An Incremental Clustering Crawler for Community-Limited Search

Gye-Jeong Kim*, Kyu-Young Whang, Min-Soo Kim,
Hyo-Sang Lim, Ki-Hoon Lee
Department of Computer Science
KAIST
Yuseong-Gu, Daejeon, Korea
{gjkim, kywhang, mskim, hslim, khlee}@mozart.kaist.ac.kr

Byung Suk Lee
Department of Computer Science
University of Vermont
Burlington, Vermont, USA
bslee@cems.uvm.edu

Abstract

We propose an incremental clustering crawler*, a novel algorithm for finding communities for community-limited search in the web. A web community is a set of semantically related sites found through link-based clustering. The key idea of the proposed algorithm is to perform clustering incrementally while crawling is in progress. This algorithm does not need to crawl all the web pages a priori, but needs to crawl only as many web pages as are relevant to the clusters that are being formed. This ability to crawl on the fly is an important advantage since it is infeasible to crawl the entire set of web pages in the world and since we often do not even know which web pages or sites to crawl. Another advantage is that the time spent on clustering is reduced because at any time the clustering is performed on only the relevant web pages collected thus far. An apparent disadvantage is that the resulting clusters are not optimal since the algorithm does not have all the crawled sites available at the time of clustering. Experiments show, however, that the achieved cluster quality is comparable to the optimal cluster quality which, in our experiments, is achieved using the minimum spanning tree clustering algorithm.*

1 Introduction

Web is a vast source of information affecting everyone's life in modern society. As the data on the web are becoming increasingly large and diverse [19], the web has evolved to become a space that cannot be searched efficiently or effectively without a good web search system [23]. Traditional web search systems use either the *web search engine* approach or the *web directory* approach [3].

Recently there have been hybrid approaches proposed to combine these two approaches to have a system that can retrieve a larger and more precise search result for a user query. Examples are the *limited search* approach [14], the *focused crawler* approach [2, 5, 9, 10], and the *web clustering* approach [4, 7, 11, 22, 24, 27, 30]. These hybrid approaches improve the size and the precision of a search result, but they still have drawbacks. The limited search limits the search to only the sites or domains specified by the URL, and thus it excludes all other semantically related sites or pages. The focused crawler crawls at query time, and thus it takes long to retrieve the search result. The web clustering processes a large number of sites or pages to find clusters, and thus it is costly in computation time and space. (More details will appear in Section 2.)

In this paper we use the notion of *community-limited search* first introduced by G.-J. Kim et al. [16, 17, 18] and propose the *incremental clustering crawler (ICC)* as a novel approach to finding a community. A community on the web is defined as a set of semantically related sites found through link-based clustering. The key idea of the proposed ICC approach is to perform clustering incrementally *while crawling is in progress*. The advantage of this is that it does not need to crawl all the web pages beforehand but needs to crawl only the web pages that are relevant to the clusters as they are being formed. This is an important advantage since it is infeasible to crawl the entire set of web pages in the world and since we often do not know which web pages or sites to crawl. Another advantage is that the time spent on clustering is reduced because the clustering is performed only on the relevant web pages collected so far.

The proposed ICC algorithm works by applying Prim's minimum spanning tree (MST) algorithm on each of the provided seed sites. For each seed, the algorithm stops when it does not find any more edge to add to the partial minimum spanning tree found so far. It thus finds *locally optimal* clusters for each seed.

In this paper we empirically compare the proposed algorithm with another algorithm employing the minimum spanning tree clustering [21, 29]. This algorithm crawls all needed sites before clustering them, and always finds *globally optimal* clusters. The experimental results demonstrate the high quality of clusters generated by ICC compared with the quality of the optimal clusters generated by the minimum spanning tree clustering. This cluster quality is particularly noteworthy in consideration for the inherent

*Current affiliation: LG Electronics Institute of Technology, Korea.

978-1-4244-4457-1/09/$25.00 ©2009 IEEE

disadvantage of ICC that not all sites are available at the time of clustering. Besides, the experimental results verify the significantly shorter (by two orders of magnitude) clustering time of the ICC algorithm.

This paper makes the following contributions. First, it introduces the concept of community-limited search as a new approach to retrieving precise and large search result for a user query. Second, it proposes ICC, which crawls only the relevant sites and performs clustering incrementally, interleaved with the crawling process. Third, it demonstrates the merit of the proposed approach through extensive experiments.

The rest of this paper is organized as follows. In Section 2, we discuss related work on the web search systems and the recent hybrid approaches. In Section 3, we provide some background on web page clustering techniques. In Section 4, we introduce the concept of community-limited search. In Section 5, we present the proposed ICC algorithm and describe its implementation. In Section 6, we evaluate the performance of the ICC algorithm. In Section 7, we conclude the paper and outline the future work.

