Mining relaxed graph properties in Internet

Wilhelmiina Hämäläinen
Department of Computer Science,
University of Joensuu
P.O. Box 111, FIN-80101 Joensuu
Finland
whamalai@cs.joensuu.fi

Vladimir Poroshin
Department of Computer Science,
University of Joensuu
P.O. Box 111, FIN-80101 Joensuu
Finland
vporo@cs.joensuu.fi

Hannu Toivonen
Department of Computer Science
University of Helsinki
P.O. Box 26, FIN-00014 University of Helsinki
Finland
htoivone@cs.helsinki.fi

Abstract

Many real world datasets are represented in the form of graphs. The classical graph properties found in the data, like cliques or independent sets, can reveal new interesting information in the data. However, such properties can be either too rare or too trivial in the given context. By relaxing the criteria of the classical properties, we can find more and totally new patterns in the data. In this paper, we define relaxed graph properties and study their use in analyzing and processing graph-based data. Especially, we consider the problem of finding self-referring groups in WWW, and give a general algorithm for mining all such patterns from a collection of WWW pages. We suggest that such self-referring groups can reveal web communities or other clusterings in WWW and also help in compression of graph-formed data.

1 Introduction

In this paper, we introduce new concepts and methods for finding new information in graph-structured data, like World Wide Web (*WWW*), citation indexes or concept maps. Our aim is to study systematically, whether the relaxed versions of the classical graph properties (Table 2) like "nearly cliques" or "nearly independent sets" could reveal new interesting information. As an example, we will loosen the definition of a clique and consider the resulting "self-referring groups" in the WWW context. The self-referring groups can be interpreted as clusters in WWW. Such clusters give compact description of the WWW graph structure and and are especially useful for finding web communities. In addition, we will demonstrate how this new concept can be used for compressing other graph-based data, like co-operational concept maps.

In the next sections, we will first introduce the idea of self-referring groups and give the formal definitions. Then we will give the algorithm for finding all self-referring sets in a given graph and analyze its complexity. We will introduce our first experiments for finding groupings (potential web-communities) among WWW pages and applying our algorithm for compressing co-operational concept maps. Finally, we will compare our new method to previous work and draw the final conclusions.

The basic concepts and notations used in this paper are listed in Table 1.

Table 1: Basic concepts and notations for graphs.

$G = (V, E)$	A graph which consists of a set of vertices V and a set of edges $E = \{\{u, v\} \| u, v \in V\} \subseteq V \times V$. For a directed graph, E consists of ordered pairs, $E = \{(u, v) \| u, v \in V\}$.
$u \rightsquigarrow v$	A *path* (of length k) from vertex u to vertex v in a directed graph G, i.e. a sequence $(v_0, ..., v_k)$ such that $u = v_0$ and $v = v_k$ and $(v_{i-1}, v_i) \in E$ for $\forall i = 1, 2, ..., k$. In an undirected graph, ordered pairs (v_{i-1}, v_i) are replaced by unordered pairs $\{v_{i-1}, v_i\}$.
$degree(v)$	The *degree* of a vertex v. In an undirected graph, the degree of v is the number of edges containing v: $\|\{(v, w) \mid w \in V\}\|$. In a directed graph, the degree of a vertex v is $degree(v) = outdegree(v) + indegree(v)$.
$outdegree(v)$	The *outdegree* of a vertex v i.e. the number of edges leaving it.
$indegree(v)$	The *indegree* of a vertex v i.e. the number of edges entering it.
$\|G\|=\|V\|$	The *size* of a graph $G = (V, E)$ is the number of vertices in V.

Table 2: Classical graph properties. First four properties are defined for undirected graphs and the last property for both directed and undirected graphs.

Complete graph	G is *complete*, if all of its vertices are connected to each other: $\forall u, v \in V$ $\{u, v\} \in E$.
Clique	A subset of vertices $V' \subseteq V$ such that each pair in V' is connected by an edge in E: $\forall u, v \in V'$ $\{u, v\} \in E$. I.e. a clique defines a complete subgraph in G.
Independent set	A subset of vertices $V' \subseteq V$ such that no pair in V' is connected by an edge in E: $\forall u, v \in V'$ $\{u, v\} \notin E$. Observe that V' is an independent set in G iff V' is a clique in the complement graph $\overline{G} = (V, (V \times V) \setminus E)$.
Vertex cover	A subset of vertices $V' \subseteq V$ such that all edges in E are "covered" by vertices in V'. I.e. for each $\{u, v\} \in E$ either $v \in V'$ or $u \in V'$ or both. Observe that V' is a vertex cover of G iff $V \setminus V'$ is an independent set in G.
Strongly connected component	A maximal subset of vertices $V' \subseteq V$ such that for each pair of vertices $u, v \in V'$ there is a path $u \rightsquigarrow v$, $v \rightsquigarrow u$, or both. I.e. vertices u and v are reachable from each other. A graph is *strongly connected* if it has only one strongly connected component.

