

# Detecting sequences and cycles of web pages

B. L. Narayan and Sankar K. Pal

Machine Intelligence Unit,

Indian Statistical Institute,

203, B. T. Road,

Calcutta - 700108, India.

E-mail: {bln\_r, sankar}@isical.ac.in.

## Abstract

*Cycle detection in graphs and digraphs has received wide attention and several algorithms are available for this purpose. While the web may be modeled as a digraph, such algorithms would not be of much use due to both the scale of the web and the number of uninteresting cycles and sequences in it. We propose a novel sequence detection algorithm for web pages, and highlight its importance for search related systems. Here, the sequence found is such that its consecutive elements have the same relation among them. This relation is measured in terms of the positional properties of navigational links, for which we provide a method for identifying navigational links. The proposed methodology does not detect all possible sequences and cycles in the web graph, but just those that were intended by the creators of those web pages. Experimental results confirm the accuracy of the proposed algorithm.*

## 1. Introduction

The Web has a complex graph structure and is usually modeled as a digraph [3, 6], with the pages forming the vertices and the links between them being the arcs. We use the following terminology related to graphs [8]. A digraph or directed graph  $G$  is an ordered pair  $(V, E)$ , where  $V$  is a set of vertices and  $E$  is the set of arcs or directed edges between these vertices. We denote the arc from vertex  $i$  to vertex  $j$  by  $e_{ij}$ . A path in a digraph is a sequence of vertices  $x_1, x_2, \dots, x_i, x_{i+1}, \dots, x_n$  such that  $x_i$  and  $x_{i+1}$  are connected by an arc in  $E$ , and the vertices are not repeated. A cycle is a sequence of vertices  $x_1, x_2, \dots, x_i, x_{i+1}, \dots, x_n, x_{n+1}$  such that  $x_{n+1} = x_1$  and if  $x_{n+1}$  is removed, the remaining part is a path. We denote the path and cycle defined here by  $x_1.x_2 \dots x_i.x_{i+1} \dots x_n$  and  $(x_1.x_2 \dots x_i.x_{i+1} \dots x_n)$ , respectively. A digraph is strongly connected if there exists a

path from any vertex  $x_1$  to any other vertex  $x_n$  in  $V$ . The in-degree (out-degree) of a vertex is the number of arcs leading to (going out of) the vertex. In what follows, we use the term *sequence* to refer to a path in a digraph.

Link analysis algorithms dealing with the web graph assume that it is strongly connected, and in practice, this is ensured by adding artificial arcs to the graph [6]. According to recent estimates, the number of pages on the WWW is several billion, with Google claiming to have indexed over 8 billion pages (Source: <http://www.google.com/googleblog/2004/11/googles-index-nearly-doubles.html>).

In such massive digraphs, with several arcs between the vertices, sequences and cycles are a common phenomenon. However, only a few of these sequences and cycles were conceived at the time of their creation by the authors of the constituent pages. While the reason behind creating such special structures in the web graph may be convenience or Search Engine Optimization, sometimes it borders on malicious intent or spam. Moreover, with web authoring styles varying widely, the same kind of content may be presented in a single page, or broken up into several pages. Such dissimilarities bring about large differences in link based analyses. So, for the sake of uniformity and consistency during comparison of web pages, the detection of these special structures is essential.

In the present article, we propose a novel algorithm that detects sequences of web pages which were intended by their creator to be traversed in that order. The algorithm broadly consists of three parts. The first deals with identifying navigational links. The second part finds ordered triplets of pages  $(A, B, C)$  such that  $C$  is related to  $B$  in the same way as  $B$  is related to  $A$ . These form the segments of the sequences to be identified. The final part concatenates the detected segments, to obtain sequences as long as possible. Certain speedups and improvements over this naive algorithm are also discussed. Experiments performed on a data set containing cycles of HTML pages show the performance

of the proposed methodology to be satisfactory.

This article is organized as follows: Section 2 lists the background work related to identifying navigational links and detecting cycles in graphs. In Section 3, we describe the proposed methodology for detecting sequences and cycles of web pages. We discuss the characteristics and importance of our methodology in Section 4, which is followed by Section 5 where we present our experimental results. We draw our conclusions in Section 6.