2 Related Work

As mentioned in the Introduction, a traditional web search system is either a *web search engine* or a *web directory*. A web search engine collects web pages using an automatic crawler, creates an index on the collected web pages, and searches those web pages using the index in response to a user query. Google [12] and Altavista [1] are the examples. Since the collection is done automatically, the system can collect and search a very large number of pages and, as a result, may retrieve a large number of pages for a user query. This large number, however, often makes it difficult for the user to identify the pages she wants. In contrast, a web directory organizes web sites into a hierarchical category through manual editing and searches those categorized sites in response to a user query. Yahoo! [28] and Google Directory [13] are the examples. Since the categorization is done manually, the system can retrieve highly precise results for a user query. The search, however, is limited to only those sites manually collected (less than 1% of the entire web). Besides, the search cannot be done at the page level because a site is the smallest category in the web directory.

The hybrid approaches recently proposed – *limited search* [14], *focused crawler* [2, 5, 9, 10], and *web clustering* [4, 7, 11, 22, 24, 27, 30] – have the following characteristics. The limited search limits the search scope to the web pages in a particular site or domain instead of the entire web so that the search result shows higher precision than in the traditional web search engine. Key search systems like Google, Altavista, and Yahoo! all support this approach now. Home search [18] is an advanced form of site-limited search, whereby the search scope is limited to the user's home site, and is characterized by the fact that the user issues a query directly at one's home site instead of visiting the web search engine site. The query is sent to and executed at the search engine transparently to the user.

The focused crawler collects the web pages or sites dynamically with a focus on those sites relevant to the topic of a user query so that the search result shows higher precision than in the traditional web search engine. The web clustering collects web pages using a crawler and builds a category automatically by clustering the collected sites or pages based on the terms or links in the pages so that there can be more web pages in more categories (hence more search result as well) than in the traditional web directory.

These hybrid approaches indeed improve the size and the precision of a search result, but they still have the drawbacks mentioned in the Introduction. Our proposed incremental clustering crawler (ICC) approach has the following advantages over each of the three hybrid approaches. First, with the search scope being a community, it can search semantically related sites and thus retrieve a more precise search result than the limited search. Second, without the need to collect web pages at query time, it can retrieve the search result faster than the focused crawler by using an index created on pre-collected web pages and communities. Third, since it performs clustering on only the relevant sites crawled incrementally, it spends less time on clustering than the web clustering.

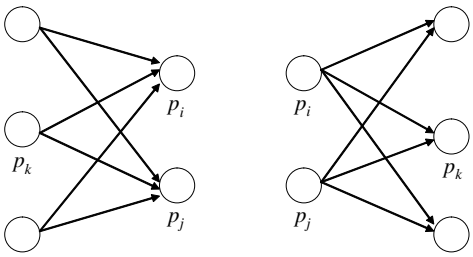
3 Web Page Clustering

Web page clustering can be classified into *term-based* clustering [4, 6, 7] and *link-based* clustering [11, 20, 22, 24, 27]. The term-based clustering performs clustering based on the common terms among web page documents. This clustering approach is not applicable to non-textual documents information, such as image, audio, video, and frame. Unfortunately, however, a majority of web pages have non-textual contents in them.

The link-based clustering is based on the hyperlinks, which are contained in every web page and characterize how the web pages are related. It measures the similarity between web page documents using such link-based metrics as *co-citation* [26] and *bibliographic coupling* [15], and finds clusters using such clustering algorithms as *K-means* [25] and *minimum spanning tree (MST) clustering* [21, 29]. Let us further discuss these similarity metrics and clustering algorithms in the remainder of this section.

The co-citation between two web pages, p_i and p_j, is defined as the number of pages that cite (i.e., have direct links to) both p_i and p_j. The bibliographic coupling between p_i and p_j is defined as the number of pages that are cited by (i.e., have direct links from) both p_i and p_j. Figure 1 illustrates these two similarity metrics.

K-means is a frequently used clustering algorithm [8], and has been applied to web page clustering [22, 24, 27]. Figure 2 outlines the algorithm. This algorithm can find clusters faster than other clustering algorithms, but it requires the number of clusters, k, to be pre-determined, and



(a) Co-citation. (b) Bibliographic coupling.

In (a), p_k co-cites p_i and p_j; in (b), p_k biblio-couples p_i and p_j.

Figure 1. Co-citation and biblio coupling.

K-means clustering algorithm
Input:
D: data set
S: similarity matrix between data items
k: required number of clusters
Output:
k clusters partitioning *D*
begin
1. Randomly choose *k* data items as the initial centroids.
2. Assign each data item to the closest centroid.
3. Recompute the centroid of each cluster.
4. Repeat Steps 2 and 3 until no centroid changes in Step 3.
end

Figure 2. The K-means clustering algorithm.

the resulting cluster quality is sensitive to the value of k and the initial partitioning. K-means clustering always achieves a globally optimal clustering for a given set of initial centroids, but the resulting clustering is not necessarily optimal for another set of initial centroids.

