Using Delta Encoding for Compressing Related Web Pages

Zan Ouyang, Nasir Memon and Torsten Suel
Dept. of Computer Science,
Polytechnic University,
Brooklyn, NY 11201
November 19, 2000

Abstract

This paper examines the compression benefits that can be obtained by exploiting the possible similarities of pages on the World Wide Web.

Two of the elements of our approach are innovative. First, we provide a new differential compression schemes for similar web pages, which we call *wdelta*, it is competitive with the current delta algorithm *vdelta*, and quite outperform diff+gzip. Secondly, we present an efficient algorithm for computing the optimal page ordering for related pages. This algorithm has time complexity $O(n^2)$ for a n-page group. In order to accelerate download speed over narrowband links (like, dial-up modems wireless communication), a heuristic method, is also presented here to reduce the time complexity and simplify the decoding chain, that is, for every clustering group, an anchor page is chosen to act as an base page for all the other pages to have the differential compression.

The result is very promising, compressing a page based on another related page can obtain large amount of increase in compression ratio.

1 Introduction

Perhaps one the most significant limitations of the World-Wide-Web as we know it today is the often intolerable delay, or latency, that is encountered while retrieving web content. Generally the following techniques are used to alleviate this problem [1]:

- **Caching**: increase the size of browser caches.
- **Prefetching**: push the predicted page to the user ahead of time.
- **Delta compression**: efficiently transmit the difference between two web pages.
- **HTML compression**: compress the HTML files themselves.
The algorithms we present in this paper relate to Delta compression. Delta compression algorithms, i.e., algorithms that compute differences between two files or strings to reduce communication or storage costs, are not new [2]. They basically exploit the fact that the sender and receiver both possess a reference file and the file to be transmitted is very similar to this reference file. Hence, transmitting only the difference between the two requires only a significantly smaller number of bits. In applications like software version control, a delta is often one or two orders of magnitude smaller than the original. Delta compression is natural to use in web applications where the content of web pages changes slowly. When a client requests a page, its delta can be sent with respect to the previous version located in the clients cache, thereby resulting in significant savings as shown in [xxx]. Although delta compression has been demonstrated to be very useful, the focus of its use has mostly been on representing content that is only slowly changing with time. In this paper we show that delta compression is in fact useful in a wider variety of situations. Specifically, we investigate algorithms and techniques aimed at broadening the application of delta compression to the following situations:

- Web pages with same URL but different versions.
- Web pages with similar structure but different content.
- Web pages with similar content but different structure.
- Web pages with links to each other.

The potential applications that can benefit from our work include: 1) storing web pages in compressed form in a main memory to reduce disk paging, 2) compressing a collection of web pages to facilitate faster downloads for offline browsing, 3) compressing web pages to give client more offline content with the same storage capacity, 4) compression of the contents within a client’s cache (especially when the client is a hand held device with limited memory) and 5) archival of web sites.

The rest of this paper is organized as follows: in Section 2, we describe a simple delta compression scheme that is shown to work well with web content. In Section 3, we examine the problem of compressing a collection of web pages using delta compression. We show that the problem of obtaining optimal compression, given a specific delta compression algorithm, requires finding a maximum weight directed spanning forest of a graph [3,4]. For an n-page website, the algorithm runs in $O(n^2)$ time. As a result, the entire website, is compressed better if we use delta compression instead of compressing pages individually. For some cases, however, the optimal solution is not a practical method for a large number of pages. Also, the optimal solution may require in the worst case, decompression of all the pages in order to access a given page. Hence in Section 3 we also look at some heuristics that are designed to alleviate this problem. Finally, in Section 4, we present some experimental results using techniques and algorithms presented in Section 2 and Section 3. Our experiments indicate that the proposed techniques can give significant savings for collections of web pages that share sufficient common structure and/or content.
2 Computing delta for the webpages

The two main delta compression algorithms in use today are diff+gzip and vdelta. Using diff to find the difference between two files and then using gzip to compress the difference is a simple and straightforward way to do delta compression. Nevertheless, it is quite commonly used [xxx]. Vdelta, on the other hand, is a relatively new technique that integrates both data compression and data differencing [5]. It is a refinement of W.F.Tichy’s block-move algorithm [6]. It essentially generalizes the well known Lempel-Ziv technique by allowing string matching to be done both within the target data and between a source and target data. Although vdelta offers big improvements over a naive diff+gzip approach, it has been mainly been designed for software version control applications and does not perform well when applied to HTML files, which represent our main concern in this work.