## 2. Related Work

The methodology proposed in this article relies on both identification of navigational links and detecting sequences and cycles from graphs. Here, we survey the literature on both these topics.

Identification of navigational links has been studied in many works available in the literature [2, 4, 16, 7]. Bharat and Mihaila [2] disregarded links between pages on affiliated hosts. Two hosts were called affiliated if they shared the first three octets in their IP addresses, or the right-most non-generic tokens in their hostnames was the same. Borodin, *et al* [4] identified and eliminated navigational links using a very similar idea. However, not all navigational links are detected in this manner [4]. Moreover, this approach is quite severe, and some links connecting pages wholly on the basis of their content would be wrongly classified as navigational links. This would especially be the case where, say, a member of an organization links to content on a colleague's page.

Yi and Liu [17] detect navigational links along with banner ads, decoration pictures, *etc*, which were collectively termed web page noise, in a set of pages by constructing a compressed structure tree (CST). The diversity of tags in an element node of the CST determines how noisy the corresponding block in the web page is.

In the above mentioned studies, navigational links have been treated as noise and discarded. Also, in [17], they have not been separated from other kinds of web page noise like banner ads, decoration pictures, *etc*.

Sequence or path detection in graphs has largely been restricted to finding shortest paths, which conveys the feeling that sequence detection, by itself, was not considered very interesting. On the other hand, cycle detection has been widely studied under two related areas, namely, pseudo-random number generation [5, 11] and graph theory [14, 9]. In pseudo-random number generation, a sequence of numbers is generated by applying the same function to the last generated number. The objective is to detect cycles in the sequences of random numbers being generated. By adding arcs between consecutive elements of such sequences, this problem may be studied as a graph cycle detection algorithm. Since, every element of such sequences has out-

degree one, stack based algorithms are employed for cycle detection [11].

Detecting cycles in general graphs and digraphs is more complicated, because the out-degrees of the vertices may be more than one. There are several digraph cycle detection algorithms, as evident from the opening statement of [9]. Some of them like the one by Read and Tarzan depend on depth-first search while others like the Szwarcfter-Lauer algorithm employ a recursive backtracking procedure which search strongly connected components of the graph [9]. For the web graph, where the whole of it may form a single strongly connected component, such algorithms may not be of much use.

We now look at an algorithm that detects sequences and cycles of interest from the web graph.

## 3. Proposed Sequence Detection Algorithm

We begin this section with the motivation for why yet another cycle detection algorithm is needed.

### 3.1. Motivation

Consider the following statement.

A: <http://www.stanford.edu/index.html>,  
<http://www.yahoo.com/index.html> and  
<http://wi-consortium.org/index.html>  
 form part of a cycle of web pages.

One can easily be convinced that Statement A is true, primarily because of the high in- and out-degrees of the chosen web pages. Moreover, Statement A would hold true even in conjunction with any of the following statements:

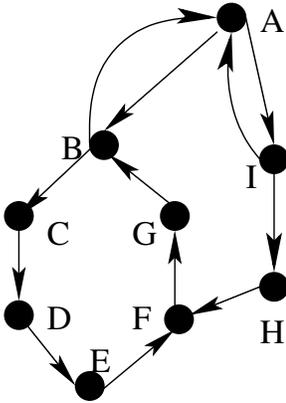
- B: The three pages appear in a prespecified order
- C: The length of the cycle is exactly 20
- D: The cycle contains no web page from a `co.uk` or a `.net` domain

The ease with which these statements have been made, and can be verified to be true, provides an idea of how huge the number of cycles involving these three web pages would be. This in turn would imply that the total number of cycles on the web (without imposing any conditions) would be extremely large, say of the order of billions, with the number of vertices (web pages) being a few to several billion. Enumerating all such cycles (say, by an algorithm as mentioned in [13] or [9]) would be tedious and computationally expensive, and several individual cycles that have been formed just by chance may not be interesting at all. Perhaps, of more interest, and consequence, would be statements about the absence of cycles of a certain kind.

However, there are some that stand out in this galaxy of cycles. These are the objects of our interest in the present investigation.