Minimum spanning tree (MST) clustering performs clustering by first building an MST and partitioning it by removing all edges whose weights are above the weight threshold [29]. Figure 3 outlines the algorithm. The algorithm terminates when either the required number (k) of clusters are found or the weights of all remaining edges fall below the weight threshold (wt). MST clustering thus can be used even if the number of clusters is not known. Figure 4 illustrates how the algorithm works. Given the graph $G = (V, E)$ at the top, the MST $G' = (V, E')$ shown in the middle is generated, and then the cluster shown at the bottom is generated if a termination condition k=4 is satisfied.

MST clustering always achieves a globally optimal clustering. This is because the input MST is an optimal spanning tree, that is, with the minimum total weight of edges. Since MST clusters are generated by removing edges with the largest weight first, the generated clusters are guaranteed to have the minimum total weight of edges in them. We apply this algorithm to web clustering in this paper to find globally optimal web clusters (i.e., communities). In the context of web clustering, optimal clustering means that the clusters formed have the *highest* total similarity. Thus,

MST clustering algorithm
Input:
G=(*V*, *E*): graph
either *k* or *wt*, where
k: minimum required number of clusters
wt: weight threshold (the largest edge weight allowed in a cluster)
Output:
clusters of the vertex set *V*
begin
1. Find a minimum spanning tree *G'*=(*V*, *E'*) from *G*=(*V*, *E*).
2. Remove from the edges in *E'* the edge (v_i, v_j) with the largest weight.
$E' = E' - \{(v_i, v_j)\}$
3. Repeat Step 2 while either the number of subtrees (i.e., clusters) generated is smaller than k (if *k* is given) or there exists an edge in *E'* whose weight is larger than *wt* (if *wt* is given).
4. Make one cluster for the vertex set of each subtree generated from Steps 2 and 3.
end

Figure 3. The MST clustering algorithm.

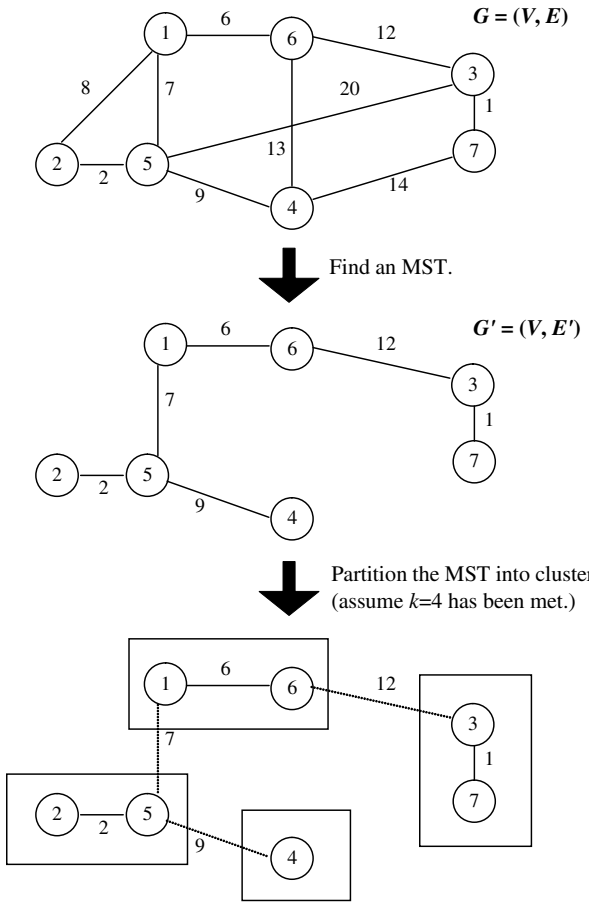


Figure 4. An MST clustering example.

we only need to represent the edge weight in the MST clustering as the inverse of the similarity between web sites.

4 Community-Limited Search

The community-limited search is an advanced limited search approach. It limits the web search to a particular community, where a community is defined as a set of semantically related sites obtained through link-based clustering. Thus, the community-limited search achieves higher precision in the search result than the site- or domain-limited search.

An example of the community-limited search is as follows. Suppose the user who wants to find information about job openings at companies in Samsung Group gives the query terms "Samsung" and "job opening". If the search is done against pages in the entire web, then so many irrelevant web pages (at news sites, job information sites, etc.) are retrieved that it is hard for the user to find the right pages. If the search is done against sites or categories (instead of pages), then the sites of Samsung Group companies can be readily found, but the user should still explore the sites one by one in order to find the right pages containing the job information. In contrast, if the search is done against communities, the term "Samsung" can be used to find the community of Samsung Group companies and then the job information pages can be found easily by limiting the search to that community.

The community-limited search involves the steps of collecting web pages and finding a community from them. The existing approach to finding a community applies the focused crawler at the level of a web site[9, 10]. In this approach, the sites near each seed are crawled and the maximum flow/minimum cut is found from the graph that is made of the collected sites and links. The found cut separates the graph into two parts, and the part on the side of the seed becomes the community of sites relevant to the seed. This approach finds a community after the crawling is completed and is applicable only if a query topic is provided. In contrast, in the ICC approach we propose, the web crawling and the web clustering are performed hand in hand at the same time and it works without a query topic. We discuss this approach in detail in the next section.

