A Modified Algorithm to Handle Dangling Pages using Hypothetical Node

Shipra Srivastava
Student
Department of Computer Science & Engineering
Thapar University, Patiala, 147001 (India)

Rinkle Rani Aggrawal
Assisstant Professor
Department of Computer Science & Engineering
Thapar University, Patiala, 147001 (India)

ABSTRACT
The information on the web is intensifying day by day due to which there is a bulky amount of information available on the web. With this large amount of information available on the web, the process of searching has become a complex task. To overcome this problem there is a necessity of some efficient search engine. With such a search engine information retrieval becomes simple. A PageRank algorithm is used to serve this process. A PageRank algorithm places the most important pages at high ranks depending upon the number of in-links linking to that page. Google search engine developed the Page Rank technique to rank pages and had become the most effective search engine. Page Rank arranges the order of the user's search result based on the relevance of the search of the information In this paper problem of the Dangling pages that is associated with page rank algorithm is been discussed. Dangling pages produce the inaccurate page rank of non-dangling pages. These pages also produce the philosophical issues and computational issue. In previous study for solving the philosophical issue with dangling node a hypothetical node is added to the web graph. But this hypothetical node increases the computation issue because the number of iteration to converge the algorithm is increased. In this paper a' modified algorithm to handle the dangling node with hypothetical node without increasing computational issue which is raise due to addition of hypothetical node is discussed.

General Terms
PageRank algorithm, Dangling Pages, Hypothetical Node

Keywords
Page Rank, Dangling Pages, Convergence, Stochastic matrix, Markov chain

1. INTRODUCTION
Page Rank is Google's method of measuring a page's importance PageRank algorithm can be defined as a link based probabilistic algorithm. This algorithm assigns a numerical value to each page in the World Wide Web so that the relevant importance of each page rank is measured with other pages on the web. PageRank algorithm is developed by Google founders Surgey Brin and Larry Page [1] and they use PageRank algorithm as a foundation for their search engine. Pages that are deemed more "important" will move up in the result of the pages of a user's search when a user enters their query in the search box. PageRank algorithm finds the pages on the Web that matches the user's query and lists those pages in the order of their Page Rank. When people calculate the page rank of any page we take into account the number of back links to that page and also the number of out-links of each page that point to it. In the web there is also some pages that has no out-links, these pages are called dangling pages or Zero-out link pages. These dangling pages can produce philosophical, storage and computational issues for the search engine. Pages that are protected by robots.txt, are dangling pages, pages not having genuine out link like the pdf files, images etc. are also comes under the dangling pages. Dangling pages also include URLs with meta tag indicating that links should not be followed from the page, and pages that return a 500 class response at crawl time due to configuration problems, network problems and bad link problems [2]. In the random surfer model, user move from one page to other page by randomly choosing an outgoing link from one page. But sometimes this random choice lead to dead end, meaning, to a page that has no outgoing links. This theoretical random walk is known as a Markov chain or Markov process [3]. According to N.Eiron et al. [2] the number of dangling pages is growing rapidly so it cannot be remove from the web and also sometimes these dangling pages contain important and useful information like a pdf file which may contain some good quality information regarding any topic so deleting it from the web is not the solution from overcoming the difficulty of dangling pages.

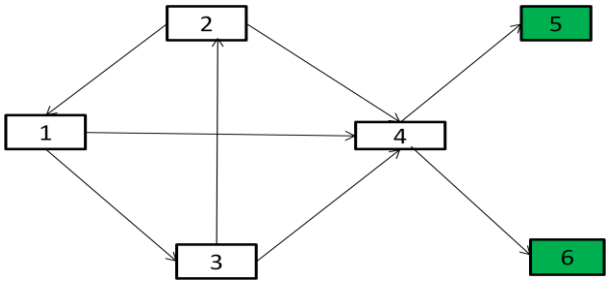

Figure 1: A directed Web Graph with 6 Web Pages

In the above diagram node 5 and 6 (green node) are dangling nodes because they have 0 out link, from node 5 and node 6 user cannot go on any other pages on the web.

This paper is organized as follow: In section 2 Issue related to dangling nodes like philosophical computational issues are discussed and the entire problem which is raised because of dangling node removal is also described. In section 3 previous work to handle dangling pages by hypothetical node is discussed. In section 4 the modified algorithm to handle dangling page with hypothetical node without increasing computational issue is discussed. Section 5 contains the experimental result with comparison to previous work.