### 3.2. Structures of Interest

With the help of Fig. 1, we characterize the objects we are interested in detecting from the web graph. Fig. 1 shows



**Figure 1. A strongly connected digraph**

a directed graph, say  $\mathcal{G}$ , with the vertices labeled  $A$  to  $I$  being web documents and arcs being the links between them. This graph is strongly connected because any vertex can be reached from any other vertex.

We now look at the directed cycles of the graph  $\mathcal{G}$ . There are two of them of length greater than two, namely (B.C.D.E.F.G) and (A.I.H.F.G.B), each consisting of six vertices. Although, technically, both the above cycles may be reported by a cycle detection algorithm, we observe that the former looks more regular compared to the latter. Also, the names of the web pages are also quite suggestive that the first cycle was intended right at the outset, while the second one was formed just by chance.

We are interested only in cycles of the first kind which exhibit a regular pattern. In the present article, we study sequences and cycles of web pages that have been deliberately so generated by their author(s). At the time of creation of these pages, it was intended that a surfer would view them in a specific order.

### 3.3. Detecting Sequences of Web Pages

Mathematically, a sequence  $\{X_n\}$  is defined as a function with its domain being a subset of  $\mathcal{N}$  (the set of non-negative integers) and taking values in some set  $S$ . In our case, each value is a web page. The kind of sequences that we are interested in may be defined as an ordered set of elements where the relation between any two consecutive

elements remains the same. In other words,  $X_{n+1}$  is related to  $X_n$  in exactly the same way as  $X_n$  is related to  $X_{n-1}$ .

We employ this notion for detecting sequences of web pages by considering the type and position of links between them for defining the relation “next in sequence”. In order to detect a sequence, we first detect its segments whereby, we find each triplet (three consecutive elements) of the sequence. Such triplets are found by examining the similarity of the pairs of links  $e_{AB}$  and  $e_{BC}$ , where  $A$ ,  $B$  and  $C$  are three web pages present in the web graph under consideration.

Detection of the desired triplets of the form  $(A, B, C)$  is performed in two steps. First, for each page  $B$ , we identify the navigational links leading into and out of  $B$ . Then, each pair of links consisting of an inlink and an outlink are checked to see if they form a part of a sequence. All such triplets are stored and finally, they are merged to obtain sequences of web pages. Closed sequences are flagged as cycles. Among the pages in a cycle, any page that has inlinks from several of the other pages is identified as the “start” or “contents” page.

We consider only navigational links because the other links in the page, usually available in the content, are present mainly for leading to a reference page, or for a lookahead. It is generally the case that though a link is provided in the content for ready reference, the surfer is expected to return to the same page and continue by following a navigational link from that page.

Having specified the basic plan, we now elaborate on the implementation aspects of identification of navigational links and sequence detection.

### 3.4. Implementation

Identification of navigational links is of primary interest while detecting sequences and cycles on the web. To check if a given link is navigational or not, we employ the following method which is not as severe as those in [2, 4], and is simpler than that described in [17].

Our approach for detecting navigation links is relatively simple compared to several other approaches mainly because we are not interested in various other structures like content blocks and templates, *etc.* We assume that navigational links mostly occur unaccompanied by text, the likely reason being that the anchor text is descriptive enough for the user to understand where the link leads to. Since most navigational links lead to neighboring pages, which in turn, have a similar look and feel, the user is generally expected to be familiar with the navigational links and no further description is required.

For a given web page, we look at the link elements (delimited by “<a href=... > ... </a>”), and content elements which may either be a text token or an image.

```

TTTTLLLLLLLLTLLTLLTTTTTTTTTTTTTTTTTTTTTTTTL
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTTTLTTTTTTTTLTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

```

**Figure 2. String of links and content elements of a web page**

We ignore all other content types for the time being. A typical such sequence of text and links looks like in Fig. 2 (this particular example sequence corresponds to the URL <http://clgiles.ist.psu.edu/index.html>). Here, 'L' and 'T' denote a link and content element, respectively.