If we examine delta compression methods, they mainly consist of a building difference stage and an encoding difference stage. Keeping this in mind, we develop a simple delta compression algorithm, wdelta that combines the ideas of vdelta and gzip, allowing for string matching to be done both within the target data and reference data. Gzip uses previously seen text as a dictionary, it replaces variable-length phrases in the input text with fixed-size pointers into the dictionary to achieve compression[7], the amount of compression depends on how long the dictionary phrases are, and how large the window into the previously seen text is.

Given two files that are similar, concatenating the two and then gzip’ing them would surely result in better compression than compressing them individually with gzip. The same applies to a collection of related files. However, there are two obvious situations in which this approach will not work well. First, if we have two files that are very long, for example, more than 32k which is the default gzip window size. In this case, even though the two files may be very similar, while parsing the second file one may not find the best match as it is located in the first file which is already beyond the leading edge of the sliding window. Second, for every match found, gzip will encode the distance as a relative distance from the current position, while for two similar files, this is not economic. Our wdelta solves these problems by applying a different sliding window scheme and a different distance coding method. In the rest of this section we give two variations of wdelta.

2.1 Two sliding window approach

This approach is suitable for two files that are similar in structure, that is, the similar content are located in the similar positions. Figure 1 shows the two-sliding window scheme.

For both reference and current files, there are sliding windows. The whole reference file
is first read in, and a hash table is constructed for fast string matching. Each index is a position which is keyed by the three bytes starting at that position. Then the current file is read in three bytes at a time. The sliding window for the current file is contained entirely in the current file, while the corresponding positions in reference form the sliding window in the reference file. If the longest match is found in the reference file, the distance between pointer M and R is encoded. Else if the match is found in the current file, the distance between R2 and C is encoded. We use separate entropy codes to encode the distance in each of these situations. It is easy to see that this two-window scheme will work well even if we have very long files and also allows us to encode copy locations more compactly.

2.2 Last-copy scheme

When the second file add some bulks of data or delete bulks of data from the reference file, we will put a pointer in the position of last match of reference file as Lr, and encode the new match distance corresponding to this pointer. Figure 2 shows the Last-copy scheme.

The example above shows that when the second file (current) is changed by inserting some strings into the first file (reference file), then we first have a copy beginning from position Lr, and for the second copy, we will encode the match position X corresponding to Lr + copyLength, in this example, the copy length is 26, and the distance between X and Lr + copyLength is 0. Obviously, we haven’t included all the possible situation into this scheme and the effect of the scheme also depends on the characteristics of the two input files. So, two-sliding window scheme is a more general scheme. In this paper, we will use *wdelta* as our compressing method.

In figure 1 we provide a comparison between diff+gzip, vdelta and wdelta. It can be seen that wdelta provides the best compression. We also compared wdelta and vdelta on the benchmark data set provided in [xxx] which represents two different versions of emacs and gcc. Even on this benchmark wdelta gives us slightly improved results though we omit the details for the sake of brevity.

3 Determining an Optimal Delta Encoding Scheme for a Collection of Files

Now that we have an efficient delta compression technique, in this section we investigate its application to compression of a collection of related web pages. The question that arises is, given a collection of files, how do we efficiently represent the entire collection my means of appropriate deltas? We show that the problem of finding an optimal delta encoding for a collection of files by taking pair-wise delta’s can be formulated as a problem on a weighted directed graph which has a known and efficient solution [4,8].

we will think about the following problem: What is the best order for the related pages to be compressed in order to get the best compression? It is simple for web pages with same URL but are different versions because we can just use the chronological order, assume the previous version is available on both sides. But for the other situations, it is not so obvious. For example, if page 3 acts as a good predictor for page 1, compressing page 3 first will have more compression benefit than compressing page 1 first.
Suppose we have N pages, by using \( wd \) for every pair of pages, we can get a matrix with \( D = d_{ij} \) where \( d_{ij} \) (i,j=1...N) represents the cost (in bits) of delta compression when page i is compressed based on page j. For example, if we have a 3-page website, the \( d_{ij} \) matrix will be:

\[
\begin{pmatrix}
  d_{11} & d_{12} & d_{13} \\
  d_{21} & d_{22} & d_{23} \\
  d_{31} & d_{32} & d_{33}
\end{pmatrix}
\]

where, for example, \( d_{12} \) is the delta value when page 1 is compressed based on page 2.