5 An Incremental Clustering Crawler

5.1 Algorithm

As mentioned above, the incremental clustering crawler (ICC) performs clustering while web pages are being collected by a crawler. We design our algorithm as an integration of the Prim's MST algorithm and the MST clustering algorithm (Figure 3). Specifically, it is based on the Prim's algorithm and uses the edge weight-based termination condition as in the MST clustering algorithm. (Note that a smaller edge weight in MST clustering is equivalent to a larger edge weight in ICC.) The reason we adopt the MST clustering algorithm is that it can work without knowing the number of clusters to be generated and few outlier sites are generated because those sites (which are not similar to other sites) are excluded when an MST is built.

The K-means algorithm (Figure 2), on the other hand, is not a good fit for ICC because it needs to know how many clusters will be generated, which is nearly impossible in the web environment, and it leaves many outlier sites that do not belong to any cluster. Besides, unlike MST clustering, K-means clustering is not always optimal but is optimal only for a given set of seeds, as indicated in Section 3.

Incremental clustering crawler algorithm
Input:
$G=(V, E)$: web graph where V is the set of sites and
E is the set of direct links between sites in the web
S: set of seed sites
wt: weight threshold (minimum required edge weight in a cluster)
Output:
clusters of sites crawled starting with the seed sites in S
begin
For each $s \in S$ **begin**
1. Start crawling from the seed site s.
$V_{cr} = \{s\}$
$V_{fr} = \{d \mid d \neq s \text{ and } (s, d) \in E\}$
2. Select the edge (v, w), where $v \in V_{cr}$ and $w \in V_{fr}$, with the largest weight, and crawl to w.
$V_{cr} = V_{cr} \cup \{w\}$
3. After crawling to w, obtain the set V_{fr}^{w} of sites that are directly linked from w and not in V_{cr}.
$V_{fr} = V_{fr} - \{w\} \cup V_{fr}^{w}$
4. Repeat Steps 2 and 3 while there exists an edge with a larger weight than wt among the edges (v, w) where $v \in V_{cr}$ and $w \in V_{fr}$.
5. Output the set of crawled sites V_{cr} as the cluster for s.
end
end

Figure 5. The ICC algorithm.

Figure 5 outlines the ICC algorithm. Starting from each seed site s, clusters are formed by including edges incrementally using the Prim's MST algorithm. Note that in Prim's algorithm the vertices added at each step are always connected to the seed, thus naturally forming a cluster. V_{cr} denotes the set of vertices included in the cluster (i.e., cut) so far, and represents the set of sites already crawled. V_{fr} is the set of fringe vertices that are not in the cluster but directly connected from the vertices in it, and represents the set of new sites that can be crawled next. The weight of an edge represents the similarity between two sites. (The similarity metric used in this algorithm is described below.) The weight threshold (wt) is the minimum weight required for every edge connecting V_{cr} and V_{fr}, and represents the minimum similarity required between two sites in the same cluster. It is used to determine when to stop "growing" a cluster. This is opposite to the way the weight threshold is used by MST clustering (Figure 3) which forms the clusters through edge removal.

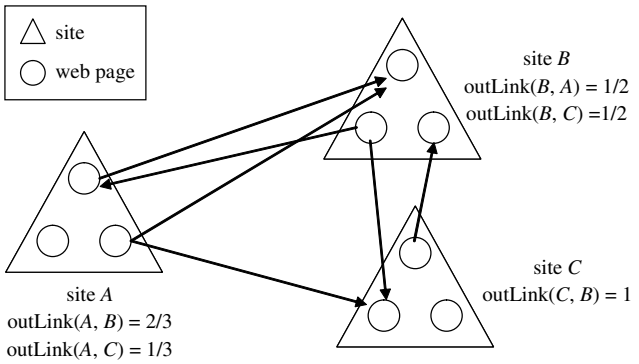
Note that the Prim's algorithm used here stops when there is no more edge that can be added among those con-

necting the currently included vertices (V_{cr}) to the fringe vertices (V_{fr}), unlike the original Prim's algorithm which does not stop until all vertices in the graph are discovered. For this reason, the proposed ICC algorithm achieves locally optimal clustering for each seed, while globally optimal clustering (as possible with the MST clustering) is not necessarily achieved after completing the algorithm for all seeds. Naturally, the choice of seeds influence the quality of generated clusters; it may well be a random selection when there is no prior knowledge of the web sites crawled.