Note that the navigational links are clumped together, whereas, the other links in the page are surrounded by text. We formalize this notion as follows: Let the string formed as above be denoted by  $S$  and let its length be  $n$ . Also, let  $k$  be such that  $S[k]$  is 'L'. Let the set  $\{S[k-5], S[k-4], S[k-3], S[k-2], S[k-1]\}$  be called the left neighborhood of  $S[k]$  and  $\{S[k+1], S[k+2], S[k+3], S[k+4], S[k+5]\}$  be the right neighborhood (one of them may have less than five elements, too, if  $k < 6$  or  $k > n - 6$ ). Now, if either the left or the right neighborhood of  $S[k]$  contains at least two 'L's, then we label  $S[k]$  as a navigational link. Otherwise, the link is considered to be a link to content. Basically, we are looking at the two neighboring windows (of size 5) to see if it is predominantly text in that locality or not. It is not clear what the optimal choice of window size would be, but we find that our present choice is reasonably good.

This procedure might wrongly mark some non-navigational links, too, but we do not make it any more sophisticated as it is fast and simple in its present state. Since, our end goal is not to find navigational links but to detect sequences of web pages, this method suffices for our purposes. Among the navigation links found, we further classify them into two kinds: top and bottom. We assume that a navigational link would appear at the ends of a page (and not at the middle). If  $k$  mentioned above is smaller (greater) than  $\frac{n}{2}$ , we label the link as top (bottom).

We now represent the features of the link by using a byte of information. The first (second) bit denotes whether the link is a top (bottom) navigation link. The first (latter) three of the remaining six bits keep the count of the link from the top (bottom). So, for example, the second navigation link from the top would have 1\*001\*\*\* and the third navigation link from the bottom would have \*1\*\*\*010 (the values at '\*\*' are determined by the position of the link at the other end

of the page). We provide another example to clarify this. A navigational link that appears as the fourth link from the top, and appears again as the last link in a page, would have the information byte 11011000 associated with it. Since we use only three bits to store the count of the link, only the top and bottom eight navigational links are retained, and the rest are not called navigational links and the corresponding bits at the beginning are set to 0.

Now that the navigational links have been identified, our task is to find the successor, if any, of each such link. For each page  $B$ , we look at the navigational links that lead to  $B$ . For any such link  $e_{AB}$ , we look at its information byte and identify the link on  $B$  with the same information byte. If there exists one such link, say  $e_{BC}$ , we call it the successor of  $e_{AB}$  and the triplet is noted as  $A.B.C$ , and indexed by  $B$  for ease of retrieval. What we are doing in terms of finding the successor of a link is trying to identify the link that appears in almost the same position as  $e_{AB}$ . Since, the user has reached page  $B$  by clicking a link at that very position on  $A$ , it is very likely that the link on page  $B$  at the same position would be followed. This is what we mean by saying that page  $C$  is related to  $B$  in the same way as  $B$  is related to  $A$ .

In this manner, all links which are present in sequences are assigned a successor and are stored as triplets. The relation between the consecutive elements of the sequence might not hold at the extreme ends of the sequences, but all triplets from the interior would be found. These form the basic building blocks of larger sequences. Our task is now to concatenate these segments into the actual sequence. Let  $x_1.x_2 \dots x_n$  be an existing subsequence (initially, all of them are of length 3). A subsequence  $y_1.y_2 \dots y_m$  is prepended to it if  $y_{m-1} = x_1$  and  $y_m = x_2$ , and the new subsequence becomes  $y_1 \dots y_m.x_3 \dots x_n$ . Similarly, it is appended if  $y_1 = x_{n-1}$  and  $y_2 = x_n$  and then the new subsequence becomes  $x_1.x_2 \dots x_n.y_3 \dots y_m$ .

We traverse these triplets, merging them into larger and large subsequences to obtain all the desired sequences. Finally, when we have all the sequences, we check if there are any cycles to be found. Note that this scheme of cycle detection corresponds to the one used for detecting cycles in sequences of pseudo-random numbers. We do not need the more general graph cycle finding algorithms because we are now dealing with only a sequence of web pages and not the whole web graph. We refer to this algorithm as SC1.

### 3.5. Speedups and other improvements

We now have a basic algorithm (SC1) to detect sequences and cycles of web pages. Certain modifications may be performed to improve its performance.