Correspondingly, we define a vector \( O = o_j \) (j=1...N) that represents the original page size of each file j. In the following sub-sections, we will examine various ways of choosing a page ordering when given Compression Delta matrix \( D \) and Original size vector \( O \).

### 3.1 Optimal Solution

The problem of finding an optimal delta encoding for a collection of files by taking pair-wise delta’s can be formulated as a problem on a weighted directed graphs which has a known and efficient solution [4,8]. Specifically, we construct a complete directed graph \( G=(V,E) \) such that each edge \( E_{i,j} \) has a corresponding weight \( S_{ij} \) that represent the compression savings attained by compressing file i based on file j. For a N page website, the graph G will also include an N+1’th node, which is the null node used to represent the compression savings when a page is compressed by itself. So we define the \( S_{ij} \) as the following:

\[
S_{ij} = o_{ij} - d_{ij}, i, j = 0, 1, \ldots, N
\]  

For the example in Section 3.2, its Compression Saving Matrix S will be a 4 \times 4 matrix:

\[
\begin{pmatrix}
  S_{00} & S_{01} & S_{02} & S_{03} \\
  S_{10} & S_{11} & S_{12} & S_{13} \\
  S_{20} & S_{21} & S_{22} & S_{23} \\
  S_{30} & S_{31} & S_{32} & S_{33}
\end{pmatrix}
\]

So the problem of finding an optimal compression order is transformed into the problem of finding a maximum weight directed spanning forest of the graph G as defined above. It is known that a maximum weight directed spanning forest can be found in \( O(|V|^2) \) time for a dense graph[3,4].

### 3.2 Sub-optimal Solution

Although an optimal delta encoding scheme for a collection of files can be computed, such an approach has two problems. First, it needs a lot of computation for large N. Second, if the optimal encoding scheme forms a long chain, then in order to decompress the last page in the chain, all the other pages have to be decompressed. Hence, the optimal ordering is not always practical, especially for reducing client latency, as any savings obtained by compression could be offset by the required decompression time.

A reasonable sub-optimal approach is then to choose an anchor file in the given collection of files, and use it as the source page for compressing all the other files in that collection.
This guarantees that only one file is needed to decompress any other file. Choosing an anchor page is simple: Given matrix $W$ as described in last section, the anchor page $i$ satisfy the following condition: page $i$ is the anchor page if

$$S_{i0} + \sum_{j=1}^{n} S_{ij} = \max(S_{k0} + \sum_{j=1}^{n} S_{kj}) \text{ for all } k = 0, 1, ..., N$$

(2)

This means we choose the file $i$ that maximizes savings when it is compressed by itself and all the other files are compressed based on it.

### 3.3 Heuristic Solution

Both solutions above need to compute an NxN matrix at the encode side. Although in many applications encoding can be done offline, the time and space consumed can still be prohibitively excessive when the number of pages is large. A simple way to reduce the computation costs involved is to cluster the given collection of files into smaller groups. For web pages, there are currently, many clustering methods that are available. Instead, we have found that in practice a particularly simple approach works quite well. Instead of performing an explicit clustering, we simply utilize the directory structure used for storing web pages to define clusters. That is, we cluster the website according to its directory tree structure and only do delta compression within the same directory. Still the problem exists: what if there are too many pages in one directory? From our observation and experiments, it appears that most of the time such pages have very similar structure. Hence in such cases, we randomly choose a small number of pages, get an optimal anchor for this set by using the method in the last section and use the result as the anchor for the whole directory. As conclusion, when a directory has less than 5 pages, we just use the optimal ordering for them, otherwise use the sampling anchor page???

### 4 Experimental Results

In order to demonstrate proof of concept for the proposed algorithms we conducted a set of experiments. We investigated the four types of situations listed in Section 1 that could potentially benefit from delta compression. In the rest of this section we describe our findings categorized by these four types. Our experiments seem to indicate that not all kinds of related web pages have the potential for delta compression.

#### 4.1 Compressing pages with different versions

One category of web pages that get updated frequently but at the same time preserve their basic structure are news sites like cnn.com and abcnews.com etc. To test the potential of delta compression on such a set, we collected web pages from various news site for 100 hours, starting from 3pm ET on Sept. 9, 2000 and ending at 7pm ET on Sept. 13, 2000. Here we just show results from two representative sites, abcnews.go.com and www.cnnfn.com. Results from other sites were similar.
Table 1 shows the average compression ratio obtained by different algorithms. By compression ratio we mean the original file size divided by the file size after compression. Hence, the larger the value, the better the algorithm. The tabulated results show that for the web pages that have the same URL but represent versions at different points in time, the benefit of using Delta compression method is very large. This result is of course not new. However, it should be noted that the proposed wdelta algorithm gives much better performance as compared to the previously applied techniques, vdelta and diff+gzip.