The two similarity metrics – co-citation and bibliographic coupling – described in Section 3 are not readily applicable to the proposed incremental clustering approach because they need all sites to be available at once. Besides, the computational overheads of using those metrics can be prohibitive in the large-scale web environment. Thus, we have devised a new metric called the *outLink*. With outLink, two sites s_i and s_j are considered more relevant if s_i has more direct links to s_j (or vice versa). Specifically, let n_{ij} be the number of direct links from s_i to s_j and let n_i be the total number of direct links from s_i to all other sites. Then, outLink from s_i to s_j is computed as in Equation 1.

$$\text{outLink}(s_i, s_j) = \frac{n_{ij}}{\sum_{k \in L(k)} n_{ik}} \quad (1)$$

where $L(k)$ is the set of sites (excluding s_i) directly linked from s_i. Figure 6 illustrates this computation. Since outLink can be computed considering only the outgoing sites directly linked from a node already included in a cluster, this metric can be applied incrementally with no problem and also the computation overhead is minimal.



At the site A for example, the total number of out-links is three and, among these three, two links are to the site B and one link is to the site C. Hence, the outLink(A, B) is 2/3 and outLink(A, C) is 1/3.

Figure 6. An example of computing outLink.

5.2 Implementation

Figure 7 shows the architecture of ICC. It downloads web pages in parallel using the asynchronous I/O. It can process 50 or more pages at once. The main module assigns a seed site URL to each cluster finder and starts them to find clusters while crawling. The cluster finder manages

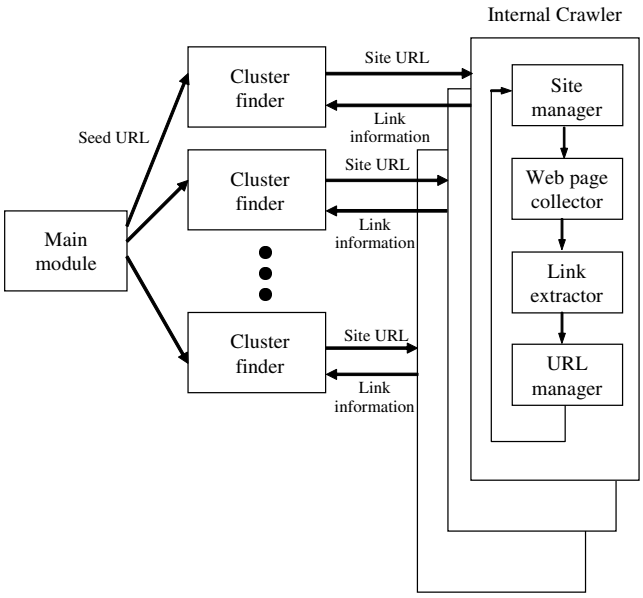


Figure 7. Architecture of ICC.

the currently included vertices V_{cr} and the fringe vertices V_{fr} shown in Figure 5 and, based on the weights of edges from V_{cr} to V_{fr}, chooses the sites to be included in a cluster.

Each time the cluster finder chooses a site, the internal crawler crawls to the web pages in that site. The internal crawler has four parts: site manager, web page collector, link extractor, and URL manager. The site manager has information about each site and checks the termination condition of crawling (e.g., the number of web pages to crawl) at the site currently visited. The web page collector downloads from the web server the web pages stored at the given URLs. The link extractor then extracts links from the downloaded web pages. Using the extracted links, the URL manager updates information about the web pages already collected and the web pages yet to be collected.

6 Performance Evaluation

For performance evaluations, we compare the cluster quality and clustering time between the incremental clustering crawler (ICC) and the minimum spanning tree (MST) clustering.[1] (In this section, we refer to the clusters generated by these algorithms as *ICC clusters* and *MST clusters*, respectively.)

To evaluate the cluster quality, we conduct two sets of experiments. The first set of experiments is to evaluate the optimality of the ICC clusters by comparing them with those generated by the optimal MST clustering. The second set of experiments is for user testing of the cluster quality, that is, to evaluate the quality of ICC clusters by comparing them with manually selected true clusters. In the clustering time

[1] We have also done experiments comparing the K-means clustering and the MST clustering and verified that the latter outperforms the former in both the cluster quality and the clustering time. The results of these experiments appear in a technical report [17].

experiment, ICC would certainly take less time due to its incremental processing, and thus the objective is to see how short it is compared with MST.

Data used in the experiments are the set of sites collected by ICC from the web with the weight threshold set to 0.01. The number of sites varies with the number of seeds, and the maximum number of sites collected in the experiments is about 11,000. The selection of seeds differ between the two sets of experiments. In the cluster optimality experiments, randomly selected seeds (varying from 1 seed to 1000 seeds) are used without the knowledge of the web sites crawled, whereas in the user testing experiment, manually selected seeds (115 seeds total) are used for the sake of objectivity. The random seeds are used in the clustering time experiments as well.

The outLink (see Equation 1) is used as the similarity metric between sites in all experiments. (Precisely, the similarity metric in the case of MST clustering is the *inverse* of OutLink, as mentioned in Section 3.) All experiments are done in Linux 2.4 OS on a PC with 1.7GHz Pentium 4 CPU and 512MB RAM. All programming is done in Python.