Navigational links might be marked out in the HTML content of the page. This is all the more common nowa-

days with a number of pages being generated automatically or being converted from other formats such as MSWord or  $\text{\LaTeX}$ . If we assume that the appearance of the words *navigation* or *navig* in HTML comments or names of blocks genuinely indicates that the block is a navigation panel, identifying navigational links becomes straightforward. This obviates the tokenization of the pages and counting the neighbors of links, and hence, improves the speed of the algorithm. Some web pages may also provide information about the relations with other pages by means of `link` elements. This again makes the task of detecting sequences of pages trivial as the relations between them are specified beforehand. However, authenticity of these specified relations is not guaranteed. We have not implemented these speedups.

When a segment  $A.B.C$  is detected, we check the navigational links of each of those to see if there is a common URL that has the same information byte associated with it in each of the three pages. Such URLs usually tend to be the “Contents” or “Start” page. We then look up the “Start” page and check if the pages  $A$ ,  $B$  and  $C$  appear in it, either in that order or the reverse order. If they do, we look at all the links that are listed in the “Start” page, and create a sequence of pages in that order. Thus, in case there are two cycles in opposite directions, the correct direction is also identified. SC1 modified in this manner is named SC2.

It may be noted that the algorithm, in its present state, detects sequences and cycles of web pages that are available in the current crawl only. A single page being missed out in the crawl, which may happen due to various reasons, would result in the sequence or cycle being missed out. Subsequences of the undetected objects may be reported, leading to suboptimal performance. For this reason, we modified our algorithm to consider the information contained in the URLs of the detected segments. If a segment  $A.B.C$  has been detected, and the page  $C$  is missing, then we look at the characters by which URLs of the pairs  $(A, B)$  and  $(B, C)$  differ. If the differing characters are numerals, we compute the difference in numeric value of the two URLs in each pair and check if they are the same. If so, we extrapolate the URL of  $C$  to obtain a URL of another page, say  $D$ . If the page  $D$  exists, we add the triplet  $B.C.D$  to the existing set of segments. Similar extrapolation may be tried with the ASCII values when the URLs do not contain numerals. This version of our sequence and cycle detection algorithm is referred to as SC3.

The information about a navigational link that we store may not convey the exact position where it would be displayed due to a variety of reasons. A more accurate way would be to locate the actual position in the page where the link would be placed by analyzing the HTML structure. This information may require more than a byte of storage space.

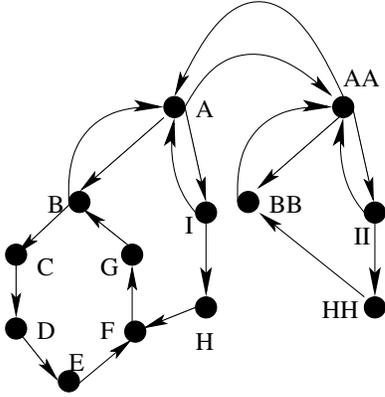
## 4. Characteristics and Uses

We now discuss some characteristics of the proposed sequence detection technique. SC1, SC2 and SC3 detect maximal sequences, but do not report sequences that have been formed just by chance. For example, these algorithms would surely not report any cycle that satisfies Statement A (of Section 3.1). Thus, despite the existence of numerous cycles in the web graph, these algorithms would report only a handful of cycles and sequences.

This methodology does not incorporate finding second level sequences and cycles, *i.e.* sequences of sequences and cycles, *etc.* This may be required for the sake of uniformity in the case where a single page is split into not just a single sequence but into multiple sequences at different levels. If such higher level sequences are to be determined, it is not clear how many levels one should consider.

We now come to the utility of detecting sequences and cycles of web pages. One apparent use of detecting sequences is to merge the pages in that order. What were individual pages prior to merging, will now become sections of a single page. In this manner, the number of vertices in the web graph may be reduced, which is computationally advantageous.