Figure 3 plots the compression ratios for every page, the curve shows that even for the same algorithm, the compression ratios have large difference, this is because of the similarity between different versions of pages are not the same. When the similarity is large, the compression ratio is large.

<table>
<thead>
<tr>
<th>Method</th>
<th>wdelta</th>
<th>vdelta</th>
<th>diff+gzip</th>
<th>gzip</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.cnn.com">www.cnn.com</a></td>
<td>116.54</td>
<td>98.30</td>
<td>29.16</td>
<td>5</td>
</tr>
<tr>
<td>average compression ratio</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>abcnews.go.com</td>
<td>201.3</td>
<td>183.6</td>
<td>39.8</td>
<td>4.7</td>
</tr>
<tr>
<td>average compression ratio</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.2 Compressing pages with similar structure

Often a user downloads a collection of pages while online in order to later peruse them offline. In this situation, the collection of pages being downloaded have sufficient common structure and are similar to each other in many ways. To examine the possible benefits of delta compression in such a scenario, we applied our algorithm to several websites. The limitations for our experiments were 1) Only static pages were downloaded, thus the total number of pages for every website was not large and 2) Only HTML files were compressed in these experiments. Image and multimedia files were ignored.

We classify the websites chosen for our experiments into three groups:

- **Commercial website.** This kind of website has more similarity for product’s descriptions and layout structure, so it has more potential for delta compression. www.marketplacenow.com, pages.ebay.com, linux.tucows.com (products parts)

- **News websites.** This kind of website also has a lot of similarity for the layout structure, so it also has much potential for delta compression. abcnews.go.com, www.usnews.com

- **Academic website** This kind of websites has some layout similarity, but they generally has more percentage of plain text than the formatting and designed by different persons. cis.poly.edu, www.usenix.org/publications

4.2.1 Optimal Solution

In this experiment, we compute the optimal delta encoding scheme for the whole website, and used wdelta for compression. Compare to it, we also apply optimal ordering with vdelta and tar+gzip. Actually, tar+gzip is not a pair by pair compression, ”tar” command combine
all the files together, that gives the later pages more opportunity for finding matches in not just the previous one page. The disadvantage of it is: in order to get one page, all the files always have to be decompressed together. While our method is a kind of pair by pair compression method, in order to get one page, only its parent pages needs to be decode, in the best case, only one page is needed to decode the current page, but the worst case is a chain page ordering that also needs to decode all the pages in order to the last node on the chain. Since the tar+gzip is a possible method to make a group of pages compressed, we apply it here to compare with our algorithm. Table 2 and figure 4 shows the compression ratio for different group of website.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>average size (bytes)</th>
<th>total page (bytes)</th>
<th>wsdelta optimal</th>
<th>vdelta optimal</th>
<th>tar+gzip</th>
</tr>
</thead>
<tbody>
<tr>
<td>pages.ebay.com</td>
<td>23K</td>
<td>131</td>
<td>9.70</td>
<td>7.49</td>
<td>8.96</td>
</tr>
<tr>
<td><a href="http://www.marketplacenow.com">www.marketplacenow.com</a></td>
<td>13.5K</td>
<td>23</td>
<td>43.3</td>
<td>40.2</td>
<td>7.91</td>
</tr>
<tr>
<td>linux.tucows.com</td>
<td>24K</td>
<td>148</td>
<td>20.43</td>
<td>13.86</td>
<td>17.05</td>
</tr>
<tr>
<td>abcnews.go.com</td>
<td>31.4K</td>
<td>122</td>
<td>8.54</td>
<td>7.5</td>
<td>6.4</td>
</tr>
<tr>
<td>usnews/news</td>
<td>15.7K</td>
<td>43</td>
<td>8.47</td>
<td>7.55</td>
<td>6.4</td>
</tr>
<tr>
<td>cis.poly.edu</td>
<td>5.95K</td>
<td>222</td>
<td>7.22</td>
<td>6.24</td>
<td>7.4</td>
</tr>
<tr>
<td>usenix.org/publications</td>
<td>6.57K</td>
<td>148</td>
<td>4.5</td>
<td>3.9</td>
<td>4.82</td>
</tr>
</tbody>
</table>