1.1 Page Rank Calculation

The original page rank algorithm [1] is described by the formula:

$$pr(A) = (1-d) + d \sum_{s \in I_A} \frac{pr(s)}{\deg(s)} \quad (1)$$

Where
PR (A) is page rank of page A (whose page rank is need to find out).
d=damping factor and normally the value of d is set to 0.85.
(1-d) is a normalization coefficient.
I_A = the number of in-link of page A.
Pr (s) = page rank of page s which provide in-link to page A.
deg (s) = out-degree of page s.

Random surfer model is model behind the PageRank algorithm and it uses the web graph to model the behavior of random web surfer [4]. In web graph each page is denoted by the node and link from the one page to another page is denoted by the edge between these two pages. The Markov model represents the web directed graph as a transition probability matrix P. With Markov chains theory steady state vector (Page Rank) vector can be determined which probability vector is

$$\sim\pi = \sim\pi M$$

Where, $\sim\pi$ is the steady state vector.
M is transition probability matrix.
Page Rank vector can be determined by solving this matrix equation. $\sim\pi$ is the principal left eigenvector for M.
In the transition probability matrix M [5] the elements of a matrix M as

$$Mij = \begin{cases} \frac{1}{|Oi|} & \text{if there exists a link from page i to page j} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Where, Oi is out-link degree of node i. For computing the page rank vector the matrix need to be stochastic because markov chain is only defined for stochastic matrix.

1.2 Dangling Page

Dangling pages are pages which do not have any out link or the page which not provide reference to other pages. These Dangling pages create many issues like philosophical, computational and storage issue. Dangling pages are also called Hanging pages [6]. When the existing PageRank algorithm was developed it is easy to crawl the entire web because of the web static nature. The static nature of the web not allow many changes in the link structure of the web but now the web is evolving into a dynamic one driven by databases. These Dangling pages also create Link rot problem because of the dynamic behavior of web [2].The link rot is kind of problem in which the links which was working at one time is not working now a days and reason behind is not working now is the content are removed from that link or URL of that link is changed or the links are broken and sometimes it will return HTTP code 403 or 404 [7]. Penalty pages are the pages which return this HTTP code. There are several algorithms "push-back", "self-loop", "jump-weighting" and BHITS" proposed by N. Eiron [2] to adjust the ranks of pages with links to penalty pages. The BHITS algorithm resembles the HITS algorithm [8] is also used for handling dangling pages. The numbers of Dangling pages are increased day by day because of the evolving nature of the web. Since the number of Dangling node are growing so rapidly we need to find out some solution for this [9]. Removal of dangling pages from the web graph is not the solution for handling dangling web pages because removal of these pages also create new problem.

2. ISSUES RELATED TO DANGLING PAGE

In order to compute the page rank vector the transition probability matrix need to be stochastic but when there is dangling node in the graph then the row corresponding to dangling node contains all zero. O^T row of the matrix makes the matrix not stochastic because for the matrix to be stochastic sum of the row must be 1.For example consider the web graph in Figure 1:

The transition probability matrix of web graph in figure 1 according to equation 2 is

	1	2	3	4	5	6
1	0	0	1/2	1/2	0	0
2	1/2	0	0	1/2	0	0
3	0	1/2	0	1/2	0	0
4	0	0	0	0	1/2	1/2
5	0	0	0	0	0	0
6	0	0	0	0	0	0

Figure 2: Transition Probability Matrix of Web Graph

In the above transition probability matrix M row corresponding to dangling node 5 and 6 contain 0^T row so the above matrix is not stochastic. For overcoming from this problem Surgey Brin and Larry Page suggest that replace the each 0^T row of the transition probability matrix with a dense vector. The main solution for this problem is use a uniform vector e^T/n (e is the vector of all ones) [10] but with this solution problem of the storage requirement increase considerably.

There are also two main issues which related to dangling pages first philosophical and second is computational issue. The philosophical issue occurs due to excluding the dangling node from the computation because Surgey Brin and Larry Page remove the entire dangling node from the graph during the computation in the original PageRank algorithm [11]. After that when the page rank of non-Dangling nodes is calculated the Dangling pages are re-inserted back to graph without disturbing the result of the non-Dangling pages. Some dangling nodes should receive high page rank. For example, a very authoritative pdf file could have many in-links from respected sources and thus should receive a high page rank. Simply removing the dangling nodes biases the page rank vector unjustly and additionally it also create more other problem [12]. But removing the Dangling node from the graph is not the solution because sometimes removal of dangling node creates more Dangling node. The removal of Dangling node also produce the inaccurate page rank of non-Dangling node because removal of Dangling pages also effect the out-degree of non-Dangling pages which has a link to these Dangling pages. Here is the example which explains the above problem associated with removal of Dangling pages.

