Chapter 3

MODELS FOR CRAWLER REVISIT FREQUENCY

3.1 INTRODUCTION

With the exponential growth of the web, the contents of the web pages change very frequently [43, 46]. A periodic crawler [20] visits the websites, old as well as new, after a certain period. The old sites are visited for the purpose of maintaining freshness where it downloads the new version of the document. The new web sites are visited to download the new pages and replace the old versions. Nevertheless, the collections are updated in batch mode. Whereas an incremental crawler [20] crawls the web pages comparatively more frequently with a view to refresh existing collection and replaces only those pages which have undergone up-dation thereby providing fresh information to the user. When pages change at a very fast rate, the crawler needs to revisit the sites as frequently as possible leading to more network resource consumption and putting tremendous load at the target site as well. Revisit frequency is an important factor towards designing an incremental crawler.

From chapter 2, it is evident that almost all studies have focused on change frequency of web page while considering revisit policies to refresh a web page [20]. However, all pages do not change at the same interval of time and therefore they can not be refreshed at same frequency. This drawback arises because change in documents takes place on account of various reasons based on which the documents can be categorized as follows [48, 49].

- Static web pages do not change over a long period of time.
- Web page change only when website administrator updates or modifies its website.
- Dynamically generated web pages.
- Very frequently updated parts of web pages e.g. News website, Share market’s website.
Based on the above said categories a self adjusting refresh time model for incremental crawler has been developed. The detailed discussion for self adjusting refresh time for incremental crawler is given in the next section.

3.2 SELF ADJUSTING REVISIT INTERVAL FOR INCREMENTAL CRAWLER

In this work an incremental crawler has been developed that takes care of the above given diversity of web pages as far as change in contents of document is concerned.

If it can be estimated how often pages change, the incremental crawler may revisit only the pages that have changed (with high probability), instead of refreshing the entire collection altogether as done by the periodic crawler. Therefore, the incremental crawler is more effective.

In view of this, it is proposed that incremental crawler may visit a site frequently and after every visit its revisit interval of future visits may be adjusted according to the category of the site and the page change probability [44, 88, 91] i.e. the sites/pages which contains volatile information should be revisited more often in comparison to those pages which contains static information.

A novel method for dynamically updating the revisit interval has been developed. Wherein, the “self adjusted revisit interval” for downloading the documents from a site is computed by the following formula:

$$t_{n+1} = t_n + \Delta t$$  \hspace{1cm} (3.1)

where $t_n$ : is current revisit interval for any site.

$t_{n+1}$ : is adjusted revisit interval.

$\Delta t$ : is dynamic change in revisit interval.

In equation (3.1) the value of $\Delta t$ may be positive or negative, based upon the degree of success ($p_c$) that the site contains the volatile documents. The degree of success is
computed in terms of number of hits. Where, if a visited site is found to contain documents that have changed then the visit is called a hit. For example, if the crawler encounters a document being updated 6 times out of its 10 visits, the degree of success \( p_c \) is assigned as 0.6 to that site. A fingerprinting scheme such as suggested by Sharma et. al. in reference [15] can be suitably applied to determine whether a particular document has changed or not.

A unit step function \( u(x) \) has been employed for the computation of \( \Delta t \), which is defined in equation (3.2):

\[
\Delta t = \{(1-p_c/p_g)*u(p_c-p_g) + (1-p_c/p_l)*u(p_l-p_c)\} * t_n \tag{3.2}
\]

where: \( p_g, p_l \) are the boundary conditions i.e. upper and lower threshold values of \( p_c \) respectively,

and \( u(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \)

The proposed mechanism has been implemented on the three different sites belonging to different categories. The following examples show the dynamic computation of revisit interval for observed data.

### 3.2.1 EXAMPLES

The mechanism has been applied for retrieving the pages from the site of a popular news paper (http://timesofindia.indiatimes.com/). It is observed that the pages are updated with a probability of 0.9. That is out of 10 crawler's visits documents are found updated 9 times.

The following initial revisit interval and boundary conditions are taken:

\[ t_n = 100 \text{ hours} \]
\[ p_l = 0.3 \]
\[ p_g = 0.7 \]
and \[ p_c = 0.9 \] because documents are found to updated 9 times out of 10 visits.

By putting the data in to equation (3.2) the \( \Delta t \) is computed as follows:

\[ \Delta t = \{(1-0.9/0.7)*1+ (1-0.9/0.3)*0\} * 100 = 28.57 \]

The new refresh time or revisit interval as per the equation (3.1) is computed as follows:

\[ t_{n+1} = 100 - 21.42 = 71.42 \text{ hours} \]

**Revisit interval for the site is decreased**

In the next case, the mechanism has been applied for retrieving the pages from the site of Haryana Urban Development Authority (http://huda.nic.in/). It is observed that the pages are updated with a probability of 0.2. That is out of 10 crawler’s visits documents are found updated only 2 times.