2 Self-referring groups

The idea of self-referring groups is to find a group of pages, which mostly refer to each other. Let us denote this set of self-referring pages as S. If all pages in S refer to all other pages in S, then S is a clique and the corresponding undirected graph is complete. However, we can loosen this criterion and require that each page in S refers to at least min_f part of other pages in S, in which min_f is a user-defined threshold for S's reference frequency. This kind of self-reference is called here as *strong self-reference.* (See example in Figure 1.) Formally, we define:

Definition: Let S be a set of vertices, $|S| > 1$, and min_f a user-defined frequency threshold. S is *strongly self-referring*, if $\forall p \in S$, $\frac{degree(p)}{|S|-1} \geq min_f$. The *reference frequency* of S is $f_{ref}(S) = \frac{\min\{degree(p) | p \in S\}}{|S|-1}$.

Observe that we are now considering the underlying undirected graph structure of the WWW, and according to definition of graph the duplicate edges are removed. By considering the directed graph structure we can find more information about self-referring groups and the relations between them. For example, we can adopt the idea of HITS-algorithm [6], and search the best *authorities* (pages referred by good "hubs") and the best *hubs* (pages referring to good authorities) in the group. It is quite plausible that the authorities reveal the good

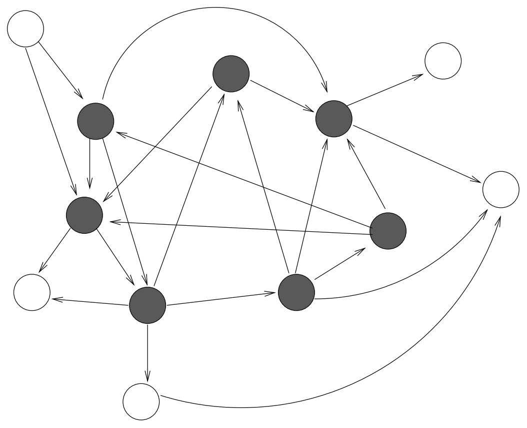
Figure 1: Graph defined by link structure of web pages. The dark nodes construct a strongly self-referring set with threshold $min_f = 2/3$.

sources of new information in the group, and the good hubs give good overviews of the topics considered in the group. Further, if we identify the WWW addresses, we can also determine the geographic distribution of a group (whether it is international or represent only some nationalities).

Similarly, we can define the relaxed counterpart of independent sets:

Definition: Let S be a set of vertices, $|S| > 1$, and min_f a user-defined frequency threshold. S is *weakly independent*, if $\forall p \in S$, $\frac{|S|-1-degree(p)}{|S|-1} \geq min_f$. The *independence frequency* of S is $f_{is}(S) = \frac{|S|-1-\max\{degree(p)|p\in S\}}{|S|-1} = 1 - \frac{\max\{degree(p)|p\in S\}}{|S|-1}$.

Observe that the weakly independent sets in G are exactly the same than strongly self-referring sets in the complement graph $\overline{G} = (V, (V \times V) \setminus E)$. Thus, we can use the same algorithm for search of weakly independent sets.

3 Algorithm for finding self-referring groups

3.1 Basic idea

In the search algorithm for self-referring pages we can restrict the search space by recognizing that the connected components are superclasses of cliques. When applying the algorithm for directed graphs, we can prune even more by restricting the search into strongly connected components.

The basic idea of the algorithm (Alg. 1) is following:

1. Search connected components from graph G by depth-first search. This can be completed in time $\Theta(3|V| + 2|E|)$ (See section Analysis.) Let the resulting vertex set be $V' \subseteq V$, and the corresponding undirected subgraph $G' = (V', E')$.
2. For each connected component search self-referring groups from G' by depth-first search (Alg. 2).