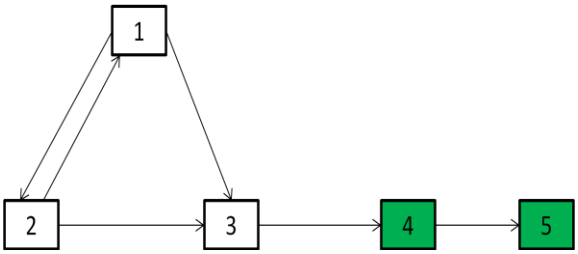

Figure 3: Directed Web Graph with 5 Web Pages

In the above Figure 3 page 5 (green node) is Dangling node if Dangling page 5 is removed from the graph it create new Dangling node which is node 4 and again the removal of web page 4 also create new dangling node which is node 3 so the removal of dangling node becomes iterative process until all the Dangling nodes are remove from the graph. So removing the dangling node sometimes makes loss of great amount of useful data from the web.

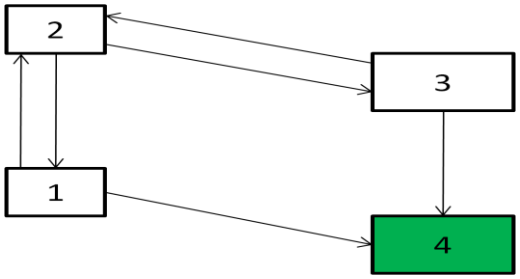

Figure 4: Directed Web Graph with one Dangling Page (Green node)

In the above Figure 4 if Dangling page 4 is removed from the graph then it also effect the out-degree of page 1 and page 3 and in this way removal of dangling node produce inaccurate value of non-Dangling nodes because in the page rank calculation the out degree of each page which provide in-links to computing page is also considered.

Computational issue is also associated with the dangling nodes because the dangling nodes are excluded from most of the computations the operation count depends to a large extent only on the number of non-dangling nodes. The efficiency of the algorithm increases as the number of dangling nodes increases [12]. But in the page rank computation the dangling node also effect the page rank value of non-dangling node as it only absorb the value, the dangling node never distribute their page rank value to other node because they do not have any out link to other page. So dangling node increase the computation issue. As the removal of dangling node also affect the out degree of non-dangling page so indirectly removal of dangling pages affect the page rank value of non-dangling pages. There is need to handle or decrease these philosophical and computational issue associated with dangling pages.

3. PREVIOUS WORK TO HANDLE DANGLING PAGES

There are many previous works done to handle dangling pages. According to [7] they handle the dangling pages by connect a hypothetical node to the web graph and connect the entire dangling node in the web graph to hypothetical node and construct the Self loop from the hypothetical node which point back to itself. By creating this hypothetical node in the graph make the matrix become stochastic which is necessary for computing the page rank vector because Markov chain is only defined for stochastic matrix. Because now the entire dangling nodes in the graph have at least 1 out degree and target of all these out-link is hypothetical node. The hypothetical node also has 1 out degree because of self loop. From this solution philosophical issue is also solved because now the dangling node is not excluded from the graph before the calculation. Page rank of dangling node is calculated along with the non-dangling node. Here figure 5 graphically shows that how the hypothetical node solves the philosophical issue by converting the dangling node into non-dangling node so that these dangling nodes are included in the graph. Below Figure 5 is result of applying the approach defined above on Figure 1.

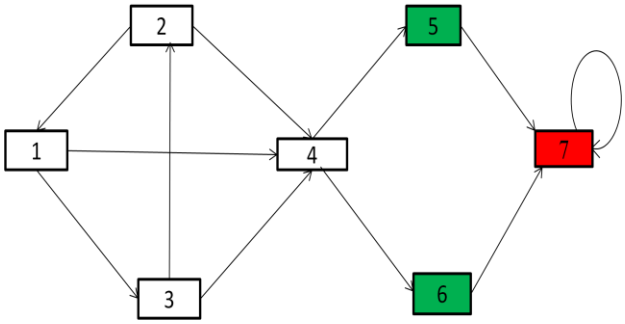

Figure 5: A Directed Web Graph with Hypothetical Node (Red node) 7

In the Figure 1 the node 5 and 6 are dangling node but after adding hypothetical node (Red node) to figure 1 and connect the dangling node (green nodes) to hypothetical node 7 and also construct a self loop on the node 7 then it converted into Figure 5. Now in the web graph there is not any node that has the 0 out-link. After applying the Equation 2 on Figure 5 the transition probability matrix of web graph in Figure 5 is