The result shows that wdelta+optimal-ordering performs best for both commercial and news websites, but a little worse than tar+gzip for academic web site, the mail reason is as described above that tar+gzip can utilize all the previous pages in the tar file, while pair-by-pair compression can only use one best reference page. We can see from table 1, that the average size of academic websites we choose are around 5-6K, for gzip algorithm, its windows can cover more than 6 pages in its match windows. For example, there is such a situation: the first part of a page is similar with one page, while its second part is similar with another page. The current page will get two long match from both pages instead of get a good match in just one best reference page. That is also the reason that tar+gzip better than velta+optimal-ordering for some cases. From this part, we can claim that for very website with similar pages, wdelta+ordering is a good method for archiving and compact the whole website. While for the pages that have small size, if the whole website is required, tar+gzip maybe is another choice.

4.2.2 Sub-optimal Solution

As mentioned in Section 3, the worst case for optimal page ordering is a long chain for which in order to decode the last page node on the chain, all the pages have to be decoded. This is not a large problem for archiving application, but for download file and decode it on the fly, it is a disadvantage. In this experiment, we shows that the sub-optimal solution is a good method that trade some compression ratio with much simple and fast decoding. For this part of experiment, we compare it with gzip instead of tar+gzip, because for anchor-page compression, all the pages are based on only one page(the anchor page is compressed by itself), so it is comparable with using gzip individually. Table 3 and Figure 5 shows the result.
Table 3. Comparison of Optimal Solution and Sub-Optimal Solution

<table>
<thead>
<tr>
<th>Data Set</th>
<th>average size (bytes)</th>
<th>total page (bytes)</th>
<th>wdelta optimal</th>
<th>wdelta anchor-page</th>
</tr>
</thead>
<tbody>
<tr>
<td>pages.ebay.com</td>
<td>23K</td>
<td>131</td>
<td>9.70</td>
<td>6.5</td>
</tr>
<tr>
<td><a href="http://www.marketplacenow.com">www.marketplacenow.com</a></td>
<td>13.5K</td>
<td>23</td>
<td>43.3</td>
<td>13.1</td>
</tr>
<tr>
<td>linux.tucows.com</td>
<td>24K</td>
<td>148</td>
<td>20.43</td>
<td>18.0</td>
</tr>
<tr>
<td>abcnews.go.com</td>
<td>31.4K</td>
<td>122</td>
<td>8.54</td>
<td>7.26</td>
</tr>
<tr>
<td>usnews/news</td>
<td>15.7K</td>
<td>43</td>
<td>8.47</td>
<td>5.28</td>
</tr>
<tr>
<td>cis.poly.edu</td>
<td>5.95K</td>
<td>222</td>
<td>7.22</td>
<td>5.67</td>
</tr>
<tr>
<td>usenix.org/publications</td>
<td>6.57K</td>
<td>148</td>
<td>4.5</td>
<td>3.64</td>
</tr>
</tbody>
</table>

Table 3 shows the compression ratio change for websites. Among them, the compression ratio of abcnews.go.com change least, that means the whole website has some similarity in their formatting. While the compression ratio of like linux.tucows.com changes largest, it shows the whole website don’t have a very consistent formatting. So we can say, anchor-page ordering is good for web page groups with similar formatting.

Figure 5 gives the comparison of wdelta+anchor-page with other methods. Both delta methods, wdelta and vdelta, performances better than gzip, their average compression ratios are 8.5 and 7.6, while average compression of gzip is 3.9. Wdelta still performances the best.

4.2.3 Heuristic Solution

For the same websites, we apply the heuristic solution to compress the website. Table 4 and Figure 6 shows the result from different aspects.