The following initial revisit interval and boundary conditions are taken:

\[ t_n = 100 \text{ hours} \]

\[ p_i = 0.3 \]

\[ p_g = 0.7 \]

and

\[ p_c = 0.2 \] likewise.

By putting the data in to equation (3.2) the \( \Delta t \) is computed as follows

\[ \Delta t = \{(1-0.2/0.7)*0 + (1-0.2/0.3)*1\} * 100 = 33.33 \]

The new refresh time or revisit interval as per the equation (3.1) is computed as follows:

\[ t_{n+1} = 100+33.33 = 133.33 \text{ hours} \]

**Revisit interval for the site is increased.**

In the next case, the mechanism has been applied for retrieving the pages from the site of YMCA University (http://www.ymcaie.ac.in/). It is observed that the pages are updating with a probability of 0.6. That is out of 10 crawler’s visits documents are found updated 6 times.
The following initial revisit interval and boundary conditions are taken:

\[
\begin{align*}
    t_n &= 100 \text{ hours} \\
    p_t &= 0.3 \\
    p_g &= 0.7
\end{align*}
\]

And

\[
    p_c = 0.6
\]

likewise.

By putting the data in to equation (3.2) the \( \Delta t \) is computed as follows:

\[
    \Delta t = \{(1-0.6/0.7)*0 + (1-0.6/0.3)*0\} * 100 = 0
\]

The new refresh time or revisit interval as per the equation (3.1) is computed as follows:

\[
    t_{n+1} = 100 + 0 = \textbf{100 hours}
\]

No Change in revisit interval

The observation indicates that the site of university is changing on an average of once per week so current revisit interval of 100 is suitable for the site and remained same.

From the above examples it may be observed that the value of \( p_c \) being greater than \( p_g \) results in decrease in revisit interval i.e. increase in revisit frequency because interval is inversely proportional to frequency. Similarly values of \( p_c \) being less than \( p_t \), results in increase in revisit interval i.e. decrease in revisit frequency. Thus the revisit frequency of incremental crawler gets updated dynamically. The relation between revisit interval and page up-dation probability is shown in Fig. 3.1.

![Fig. 3.1: Change in Revisit Interval vs Page Up-dation Probability](image-url)
The values of $p_l$ and $p_e$ can be chosen manually as per the category of a site. This mechanism ensures that the time interval between two successive visits to a site is decreased with increase in “page up-dation” probability and vise-versa.

### 3.3 A NOVEL MODEL FOR MIGRATING CRAWLER REVISIT FREQUENCY

The migrating crawler sends migrants to the different sites/servers for downloading the documents. The frequency for resending the migrants for the purpose of downloading the documents is important in the sense that it ensures the freshness of the collection at the cost of consumption of network resources. As discussed earlier in this chapter, the sites can be categorized based on their change frequency of pages. Hence based on the categorization of the site/server the frequency of revisit needs to be optimized in such a way that it maintains not only the freshness of collection but consume less network resource as well. It is proposed that migrating crawler should adopt dynamic frequency for revisiting a particular site/server based on the category of the documents housed there. A mathematical model for migrating crawler revisit frequency has been developed, the details of which are given below.

If a dynamic frequency of revisit is used then it is necessary to find at which rate crawler needs to revisit each site/server. The rate of revisit should increase linearly for a site/server that contains more dynamic information i.e. with the high change frequency of pages housed on that site. However, this conclusion may not be right. For example, if a crawler revisits a site every minute whose rate of change is once every minute then naturally downloaded pages will not reflect the updated information even after employing such a high refresh rate. The situation worsens even more for the pages belonging to a highly dynamic site such as share market’s site. So it is felt that optimal revisit frequency is not always proportional to the change frequency of a page [50, 89].

In the light of the above discussion, the mechanism for computing the revising rate of a page needs to be modified in such a way that the revisiting frequency be proportional to the page change frequency up to a certain threshold value. Thereafter it should remain
constant up to next threshold and if page change frequency exceeds this second threshold value then it is worthless to fetch the page with earlier frequency and revisit frequency be decreased. The observed behavior of revisit frequency is shown in Fig 3.2.

![Graph showing change frequency of page versus revisit frequency.](image)

**Fig. 3.2: Change Frequency of Page vs Revisit Frequency**

The frequency of revisit is computed by the formula given in equation (3.3):

\[ f_{n+1} = f_n + \Delta f \]  

(3.3)

where \( f_n \) : is current revisit frequency.