	1	2	3	4	5	6	7
1	0	0	1/2	1/2	0	0	0
2	1/2	0	0	1/2	0	0	0
3	0	1/2	0	1/2	0	0	0
4	0	0	0	0	1/2	1/2	0
5	0	0	0	0	0	0	1
6	0	0	0	0	0	0	1
7	0	0	0	0	0	0	1

Figure 6: Transition Probability Matrix of Web Graph

The matrix is now stochastic no row in the matrix contain all O. We can now apply Markov chain to compute the page rank vector. By implementing this method page rank value of all the nodes non-dangling, dangling or hypothetical are come out as a result. The value of hypothetical node is always having a high Page Rank and It can be ignored according to [7]. By creating hypothetical node approach philosophical issue is solved by including the entire dangling node but computational issue increased from this approach. For example if web graph in figure 7 is run on the original Page rank algorithm then it takes 33 iteration to converge the algorithm because original page rank algorithm remove the dangling node from the graph.

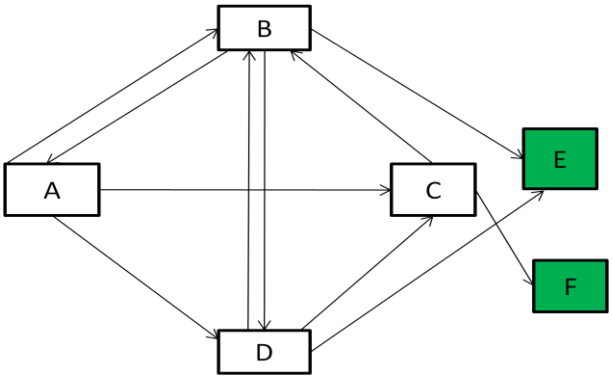

Figure 7: Directed Web Graph with 6 Web Pages

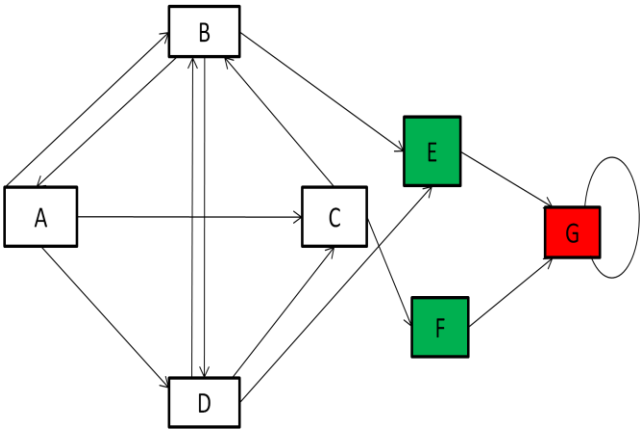

Figure 8: Directed Web Graph of 7 Web Pages with Hypothetical node G

But when the hypothetical node approach discussed above is applied on the figure 7 then it is modified into the figure 8.

The PageRank algorithm is run on figure 8 then it takes 135 iterations to converge the algorithm. The result of page rank value of every node of figure 8 is shown in figure 9.From the figure 9 it has been observed that the number of iteration is increased from 33 to 135. From the figure 9 it observed that in iteration 38th the value of all other node is converge but the value of hypothetical is going to converged on iteration 135. Actually, the algorithm is converged at the end of 38th iteration but because of the hypothetical node the iterations went up to 135 iterations. hypothetical node the iterations went up to 135 from the 38 iteration. From the figure 9 it has been see that at the 38th iteration value if all other node is same as in the previous iteration.

But the hypothetical node value (the value in yellow box) is still changed from the 38th iteration to 135th iteration. At last at the 135th iteration value of hypothetical node don't change. So because of the addition of But now the page rank value is more accurate than the previous value in which the dangling node is excluded from the computation. The only problem is that the number of iteration is increased. So by this method the philosophical issue is solved by including all the dangling pages in the computation but the computational problem is increased because of increment in the number of computation. So now in this paper we modified this approach by which we solve the computational issue which is raised due to the addition of hypothetical node.

