extraction

After dataset preparation, we apply new algorithm on dataset. We run 3 times with $i,j=2$, $i,j=3$ and $i,j=4$ respectively. Then we get 149K, 29K and 10K Cores. From the cores extracted, we can find many are not included by trawling. Fig. 2(a) depicts the

distribution of cores vs. fans. Fig. 2(b) depicts cores vs. centers. From the curve of in Fig. 2, the distribution of $C_{i,j}$ s versus fans and centers obey power-law.

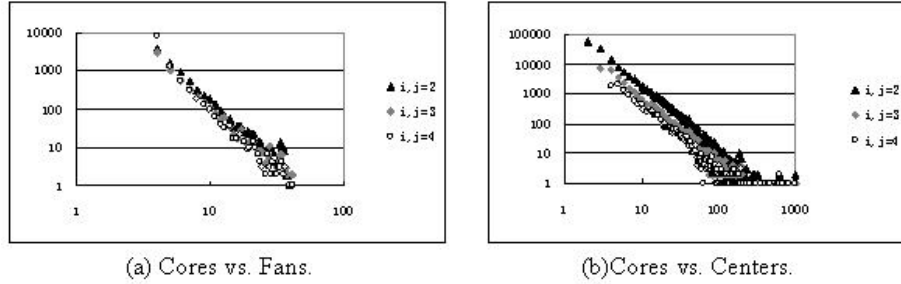


Fig. 2. Cores vs. fans and centers.

6.3 Core Examples

The cores extracted from SDL and SDR are numerous. To systematically arrange them into a reasonable structure is still an important job. This job is our next work. In this paper, we chose 3 cores by random. Each of them have topic on “Music”, “Agriculture” and “Green building”. For each topic, top 3 urls of fans and centers are shown in Table 1.

Table 1. Cores with Topic “Music”, “Agriculture” and “Green building”.

Music	Fans	http://www.ai88.net/wz/music.htm
		http://www.6cn.com/web/music_mp3.htm
		http://www.zpartner.com/data/24.htm
Centers	http://www.chinamusicnet.com/	
	http://music.silversand.net/	
	http://www.mtv114.com/	
Agriculture	Fans	http://www.3-xia.com/njz/index.asp
		http://www.animalsci.com/index.4.htm
		http://www.cqagri.gov.cn/cqagri/index.asp
Centers	http://www.jjny.gov.cn/	
	http://www.wzny.gov.cn/	
	http://www.qjqagri.gov.cn/	
Green building	Fans	http://www.asu.edu/fm/greenbuilding.htm
		http://www.greenbuilder.com/general/BuildingSources.html
		http://www.usgbc.org/Resources/links.asp
Centers	http://www.gbapgh.org/	
	http://www.ci.scottsdale.az.us/greenbuilding/	
	http://www.builtgreen.org/	

7 Conclusions and Future Works

Many communities are implicit and to discover them is a challenge job. In this paper, we discussed related researches and pointed out some defects of trawling algorithm. We proposed a new algorithm under exhaustive idea and removal of edges. A bipartite subgraph is constructed from a chosen page and our algorithm is applied to the subgraph. At each scan of dataset, we can find all possible cores and some edges are removed from web graph. In dataset collection phrase, we improve crawler by dealing with potential fans and save disk space considerably. We also use terms frequency and rank of frequencies to deduce possible topics of communities. The web pages are collected with about 7 million pages. We have set up an experiment on $i, j=2$, $i, j=3$ and $i, j=4$ and find 149K, 29K and 10K cores respectively.

The web communities are very important structures in web. Finding all possible implicit communities is still a huge project. The future work could be on these aspects. First, the page's text content should be considered with linkage information. Second, the inner structure of html document is taken into account. Third, communities have the hierarchy. Forth, how to deal with the overlap is a worthwhile research.

References

1. Kumar R., Raghavan P., et al.: Trawling the web for emerging cyber-communities. Proceedings of the 8th WWW Conference, Toronto, Canada (1999) 403-415
2. Kumar R., Raghavan P., et al.: Extracting large-scale knowledge base from the web. Proceedings of 25th VLDB Conference, Edinburgh, Scotland (1999) 639-650
3. Broder A., Kumar R., et al.: Graph structure in the web. Computer Networks 33 (1-6) (2000) 309-320
4. Gibson D., Kleinberg J., et al.: Inferring Web Communities from Link Topology. Proceedings of the 9th ACM Conference on Hypertext and Hypermedia, Pittsburgh, PA, USA (1998) 225-234
5. Chakrabarti S., Dom B. E., et al.: Automatic resource compilation by analyzing hyperlink structure and associated text. Computer Networks 30 (1-7) (1998) 65-74
6. Dean J., Henzinger M. R.: Finding Related Pages in the World Wide Web. Proceedings of the 8th WWW Conference, Toronto, Canada (1999) 389-401
7. P K Reddy and Kitsuregawa M.: Inferring Web Community through relaxed-cocition and power-law. Annual Report of KITSUREGAWA Lab (2001) 27-40
8. Flake G. W., Lawrence S., et al.: Efficient Identification of Web Communities. Proceedings of the 6th ACM SIGKDD Conference on Knowledge discovery and data mining, Boston, MA, USA (2000) 150-160
9. Girvan M., Newman M. E. J.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA 99 (2002) 7821-7826
10. Newman M. E. J.: Fast algorithm for detecting community structure in networks. Phys. Rev. E 69, 066133 (2004)
11. Newman M. E. J.: Detecting community structure in networks. Europe. Phys. J. B 38, (2004) 321-330
12. <http://www.cs.cornell.edu/home/kleinber/web-graph.ps>
13. <http://perso.wanadoo.fr/sebastien.ailleret/index-eng.html>
14. Broder A. Z., Glassman S. C., et al.: Syntactic Clustering of the Web. Computer Networks 29(8-13) (1997) 1157-1166