6.1 Cluster quality

There is no objective metric of cluster quality, but *precision*, *recall*, and *f-value* (or *f-measure*) are commonly used for that. Precision and recall are used as the similarity metrics between two clusters. Since these two metrics show opposite results, the f-value is used as the metric combining the two. (This is called the *weighted harmonic mean* of precision and recall.) Specifically, the three metrics are defined as follows.

$$\text{precision} = \frac{|C_T \cap C_A|}{|C_A|} \tag{2}$$

$$\text{recall} = \frac{|C_T \cap C_A|}{|C_T|} \tag{3}$$

$$\text{f-value} = \frac{2(\text{precision} \cdot \text{recall})}{(\text{precision} + \text{recall})} \tag{4}$$

where C_T is a true cluster and C_A is a test cluster obtained using the evaluated algorithm. $|C|$ denotes the size of a cluster C.

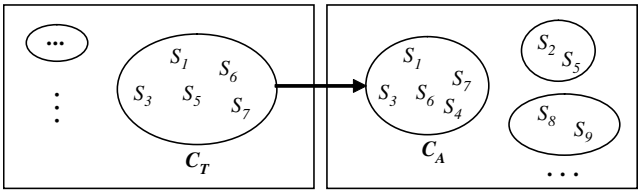
A clustering algorithm generates a *set* of clusters. Thus, given a true cluster, one of these clusters should be chosen as the cluster to be compared against it. In this experiment, we choose the cluster with the largest overlap with the true cluster, that is, choose $C_A \in \{C_1, C_2, ..., C_n\}$ such that

$$C_A = \underset{C_{A_i}}{\text{argmax}} |C_T \cap C_{A_i}| \tag{5}$$

where $C_1, C_2, ..., C_n$ are the generated clusters and C_T is the given true cluster. Figure 8 illustrates this.

Optimality of cluster quality

As mentioned above, we compare the clusters generated by ICC with those generated by MST clustering. For the purpose of this comparison, we make sure the two algorithms generate the same number of clusters by using the number of clusters (k) as the termination condition of the MST clustering algorithm and setting k to be equal to the number of clusters generated by ICC. Then, we pick one ICC cluster for each MST clustering cluster in the manner shown in Equation 5 and then compute the average of each of the three metrics over all ICC clusters. That is, if n ICC clusters $C_{A_1}, C_{A_2}, ..., C_{A_n}$ match n MST clustering clusters $C_{T_1}, C_{T_2}, ..., C_{T_n}$ one to one, then:



True clusters. Test clusters.

Among all clusters on the right, C_A overlaps most with C_T on the left, and thus is chosen for comparison with C_T.

Figure 8. An example of choosing a test cluster.

$$\text{precision}_{\text{avg}} = \frac{\sum_{i=1}^{n} \text{precision}_i}{n} \tag{6}$$

$$\text{recall}_{\text{avg}} = \frac{\sum_{i=1}^{n} \text{recall}_i}{n} \tag{7}$$

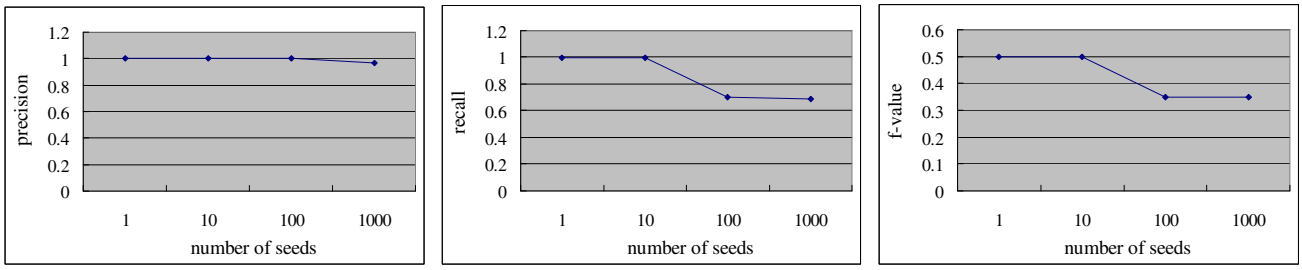
$$\text{f-value}_{\text{avg}} = \frac{\sum_{i=1}^{n} \text{f-value}_i}{n} \tag{8}$$

where precision_i, recall_i, and f-value_i are those in Equations 2, 3, and 4, respectively, for C_{T_i} and C_{A_i}.

Figure 9 shows the result (average precision, average recall, and average f-value) for varying number of seeds. The average precision is close to the maximum value 1.0 in the entire range of the number of seeds. The average recall, however, drops to about 0.7 when the number of seeds is increased to 100. We think this happens because the ICC algorithm used in the experiment disregards links between clusters stemming from different seeds and, thus, does not merge clusters, whereas such a merge may happen in the MST clustering algorithm. We thus expect that the average recall should improve if we would merge two clusters that have enough links between them. The average f-value shows the same trend as the average recall. The reason for the drop is also the same.