3.2 Depth-first search for self-referring sets

Searching the strongly self-referring or weakly independent sets is more complex than searching common cliques or independent sets. For common cliques and independent sets we can use simple depth-first search, because for each vertex set S, such that S is clique/independent set, also $T \subseteq S$ is clique/independent set. This means that we can prune a branch in search space, when we find a set which is not a clique/independent set. But this does not hold for self-referring sets or weakly independent sets. Thus, we will prove another pruning criterion for self-referring sets.

Lemma 1: Let $f(n)$ be the reference frequency of vertex set T, $|T| = n$, and $f(n+k)$ the reference frequency of T's superset S, $T \subseteq S$, $|T| = n + k$. Then $f(n+k) \leq \frac{n-1}{n+k-1} f(n) + \frac{k}{n+k-1}$.

Proof: Let $f(n) = \frac{k}{n-1}$. Then the maximum value of $f(n+1)$ is $\frac{k+1}{n} = \frac{n-1}{n} f(n) + \frac{1}{n}$. Let us suppose that on each step $f(n+1)$ gets the maximum value, $\frac{n-1}{n} f(n) + \frac{1}{n}$. Then, by solving the recursion equation, we get that after k steps f gets maximum value $f(n+k) = \frac{n-1}{n+k-1} f(n) + \frac{k}{n+k-1}$. $\square$

The following lemma is a direct consequence of the definition of self-referring sets:

Lemma 2: Let v be a vertex and S a strongly self-referring set, $v \in S$. Then the maximum size of S is $|S| \leq \frac{degree(v)}{min_f} + 1$. $\square$

Theorem: Let v be a vertex and S a vertex set, $v \in S$, $|S| = n$, and $f_{ref}(S)$ be the reference frequency of set S. If $f_{ref}(S) < 1 - \frac{degree(v)(1-min_f)}{(n-1)min_f}$, then none of S's superset can be strongly self-referring.

Proof: Let us denote by $k = \frac{degree(v)}{min_f} + 1 - n$ the number of items we can add to S until maximum size is met. In the best case, the frequency of maximal set is $f(n+k) = \frac{n-1}{n+k-1} f(n) + \frac{k}{n+k-1}$. The set is self-referring, if $\frac{n-1}{n+k-1} f(n) + \frac{k}{n+k-1} \geq min_f \Leftrightarrow f(n) \geq \frac{degree(v)min_f - degree(v) - (1-n)min_f}{(n-1)min_f} = 1 - \frac{degree(v)(1-min_f)}{(n-1)min_f}$.
Thus, if $f_{ref}(S) < 1 - \frac{degree(v)(1-min_f)}{(n-1)min_f}$, then S can never become strongly self-referring. $\square$

Now we can give the depth-first search Algorithm 2 for searching all strongly self-referring supersets of the current vertex set X, which is given as a parameter. In the main program, Alg. 1, the depth-first search is called by a singular vertex set $\{v\}$, its degree $d = degree(v)$, threshold min_f, and an additional parameter *last*, which is explained below. In each recursive call of Alg. 2, we first check, if the vertex set is self-referring, and in the positive case we output it. Otherwise we check, whether it can be expanded to a self-referring set. If the condition does not hold, we return from recursion. Otherwise we expand X with a new item, and call the algorithm recursively. The recursion finishes, when the maximum size of the self-referring sets containing initial vertex v is met, or we recognize that no more self-referring sets can be found. After returning from the recursion, we try another new item, until all of them are tried.

Observe that we have to fix some order for vertices to avoid testing the same vertex sets again. In the Algorithm 2, we denote $u > v$, when v precedes u in the given order. Variable *last* tells the last vertex added to the candidate group, and thus all vertices preceding it are already tested.

Alg. 1 SelfReferringSets(G, min_f). An algorithm for searching all strongly self-referring sets in graph $G = (V, E)$.

Input: $G = (V, E), min_f$
Output: $Y \subseteq V$

```
1   begin
2       compute all connected components in $G = (V,E)$
3       for each connected component $V'$ in $G = (V,E)$ do
4           for all $v \in V'$ dfs($\{v\}, degree(v), min_f, v$)
5   end
```

3.3 Analysis

Let $G = (V, E)$ be a directed graph. The strongly connected components can be found in time $\Theta(3|V| + 2|E|)$ by a well-known algorithm, which calls depth-first search once for the original graph G, once for its transpose

Alg. 2 dfs$(X, max(d, degree(u)), min_f, last)$. A depth-first search of the self-referent sets in subgraph $G' = (V', E')$.