The proposed methodology may also be employed for bringing a consistency into the computation of page ranks despite the varying authoring styles. Some web page authors may prefer to split their content into a sequence of pages, while others may like to keep it all in a single page. These differences have an impact on the page ranks of the pages. To see this, we revisit our example in Fig. 1, where, our cycle detection algorithm would detect only the cycle (B.C.D.E.F.G). Now, consider a page BB that has the union of the contents of all these individual six pages. We look at how the above mentioned cycle of pages would fare against the page BB with respect to page ranking. To this end, we create a new example (Fig. 3), made of two strongly connected components and connect them. We note that the two components of this directed graph, one consisting of the vertices AA, BB, HH and II, and the other with the remaining vertices, have a correspondence between them, with BB corresponding to the cycle (B.C.D.E.F.G). This graph is treated as the web graph and the (unnormalized) PageRanks [6] of these nodes are shown below (Table 1) (PageRanks computed at <http://www.markhorrell.com/seo/pagerank.asp>). Although, content-wise, the pages B to G, put together, are equivalent to the page BB, their ranks are different. In general, for the case of the actual web graph, it would be much more difficult to ascertain the effect of merging a sequence of pages, but, as evident from this example, it is not necessary that the ranks of the constituent pages remain the same. Thus, in the case of web search, the results



**Figure 3. One component of the graph has a cycle, the other has it merged**

**Table 1. PageRanks for the pages in Fig 3**

Page	Rank
A	1.5817196775
B	1.6174242046
C	0.8410460047
D	0.8612356001
E	0.8878664945
F	1.2442512723
G	1.2137333638
H	0.4047855488
I	0.5955428314
AA	1.7043599767
BB	0.9933097604
HH	0.4188371411
II	0.635888124

returned may be influenced by whether all the content has been kept in a single page or has been split into a number of pages. In this respect, our algorithm may be utilized to maintain uniformity among various portions of the web, so that comparisons between web pages become less dependent on authoring styles.

There is yet another and more important advantage of merging such sequences of web pages. Consider two pages  $X$  and  $Y$  which form part of a sequence. Suppose that the pages  $X$  and  $Y$  contain terms  $t_1$  and  $t_2$ , respectively, and are the most authoritative, individually, for the respective terms. On the other hand, a page  $Z$  which is not as authoritative on either  $t_1$  or  $t_2$  may be returned ahead of both  $X$  and  $Y$  as the most authoritative page containing both  $t_1$  and  $t_2$ . Note that  $X$  may not even contain the term  $t_2$ , and even if it does, may not be authoritative for it. However, when both  $X$  and  $Y$  form part of a merged sequence, they would correctly be

returned ahead of  $Z$  for the query consisting of  $t_1$  and  $t_2$ . In reality, pages like  $Z$  behave like pages containing spam words. They are not at all authoritative for the terms they contain, but nevertheless have terms which co-occur very rarely. Such pages permeate the initial portion of the search results when a user enters such a combination of terms, primarily because of the lack of better pages containing all the query terms.

It may be noted that the pages  $X$  and  $Y$  need not be physically merged together. A search engine may perform the analysis as if their content is combined together and output both the pages in response to a query. In this manner, sets of web pages may be output by a search engine in response to a (multiple term) query, as opposed to the current trend of providing single pages as search results.

All the above advantages are dependent on the proper detection of such sequences and cycles. The next section shows that this is indeed the case with the proposed methodology.

## 5. Experimental Results

We have conducted some preliminary experiments and present the results here. We have tested the proposed algorithm on a small collection of web pages that we downloaded from the Internet available from <http://docs.python.org>. All files under this website were obtained (available online as a tar bziped file <http://python.fyxm.net/ftp/python/doc/2.4/html-2.4.tar.bz2>). There were a total of 1412 HTML files. Of these, 1408 HTML files were under 10 subdirectories. Under each subdirectory, the HTML pages form cycles. A quick inspection at the pages contained in this data set reveals that there are at least thousands of possible (not necessarily disjoint) cycles, of which only ten are interesting (because they were so generated and placed in subdirectories).

We applied our SC2 algorithm to the given HTML pages and it output the 10 obtained cycles. No sequences were output. We checked the output cycles manually (the lengths of the cycles are 9, 18, 21, 32, 64, 83, 99, 116, 120 and 836) and found the output to be completely accurate.

To check how SC3 fares when some pages are missing, we deleted one page from each of the 10 subdirectories. It was ensured that none of the start pages were deleted. SC3 output exactly the same set of cycles as SC2 earlier.

Then, we deleted two consecutive pages from each of the subdirectories. When we ran SC3 over these reduced set of pages, the output was that there are 10 cycles and 10 sequences, as expected. This is a consequence of the fact that we do not extrapolate beyond a single page. This can perhaps be rectified by extrapolating URLs beyond a single page. The above results confirm that the proposed method-

ology provides a satisfactory performance for the task of finding sequences and cycles of web pages.