\( \Delta f \) : is dynamic change in frequency.

\( f_{n+1} \) : is adjusted revisit frequency.

The unit step function \( u(x) \) is used for the computation of \( \Delta f \) where:

\[
\begin{aligned}
\text{and } u(x) &= \\
&= \begin{cases} 
1 & \text{if } x \geq 0 \\
0 & \text{otherwise}
\end{cases}
\end{aligned}
\]

\( \Delta f \) is computed as per the formula given in equation (3.4).

\[
\Delta f = \left[ f_n \times (\lambda_{\text{current}}/\lambda_{\text{previous}} - 1) \times u(\lambda_{\text{current}} - \lambda_{\text{lower}}) \times u(\lambda_{\text{middle}} - \lambda_{\text{current}}) \times u(\lambda_{\text{upper}} - \lambda_{\text{current}}) \right] + \left[ f_n \times (1 - \lambda_{\text{current}}/\lambda_{\text{upper}}) \times u(\lambda_{\text{current}} - \lambda_{\text{upper}}) \times u(1 - \lambda_{\text{current}}) \right].
\]  

(3.4)
where:

\( f_n \) : is current revisit frequency.
\( \lambda_{\text{current}} \) : is current change frequency of page.
\( \lambda_{\text{previous}} \) : is change frequency of page at the previous visit.
\( \lambda_{\text{lower}} \) : is lower threshold value.
\( \lambda_{\text{upper}} \) : is upper threshold value.
\( \lambda_{\text{middle}} \) : is middle threshold value.

Depending on the different conditions, the value of \( \Delta f \) is shown in Table 3.1.

<table>
<thead>
<tr>
<th>Condition</th>
<th>( \Delta f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda_{\text{lower}} &lt; \lambda_{\text{current}} &gt; \lambda_{\text{middle}} )</td>
<td>+ve</td>
</tr>
<tr>
<td>( \lambda_{\text{middle}} &lt; \lambda_{\text{current}} &gt; \lambda_{\text{upper}} )</td>
<td>0</td>
</tr>
<tr>
<td>( \lambda_{\text{upper}} &lt; \lambda_{\text{current}} )</td>
<td>-ve</td>
</tr>
</tbody>
</table>

The model ensures that frequency of revisit \( (f) \) which is initialized by \( f_1 \) will increase with the change frequency of pages up to the middle threshold value \( (\lambda_{\text{middle}}) \) and it remains constant up to upper threshold value \( (\lambda_{\text{upper}}) \) i.e. unaffected by the change frequency of pages. However, beyond upper threshold value \( (\lambda_{\text{upper}}) \) it starts reducing automatically and settles itself to lower threshold.

**3.3.1 EXAMPLES**

In the following examples the revisit frequency has been computed for successive cases by assuming the following values.

\( \lambda_{\text{lower}} = 0.1 \),
\( \lambda_{\text{middle}} = 0.6 \) and
\( \lambda_{\text{upper}} = 0.8 \).
Case 1: Now consider the data given below:
Initial frequency of revisit \( f_i = 10 \) times/1000 hour.
Current change frequency of page \( \lambda_{\text{current}} = 0.15 \) and \( \lambda_{\text{previous}} = \lambda_{\text{lower}} = 0.1 \)

Using equation (3.4), \( \Delta f \) is computed as follows:
\[
\Delta f = \left[ \{ 10 \times (0.15 / 0.1 - 1) \times u (0.15 - 0.1) \times u (0.6 - 0.15) \times u (0.8 - 0.15) \} + \{ 10 \times (1 - 0.15 / 0.1) \times u (0.15 - 0.8) \times (1 - 0.15) \} \right].
\]
\[
= 5
\]

By using eq (3.3), the new revisit frequency can be computed as follows:
\[
f_{n+1} = 10 + 5 = 15 \text{ times/1000 hour}
\]

Revisit frequency is increased in proportion to change frequency of pages.

Case 2: Now taking the current revisit frequency from previous case i.e. \( f_n = 15 \) and considering \( \lambda_{\text{current}} = 0.45 \) and \( \lambda_{\text{previous}} = 0.15 \)

Using eq (3.4), \( \Delta f \) would be computed as given below:
\[
\Delta f = \left[ \{ 15 \times (0.45 / 0.15 - 1) \times u (0.45 - 0.1) \times u (0.6 - 0.45) \times u (0.8 - 0.45) \} + \{ 15 \times (1 - 0.45 / 0.15) \times u (0.15 - 0.8) \times (1 - 0.45) \} \right].
\]
\[
= 30
\]

By using eq (3.3), the new revisit frequency can be computed as follows:
\[
f_{n+1} = 15 + 30 = 45 \text{ times/1000 hour}
\]

Revisit frequency is increased in proportion to change frequency of pages.