Input: $X \subseteq V, d, min_f, last$
Output: $Y \subseteq V'$

```
1   begin
2       if $f_{ref}(X) \geq min_f$ then
3           output $X$
4       else if $(f_{ref}(X) < 1 - \frac{d(1-min_f)}{(|X|-1)min_f})$
5           then return
6       for all vertices $u \in V'$ $(u > last$ and $\exists v \in X\ (v, u) \in E)$ do
7           dfs($X \cup \{u\}, d, min_f, u$)
8   end
```

graph G^T, and outputs the vertices of each tree. [12]

Let us now consider the worst case complexity of finding all self-referring sets. If the size of the vertex set is $|V| = n$, we have to try in the worst case 2^n sets ($|\mathcal{P}(V)| = 2^n$). However, we know that the maximum size for each self-referring set X, which is superset of a vertex v, is $\frac{degree(v)}{min_{ref}} + 1$. This gives an upper limit for the depth of algorithm. In sparse graphs we achieve an enormous saving: e.g. if the graph size is 100, average degree of vertices is 18, and $min_f = 0.9$, the hight of the search tree is only $\frac{18}{0.9} + 1 = 21$ instead of 100. However, in the worst case when the graph is complete, we save nothing.

4 Initial experiments

4.1 Searching self-referring groups in Internet

In the first experiment, we searched self-referring groups in Internet. The initial graphs were constructed from the best search results with keywords "graph data mining". For comparison we used two different search engines: Google, which is based on PageRank algorithm, and AltaVista, which also uses connectivity information, but is more content-oriented. In the extraction phase, only links to HTML pages via HTTP protocol were included into results. After extracting the links, the relative hyperlinks were transformed into absolute addresses, cyclic links to page itself were filtered out, and many links to one destination were considered as one link. After that the 100 best ranked pages were selected to the input graph.

The graphs were quite loosely connected and we found only a couple of self-referring groups with frequency threshold $min_f = 0.75$. The self-referring groups of size ≥ 2 are listed in Tables 3 and 4. In the first graph based on Google results we found three self-referring groups, all of them focusing around MGTS workshops (International Workshop on Mining Graphs, Trees and Sequences) and homepages of program committee members. In the second graph based on AltaVista results we found only two self-referring groups, both of them about Graf-X program for graph-based data mining, and locating on the same cite.

This small experiment supported our initial guess that the self-referring groups are centred around one common theme. However, there are various reasons for self-reference: the occurrence of same people, institutions, products or just the location (cite). It also demonstrated the importance of the initial graph. In our experiment the initial graphs were too sparse to produce more and larger self-referring groups, and the construction of more tightly connected graphs should be further researched.

4.2 Combining concept-maps

In the second experiment, we tested our algorithm for compressing co-operational concept maps. The idea was to combine several concept maps and compress the resulting graph by replacing the strongly self-referring sets by new "super-nodes", which were given new labels.

Table 3: Self-referring groups of size ≥ 2, when the initial graph consisted of 100 best search results by Google. The page numbers are Google ranking numbers.

Page numbers	Contents
22, 34, 86	MGTS-2003 and MGTS-2004 workshop pages and a homepage of a member of program committee
22, 34, 97	MGTS-2003 and MGTS-2004 workshop pages and a homepage of another member of program committee
22, 39, 56	MGTS-2003 workshop page and homepages of a member of committee in English and Japanese

Table 4: Self-referring groups of size ≥ 2, when the initial graph consisted of 100 best search results by AltaVista. The page numbers are AltaVista ranking numbers.

Page numbers	Contents
1 23 58	Articles about Graf-X program
1 58 67	Articles about Graf-X program

The original concept maps were drawn by three university students on topic ”Tanzanian culture and education”. The concept maps were combined simply by matching the nodes, which contained same words or stems, like ”technology”– ”technology exposition” and ”social”– ”society”. All the ”loose” nodes which had at most one relation were dropped. The resulting graph consisted of 48 concept nodes.