The chosen data set had very little noise in its pages and there was no error in the identification of the navigational links. In other words, all and only truly navigational links were reported as navigational links in the present experiment. This, in turn, has fostered the performance of SC2 and SC3.

## 6. Conclusions

In this article, we have introduced a novel methodology for the important task of detecting sequences and cycles of web pages. We have explained why detecting all possible sequences and cycles of web pages is neither feasible nor interesting. Hence, the need for the proposed methodology which detects only the few interesting sequences and cycles which were created to be traversed in that order. The proposed algorithms SC1, SC2 and SC3 use varying levels of domain knowledge, but essentially capture the same notion that consecutive elements of a sequence have a constant relation between them. Navigational links in web pages are identified and their positional information is obtained. Sequences are tracked by traversing pages that are obtained by following links with the same positional information. Experiments were conducted on a data set obtained from doc.python.org and only the relevant cycles and sequences were output out of thousands of potential cycles. While the results on this data set are satisfactory, we note that this is a very nice collection. It would be interesting to see how the proposed methodology fares when applied to an unorganized collection of web documents.

## 7. Acknowledgment

The authors are thankful to the anonymous reviewers for their helpful comments. The first author's doctoral fellowship is funded by INSEAD, France.

## References

- [1] Z. Bar-Yossef and S. Rajagopalan. Template detection via data mining and its applications. In *Proceedings of the eleventh international conference on World Wide Web*, pages 580 – 591, Hawaii, USA, 2002.
- [2] K. Bharat and G. A. Mihaila. When experts agree: using non-affiliated experts to rank popular topics. In *Proceedings of the tenth International World Wide Web Conference*, pages 597–602, Hong Kong, 2001.
- [3] P. Boldi and S. Vigna. The webgraph framework i: Compression techniques. In *Proceedings of the 13th international conference on World Wide Web*, pages 595 – 602, New York, May 2004.
- [4] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas. Link analysis ranking: algorithms, theory, and experiments. *ACM Transactions on Internet Technology*, 5(1):231–297, February 2005.
- [5] R. P. Brent. An improved monte carlo factorization algorithm. *BIT*, 20:176–184, 1980.
- [6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web*, pages 107 – 117, Brisbane, Australia, 1998.
- [7] B. D. Davison. Recognizing nepotistic links on the web. In *Proceedings of AAAI*, 2000.
- [8] F. Harary. *Graph Theory*. Narosa Publications, 1988.
- [9] G. Loizou and P. Thanisch. Enumerating the cycles of a digraph: A new preprocessing strategy. *Information Sciences*, 27:163–182, 1982.
- [10] P. Mateti and N. Deo. On algorithms for enumerating all circuits of a graph. *SIAM Journal of Computing*, 5:90–99, 1976.
- [11] G. Nivasch. Cycle detection using a stack. *Information Processing Letters*, 90:135–140, 2004.
- [12] S. Sarawagi and V. G. V. Vydiswaran. Learning to extract information from large domain-specific websites using sequential models. *ACM SIGKDD Explorations Newsletter*, 6(2):61–66, December 2004.
- [13] J. L. Szwarcfiter and P. E. Lauer. A search strategy for the elementary cycles of a directed graph. *BIT*, 16:192–204, 1976.
- [14] J. T. Welch, Jr. A mechanical analysis of the cyclic structure of undirected linear graphs. *Journal of the ACM*, 13(2):205–210, April 1966.
- [15] B. Wu and B. D. Davison. Identifying link farm spam pages. In *Proceedings of the Fourteenth International World Wide Web Conference*, pages 820–829, Chiba, Japan, May 2005.
- [16] L. Yi and B. Liu. Web page cleaning for web mining through feature weighting. In *proceedings of Eighteenth International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 2003.
- [17] L. Yi, B. Liu, and X. Li. Eliminating noisy information in web pages for data mining. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 296 – 305, Washington, D.C., 2003.
- [18] W. Zhulong, Y. Hao, and N. Fumihito. Automatic special type website detection based on webpage type classification. In *Proceedings International Workshop on Web Engineering*, Santa Cruz, 2004.