User testing of cluster quality

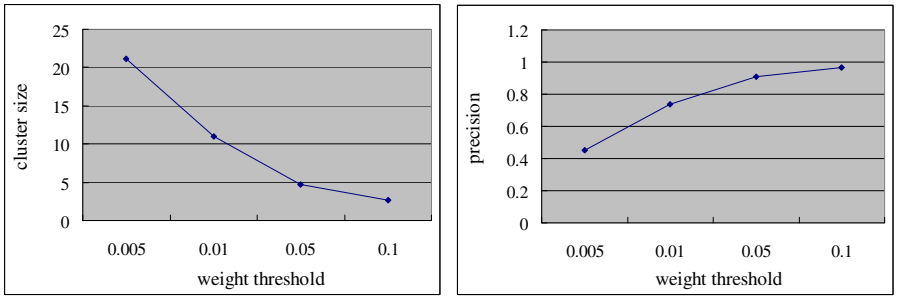
In this experiment, we measure the average size and average precision of ICC clusters. The average size is the average number of sites in each cluster, and the average precision is as defined in Equation 6. We use the home pages of 115 universities in the country as the seeds for this experiment and identify the true clusters by examining the generated ICC clusters manually. The university home pages have similar characteristics, and thus the sizes of the resulting



(a) Precision. (b) Recall. (c) f-value.

The results are shown for varying number of seeds. Precision, recall, and f-value are, respectively, an average over all ICC clusters generated. Clusters generated using the MST clustering are used as the true clusters.

Figure 9. Quality of ICC clusters relative to the optimal MST clusters.



(a) Cluster size. (b) Precision.

The results are shown for varying weight threshold. Cluster size and precision are, respectively, an average over all clusters generated. Clusters manually selected from the sites of the same university are used as the true clusters.

Figure 10. Quality of ICC clusters relative to the manually identified true clusters.

clusters have limited variations. Each of the true clusters manually selected contains sites in the same university, and this allows for an objective comparison of them against the clusters generated by ICC.

Figure 10 shows the result. As expected from the algorithm, the cluster size decreases and the precision increases as the weight threshold increases. This user testing has led to a conclusion that the cluster quality is acceptable when the cluster size is larger than 10 and the cluster precision is higher than 0.5. From the figure we see that the former condition holds for the weight threshold no higher than 0.01 (Figure 10(a)) and the latter condition holds for the weight threshold no lower than 0.005 (Figure 10(b)). We thus judge in the case of this experiment that the adequate weight threshold is in the range of 0.005 to 0.01.

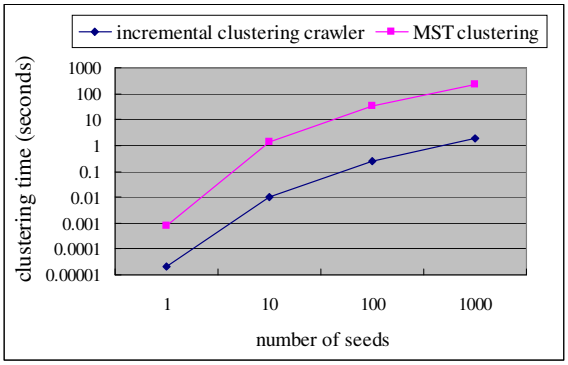


Figure 11. Clustering time.

6.2 Clustering time

We use the elapsed time as the metric of clustering time. For ICC, clustering is interleaved with crawling. Thus, we measure the net time spent on clustering, that is, the total time spent on the cluster finder and not on the internal crawler (see Figure 7). We use the number of seeds as the experimental parameter. The weight threshold in MST clustering is set to 0.01, the same value used for ICC.

Figure 11 shows the experimental result, where both the horizontal and vertical axes are in log scale. We see that the clustering time of ICC is two orders of magnitude smaller than MST clustering. Beside the reason that the clustering in ICC is performed incrementally on only the relevant pages collected so far, our analysis finds that, since clustering in ICC is interleaved with crawling, the link information needed for clustering is already in memory and the computation of the similarity between sites is already partly done during crawling.

7 Conclusion

In this paper we have proposed the incremental clustering crawler (ICC) as a novel clustering approach to finding communities for use in the community-limited search. In this approach, clustering is done incrementally while web pages are collected through crawling. Despite the fact that not all sites are available at the time of clustering, it generates high-quality clusters (or, communities) that are comparable to globally optimal ones. Additionally, it reduces the time spent on clustering. We have confirmed these through experiments.

The immediate future work is to extend the ICC algorithm to merge clusters generated for different seeds as they approach each other with enough direct links between them. Another future work is to conduct additional performance evaluations like measuring the ratio of the number of crawled sites out of all sites for varying number or type of the seeds. We also plan to study how to choose the right seed sites to start the crawling from and how to choose relevant web pages within each site during crawling, to increase the cluster quality.

Acknowledgment

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the National Research Lab Program funded by the Korean Government (MEST) (No. R0A-2007-000-20101-0). The work by the University of Vermont author was supported by the US National Science Foundation (Grant No. IIS-0415023).

References

[1] AltaVista, http://www.altavista.com

[2] Badia, A., Muezzinoglu, T. and Nasraoui, O., "Focused Crawling: Experiences in a Real World Project," *Proc. Conf. WW* , pp. 216-217, May 2006.