After combination, we searched strongly self-referring sets among concepts in the underlying unidirectional graph. We made several experiments with different threshold values min_f. First we run the program with $min_f = 1.0$, i.e. we searched all cliques in the underlying undirected graph. We found 29 cliques, which were either cliques in one of the original concept maps, or contained concept nodes with same words. This did not reveal anything new, as was expected. The threshold valued $min_f = 0.9$ and $min_f = 0.8$ gave the same result. However, with threshold values $min_f = 0.7$ we found new concept groups:

{*Contextualized knowledge, Formal knowledge, Hidden knowledge, ICT, Indigenus knowledge systems* }

{ *Education, Poor internet exposition, Tanzanian culture, Technology, Technology explosion* }

{ *Education, Tanzanian culture, Technological education, Technology, Technology exposition* }

The first concept group could be concluded from one of the concept maps alone, and the third one reveals the contribution to educational technology, which was common for all students. However, the second one is a new discovery which reveals the core of all the concept maps.

With threshold value $min_f = 0.6$ we found 49 self-referring groups, each of them composed of at least four concepts. The largest groups were

{ *Change in educational system, Diffusion into local social systems, Laptop/Desktop computers, Programming tutorial, Simputer, Technological education* }

{ *Change in educational system, Education, HIV/AIDS education, Laptop/Desktop computers, Programming tutorial, Technological education* }

{ *Education, Poor internet exposition, Tanzanian culture, Technological education, Technology, Technology explosion* }

The third one only expands the group found in the previous experiment, but the first two concept groups are totally new. They contain mostly the same concepts: *Change in educational system, Laptop/Desktop computers, Programming tutorial, Technological education.* All of these are inherited from only one concept map, which was very tightly connected and thus dominated the combination.

This experiment gave rise to two remarks: Firstly, our initial combination technique (adding edges between nodes containing same words or stems) had too strong influence on the result. Secondly, one tightly connected map can totally dominate the result, if the others are very loose. However, we found the results quite encouraging: self-referring groups could be applied for a tool, which assists in constructing cooperative concept maps by suggesting closely related concepts.

5 Related work

The existing data mining techniques for graph-based data can be coarsely divided into two approaches: mining frequent substructures, and clustering and searching the graph according to connectivity information. Our new method can be included into the second approach, although it differs from the existing methods.

In the classical *graph-based data mining* (e.g. [9, 1]) the datasets are represented as graphs, and the task of finding frequently occurring substructures reduces to search of frequent subgraphs. The graph-based data mining suits especially well for structured data, but in fact any relational database can be represented as graph. The vertices of graph correspond entities and the edges between them correspond binary relationships.

The general problem of graph-based data mining is following: We are given a set of labelled graphs S, in which each vertex and edge has a label associated to it, and some threshold $\sigma \in]0, 1]$. The task is to find all subgraphs which occur at least $\sigma|D|$ of the input graphs. Two subgraphs are considered identical, if they are isomorphic (topologically identical) and have the same labels on the vertices and edges. Usually it is also required that the graphs are connected, which reduces complexity and is uniform with the goal of finding interesting relations. The problem is very complex because subgraph isomorphism is known to be NP-complete [3].

Thus, our approach differs largely from the classic graph-based data mining task. While we search all instances of some (relaxed) graph property in one huge graph, the typical graph-based data mining task supposes a set of labelled graphs, in which they search frequently occurring subgraphs (i.e. each subgraph has several instances in the collection). However, both methods can be used for simplifying graph-structured data. Cook and Holder [1] suggest that we describe a frequently occurring substructure by some concept, and replace its instances by this concept. Our approach can be used in the similar way: the strongly self-referring sets can be replaced by "super-nodes", which are given a new label. Our future work is to study, whether these labels could also be learnt automatically.

The techniques based on connectivity information are nowadays popular in Internet context. In the clustering-based approaches, the idea is to interpret the sub-graph as a cluster, and the measure of similarity by length of the shortest path between two vertices (for more details see e.g. [5, 4, 11]). For example, the clustering techniques have been applied for analyzing and visualizing the Internet structure [2, 13], constructing taxonomies [8], and extracting web communities [7]. However, the most important applications based on connectivity analysis are the advanced search methods like HITS [6] and PageRank [10].

In HITS algorithm, we evaluate two weights for each page: *authority* value, which tells how relevant information the page contains, and *hub* value, which describes, how good references (links) the page contains. According to main assumption, a page is a good authority if it is pointed by good hub pages, and a good hub, if it points to good authorities. The goal is to find good authorities from the initial subgraph, by updating hub and authority weights until the system converges.