Case 3: Now taking the current revisit frequency from previous case i.e. \( f_n = 45 \) and considering the current change frequency of page \( \lambda_{\text{current}} \) anywhere between \( \lambda_{\text{middle}} \) and \( \lambda_{\text{upper}} \) say = 0.7 and \( \lambda_{\text{previous}} = 0.45 \)

Using eq (3.4), \( \Delta f \) would be computed as given below:

\[\text{...}\]
\[ \Delta f = \{45 \times (0.7 / 0.45 - 1) \times u (0.7 - 0.1) \times u (0.6 - 0.7) \times u (0.8 - 0.7)\} + \{45 \times (1 - 0.7 / \ 0.45) \times u (0.7 - 0.8) \times (1-0.7)\}]. \\
= 0 \\
By using eq^n (3.3), the new revisit frequency can be computed as follows:
\[ f_{n+1} = 45 + 0 = 45 \text{ times/1000 hour} \]
Revisit frequency is not changed i.e. remains the same as previous revisit frequency.

Case 4: Now again taking the current revisit frequency from previous case i.e. \( f_n = 45 \) and considering the current change frequency \( (\lambda_{\text{current}}) \) of page greater than \( \lambda_{\text{upper}} \) Say = 0.9 and \( \lambda_{\text{previous}} = 0.7 \).

Using eq^n (3.4), \( \Delta f \) would be computed as given below:
\[ \Delta f = \{45 \times (0.9 / 0.7 - 1) \times u (0.9 - 0.1) \times u (0.6 - 0.9) \times u (0.8 - 0.9)\} + \{45 \times (1 - 0.9 / \ 0.7) \times u (0.9 - 0.8)\}]. \\
= -12.85 \approx -13. \\
By using eq^n (3.3), the new revisit frequency can be computed as follows:
\[ f_{n+1} = 45 - 13 = 32 \text{ times/1000 hour} \]
Revisit frequency is decreased proportional to change frequency of pages.

3.3.2 OBSERVATION

Based on above calculations a graph has been plotted between change frequencies of page vs revisit frequency of the pages (see Fig 3.3).

![Graph showing change frequency vs revisit frequency](image)

Fig. 3.3: Observed Change Frequency of Page vs Revisit Frequency
It may be noted from the Fig. 3.3 that observed parameters based on proposed mathematical formula reflects the identical results as estimated in Fig. 3.2.

### 3.4 MODIFIED URL STRUCTURE

For incorporating the above said optimal frequency of revisit, the structure of URL record has been modified. The modified structure of URL record is shown in Fig. 3.4.

<table>
<thead>
<tr>
<th>URL</th>
<th>IP</th>
<th>Status</th>
<th>( f )</th>
<th>LCT</th>
<th>( \lambda_{\text{lower}} )</th>
<th>( \lambda_{\text{middle}} )</th>
<th>( \lambda_{\text{upper}} )</th>
<th>Doc pointer</th>
<th>Fingerprint Key</th>
<th>Nearest Location</th>
</tr>
</thead>
</table>

Fig 3.4: The Modified Structure of URL Record

where:

- **URL** is Uniform Resource Locator of the document
- **IP** is Absolute IP corresponding to the URL.
- **Status** field represent whether URL is seen before or not i.e. if URL is encountered first time then its fields needs to be initialized by field initializer. If status! =1 then it a new URL, otherwise old.
- **\( f \)** is current revisit frequency to a particular website/web page. After initializing by some default value, this field is dynamically updated by revisit frequency calculator module.
- **LCT** is Last Crawl Time. It stores the date and time stamp in Universal Coordinated Time (UTC) of the last time the page was crawled.
- \( \lambda_{\text{lower}}, \lambda_{\text{middle}} \) and \( \lambda_{\text{upper}} \) are the lower, middle and upper threshold values defining the boundary conditions for change frequency of pages.
- **The document pointer** field contains the pointer to the original document.
- **Fingerprint Key** field stores the finger print key value of the crawled page used for duplicate pages detection.
- **Nearest Location** stores the address of the nearest location of organization owning IP address URL corresponding to the mirrored page. It is initially set to null by field initializer.
A detailed architecture of scalable parallel migrating crawler based on augmented hypertext documents is being proposed in the next chapter that not only addresses the problems prevailing in the recent web crawlers but also enhances the performance of parallel crawling in particular.