Table 4. Comparison of Heuristic Solution with Other Solutions

<table>
<thead>
<tr>
<th>Data Set</th>
<th>average size (bytes)</th>
<th>total page (bytes)</th>
<th>wdelta optimal</th>
<th>wdelta anchor-page</th>
<th>wdelta heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>pages.ebay.com</td>
<td>23K</td>
<td>131</td>
<td>9.70</td>
<td>6.5</td>
<td>6.9</td>
</tr>
<tr>
<td><a href="http://www.marketplacenow.com">www.marketplacenow.com</a></td>
<td>13.5K</td>
<td>23</td>
<td>43.3</td>
<td>13.1</td>
<td>15.2</td>
</tr>
<tr>
<td>linux.tucows.com</td>
<td>24K</td>
<td>148</td>
<td>20.43</td>
<td>18.0</td>
<td>12.0</td>
</tr>
<tr>
<td>abcnews.go.com</td>
<td>31.4K</td>
<td>122</td>
<td>8.54</td>
<td>7.26</td>
<td>5.21</td>
</tr>
<tr>
<td>usnews/news</td>
<td>15.7K</td>
<td>43</td>
<td>8.47</td>
<td>5.28</td>
<td>5.63</td>
</tr>
<tr>
<td>cis.poly.edu</td>
<td>5.95K</td>
<td>222</td>
<td>7.22</td>
<td>5.67</td>
<td>6.25</td>
</tr>
<tr>
<td>usenix.org/publications</td>
<td>6.57K</td>
<td>148</td>
<td>4.5</td>
<td>3.64</td>
<td>3.34</td>
</tr>
</tbody>
</table>

Table 4 shows the comparison between three kinds of wdelta solutions. It is noticed that heuristic solution sometimes performances even better than sub-optimal solution. From analysing the data sets and the detail algorithm, we know this is reasonable because when the whole website has similar structure among most of its pages, an anchor page will give most of them benefit for delta compression, like abcnews.go.com, for this situation, suboptimal method is better than heuristic method, but if the whole web site has several groups of different structures, only choosing one anchor page will be worse than choosing an anchor page for every different groups (directories in the URL trees).
Figure 6 shows that both wdelta and vdelta performance better than gzip even using the simplified heuristic solution, this further confirms that the delta compression has potential for compressing similar structured pages.

In summary, both sub-optimal solution and heuristic solution have the potential for reducing the latency for downloading and storage.

5 Related work

How to decrease the storage space and provide faster web surf is an important research issue. A lot of researchers have worked on this issue, their work can be classified into four main methods. **Delta compression:** From 1997, J.Mogul, F.Douglis and G.Banga etc have done some research on data compression for HTTP[2] and delta compression for www latency reduction[9], Jacobson[10] and M.Degermark[11],etc both uses the similarity of TCP/IP header for delta compression to reduce the narrow-band transmission. **Caching:** A similar cache mechanism like in the operating system is also used to cache the newest several pages just read so that to reduce the latency when those are required again. Research related to this are[12][13]. Also some caching algorithm also using the delta compression idea in their algorithm[14]. **Prefetching:** This method has the potential for reducing the download latency if using the history trace as the prediction. The research of L.Fan and P.Cao[1], etc on this topic shows that prefetching is a good method in this application but it has some limitation for the accuracy of the prediction of the prefetching pages. **HTML compression:** Better algorithms need to be invented that compress the data stream more efficiently for HTML[15]. Till now, the HTML compression is mainly on the level of HTML optimizer and simply using gzip to compression.

6 Conclusion and Future Work

In this paper we have shown that delta compression can be used for compressing collections of web pages which have similar content or structure. We proposed a new kind of delta compression method, named wdelta which is competitive with current delta compression methods diff+gzip and vdelta. We then used this delta compression method for compressing different kinds of related web pages. Our study shows the web pages with the same URL but different versions have very large compression potential. The compression ratio can even reach more than 100. For pages with similar structure, we propose three solutions for delta encoding: optimal solution, suboptimal solution and heuristic solution. Optimal solution gives a lower bound for this kind of pair by pair delta compression, and it has the largest compression which is suitable for web page archiving. Supoptimal solution and heuristic solution have less compression ratio but the decoding time is much faster and can be decode on the fly. This makes them suitable for downloading a collection of related web pages for offline browsing. Our experiments shows that other kinds of related pages, like similar topic but different structure or pages with links to each other have little benefit for delta compression.

There are several limitations in our study: First, for both kind of methods, we assume users are interested in most of the pages in that related page group. The assumption is
reasonable if that is a page group clustered by subject. Second, we have not considered other methods for reducing the latency like HTML compression and prefetching. Surely, a more efficient HTML compression technique can lead to a more specialised delta compression technique designed for HTML files. Finally, we did not include all kind of web pages in our study, like dynamic web pages.

Much future work remains. We will apply our method to conduct experiments on a large scale involving millions of web pages. Applications. We also plan to combine other methods like HTML compression and prefetching with delta compression. A full proxy-based implementation for real transmission is also on the way.

7 References


7. Timothy C. Bell, Ian H. Witten, John Cleary, Text Compression, Prentice Hall.