In Page Rank algorithm, only one weight, the "page rank" (PR) is calculated for the current page. The PR value of a page is determined from the PR values of the predecessor pages (pages pointing to it). Thus, loosely speaking, a good authority is also interpreted as a good hub. In addition, the processing order simulates an idealized random surfer on Internet.

The *trawling* algorithm [7] for enumerating web communities can also be described by hubs and authorities. In trawling, the idea is to find subgraphs containing bipartite cliques of $i + j$ vertices. In such a clique each of i vertices (hubs) points to all j vertices (authorities). This method differs from our approach in two ways: first, we do not require full cliques, but only relative clique structure in the underlying undirected graph. Second, our approach does not restrict the web communities to bipartite groups, but also "democratic" groups are considered. In Figure 2 we demonstrate this difference. The first graph and any three vertices (but not all four) of the second graph construct bipartite cliques and are interpreted as a web communities by the trawling algorithm, while our algorithm for self-referring groups selects only the second graph, which constructs a self-referring group with frequency threshold $min_f = 0.75$. It should be noticed that according to the HITS algorithm, vertices 3, 4, 5, 6, 7, and 8 all have equal authority values.

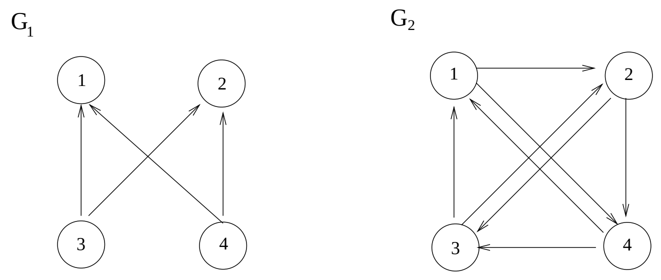

Figure 2: Example of two subgraphs G_1 and $G2$. By trawling algorithm G_1 and any three vertices of G_2 are interpreted as web communities, while our algorithm for self-referring groups produces only G_2 with frequency threshold $min_f = 0.75$.

6 Conclusions

In this paper we have introduced new interesting graph properties, which can be applied for analysis and processing of graph-based data. Especially, we have introduced a notion of self-referring set, which corresponds a relaxed version of a clique. We have given an algorithm for searching all self-referring sets from the given graph, and analyzed its time complexity.

In the future, we are going to study systematically, whether the relaxed graph properties reveal new interesting information and how they can be applied. In addition, we are going to analyze the self-referring sets further by HITS algorithm. The algorithm itself offers many interesting research tasks. For example, we can adapt it for mining self-referring groups in weighed graphs.

Mining the relaxed graph properties has also a general interest in knowledge discovery. The most important research problem for their wider use is how to optimize the algorithm. It should also be studied, whether the existing optimization algorithms of the NP-complete problems could be applied within tolerable error range.

References

[1] D. Cook and L. B. Holder. Graph-based data mining. *IEEE Intelligent Systems*, 15(2):32–41, 2000.

[2] S. Dill, R. Kumar, K.S. Mccurley, S. Rajagopalan, D. Sivakumar, and A. Tomkins. Self-similarity in the web. *ACM Trans. Inter. Tech.*, 2(3):205–223, 2002.

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

[4] I. Jonyer, D. Cook, and L.B Holder. Graph-based hierarchical conceptual clustering. *The Journal of Machine Learning Research archive*, 2:19–43, March 2002.

[5] R. Kannan, S. Vempala, and A. Veta. On clusterings-good, bad and spectral. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, Nov 2000.

[6] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998. Extended version in Journal of the ACM 46(1999). Also appears as IBM Research Report RJ 10076, May 1997.

[7] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tompkins, and E. Upfal. The web as a graph. In *Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–10. ACM Press, 2000.

[8] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. On semi-automated taxonomy construction. In *Proceedings of the 4th Web Data Base Conference*, 2000.

[9] M. Kuramochi and G. Karypis. An efficient algorithm for discovering frequent subgraphs. Technical Report 02-026, Department of Computer Science, University of Minnesota, 2002.

[10] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[11] C.R. Palmer, P.B. Gibbons, and C. Faloutsos. ANF: a fast and scalable tool for data mining in massive graphs. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 81–90. ACM Press, 2002.

[12] R. E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[13] L. Tauro, C. Palmer, G. Siganos, and M. Faloutsos. A simple conceptual model for the internet topology. In *Global Internet*, 2001.