[3] Baeza-Yates, R. and Ribeiro-Neto, B., "Searching the Web," in *Modern Information Retrieval*, pp. 367-396, 1999.

[4] Broder, A. et al., "Syntactic Clustering of the Web," *Proc. Conf. WWW*, pp. 391-404, Apr. 1997.

[5] Chakrabarti, S., vander Berg, M, and Dom, B., "Focused Crawling: A New Approach to Topic-specific Web Resource Discovery," *Proc. Conf. WWW*, pp. 545-562, May 1999.

[6] Chim, H. and Deng, X., "A New Suffix Tree Similarity Measure for Document Clustering," *Proc. Conf. WWW*, pp. 121-129, May 2007.

[7] Cutting, D. et al., "Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections," *Proc. Conf. ACM SIGIR*, pp. 318-329, June 1992.

[8] Dubes, R., and Jain, A., *Algorithms for Clustering Data*, Prentice Hall, 1988.

[9] Ester, M., Kriegel, H., and Schubert, M., "Accurate and Efficient Crawling for Relevant Websites," *Proc. Conf. VLDB*, pp. 376-407, Aug. 2004.

[10] Flake, G., Lawrence, S. and Giles, C., "Efficient Identification of Web Communities," *Proc. Conf. ACM SIGKDD*, pp. 150-160, Aug. 2000.

[11] Gibson, D., Kleinberg, J., and Raghava, P., "Inferring Web Communities from Link Topology," *Proc. Conf. ACM Hypertext*, pp. 225-234, June 1998.

[12] Google, http://www.google.com

[13] Google Directory, http://dir.google.com

[14] Google Help, http://www.google.com/help/refinesearch.html

[15] Kessler, M., "Bibliographic Coupling between Scientific Papers," *American Documentation*, 14(1):10-25, 1963.

[16] Kim, G., Kim, M., Kim, Y., and Whang, K., "Web Crawling and PageRank Calculation for Community-Limited Search" *Proc. Korea Computer Congress of KIISE* (in Korean), pp. 1-3, July 2005.

[17] Kim, G., Whang, K., Kim, M., Lee, B., Lim, H., "Incremental Clustering Crawler for Community-Limited Search," Tech. Report CS-TR-2008-283, Dept. Comp. Sci., KAIST, Yuseong-Gu, Dejeon 305-701, Korea, Jan. 2008.

[18] Kim, M., Lee, J., Kim, M., and Whang, K., "Site-, Domain-, Community-Limited Search, and Home Search in a Web Search System Using the ODYSSEUS Object-Relational DBMS," *Proc. 2005 Fall Conf. KIISE* (in Korean), pp. 175-177, Nov. 2005.

[19] Lymna, P. et al., How Much Information? 2003, Project Report, School of Inf. Management and Syst., U.C. Berkeley, 2003 (http://www.sims.berkeley.edu/research/projects/how-much-info-2003).

[20] Masada, T., Takasu, A., Adachi, J., "Link-Based Clustering for Finding Subrelevant Web Pages," *Proc. Web Document Analysis*, Sept. 2005.

[21] Mecca, G., Raunich, S., and Pappalardo, A., "A New Algorithm for Clustering Search Results," *Data & Knowledge Engineering*, 62(3):504-522, 2007.

[22] Modha, D., and Spangler, W., "Clustering Hypertext with Applications to Web Searching," *Proc. Conf. ACM Hypertext*, pp. 143-152, May 2000.

[23] Ntoulas, A., Cho, J., and Olston, C., "What's New on the Web? The Evolution of the Web from a Search Engine Perspective," *Proc. Conf. WWW*, pp. 1-12, May 2004.

[24] Ohura, Y. et al., "Experiments on Query Expansion for Internet Yellow Page Services Using Web Log Mining," *Proc. Conf. VLDB*, pp. 1008-1018, Aug. 2002.

[25] Selim, S. and Ismail, M., "K-Means-type Algorithms: a Generalized Convergence Theorem and Characterization of Local Optimality," *IEEE Trans. PAMI*, 6(1):88-87, 1984.

[26] Small, H., "Co-citation in the Scientific Literature: A New Measure of the Relationship between Two Documents," *Journal of the American Soc. of Inf. Sci.*, 24(4):265-269, 1973.

[27] Wang, Y. and Kitsuregawa, M., "Use Link-Based Clustering to Improve Web Search Results," *Proc. Conf. WISE*, pp. 114-124, Kyoto, Dec. 2001.

[28] Yahoo!, http://www.yahoo.com

[29] Zahn, C., "Graph Theoretical Methods for Detecting and Describing Gestalt Clusters," *IEEE Trans. Computers*, C-20(1):68-86, Jan. 1971.

[30] Zamir, O. and Etzioni, O., "Web Document Clustering: a Feasibility Demonstration," *Proc. Conf. ACM SIGIR*, pp. 46-54, June 1998.