1: 0.1499999765	0.1499999765	0.1499999765	0.1499999765	0.1499999765	0.1499999765	0.1499999765
2: 0.1924999702	0.2987499563	0.2349999646	0.2349999646	0.2349999646	0.2137499681	0.5324999270
.....						
.....						
37: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8404751923
38: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8430437519
39: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8452270279
.....						.....
.....						.....
133: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8575989235
134: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8575989242
135: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8575989242

Figure 9: Result of Applying the Existing algorithm on Web Graph shown in Figure 8

4. PROPOSED MODIFIED ALGORITHM

For solving the issues related to dangling node with the concept of hypothetical node without increasing the computational problem a modified algorithm is proposed. The computational problem which is generated due to addition of hypothetical node is solved by not comparing the value of hypothetical node to become unchangeable in the iterations for the algorithm to converge. The algorithm is going to be converged when value of all other node except the hypothetical node is same in both (i-1)th iteration and ith iteration. At that point accurate value of dangling node is obtained because when value of all non-dangling node is set or become unchangeable then value of dangling node at that point also become unchangeable because the inlinks to these

dangling pages are always from these non-dangling pages. According to the formula described in equation 1 if there is no change in the page rank value of all the pages which provide inlink to page for which the page rank value is calculated than the page rank value of computing page is always same untill the page rank value of any inlink page is not changed.
.The previous approach to handle dangling pages using hypothetical node solves the philosophical issue by including all the nodes in the computation. But because of this hypothetical node the number of iteration is increased too much. The Page rank value of hypothetical node is always too high as compare to all other node so it can be ignored. If for converge the algorithm the value of hypothetical node is not compared then number of iteration can be decreased. The value of all other nodes is set in few iteration only hypothetical node take large number of iteration to set their value. For solving all the issue related to dangling node the modified algorithm is proposed which does not compare the value of hypothetical node for the algorithm to converge and produce accurate page rank value of dangling node along with non-dangling node.

Modified Proposed algorithm

Input – Graph of web Pages
Output- Page Rank of every page

1. K ← 1, d←0.85.
2. Repeat for i from 0 to no_of_vertexes
   a. Prin ←0.
   b. Repeat
      for m from 0 to no_of_vertexes
         If (adjacency matrix[m][i]==1) then
         Prin = Prin + vertexes[m].Pr/vertexes[m].out
         End if.
   c. End loop.
3. Vertexes[i].Prin = (1-d) + (d* Prin).
4. End loop.
5. Repeat
   For l from 0 to no_of_vertexes
      a. Compare pr of each vertex with respective Prin of each vertex leaving
      Hypothetical vertexes
6. If same value are not achieved then
   a. Repeat
      For h from 0 to no_of_vertexes
      (i) Vertexes[h].Pr = vertexes[h].Prin
      (ii) End loop.
   b. Repeat from step 2
   c. K++.
7. End if
8. End loop.

In the modified algorithm first the value of damping factor d and value of k is set. The value of d is set equal to 0.85 which is commonly used value for d. K is used to show the number of iteration. Out represent the out degree of node. After that the intermediate page rank value (Prin) of each node is set equal to 0. Again run the for loop is run for all the vertex and wherever there is one in the matrix corresponding to some node then calculate the intermediate page rank value of node by the use of formula Prin = Prin + vertexes[m].pr/vertexes[m].out. Then the page rank value of the each node is calculated at the end of first for loop by the use of formula vertexes[i].Prin = (1-d) + (d* Prin). Again a for loop is constructed for all the vertexes, for comparing the page rank value of each node (prin) at any iteration with the previous page rank value (pr) leaving the value of hypothetical vertexes. If the same results are not achieved then again for loop is run for all the vertexes and store the page rank value of all the vertexes in pr by vertexes[h].Pr = vertexes[h].Prin. Again repeat from step 2 and increment the value of K. Else if the same results are achieved then the algorithm is converged.

On this modified algorithm if any web graph is run then it takes less number of iteration than the existing PageRank algorithm. Previously the algorithm is converged when at some iteration all the nodes in the graph is same value in the previous iteration. But now the algorithm is waiting until value of all nodes except the hypothetical node to become unchangeable in iterations. By this modified algorithm the computational issue is solved along with philosophical issue with hypothetical node approach. Now the page rank value of all the node (non-dangling and dangling) can be calculated without increasing the computational cost. Because of the high value of hypothetical node it can be ignored. The hypothetical node is added for solving the dangling problem and it is not a real page hence its accurate page rank value does not matter. At the point when Page rank value of each node except hypothetical node is set then at that point some value of hypothetical node is also obtained. The value of hypothetical node at this point may be inaccurate but it does not affect the value of dangling and non-dangling node so it does not matter. The matrix also becomes stochastic and it does not have 0^T row in the transition probability matrix corresponding to dangling page. Now page rank value of the all the nodes in the graph is obtained in less number of iteration than the number of iteration it takes when the web graph was run on the existing PageRank algorithm.

5. EXPERIMENTAL RESULTS

For experimentally showing the modified algorithm the web graph shown in figure 8 used as a input to the program (based upon the algorithm discussed above) the algorithm is implemented in C++. The calculation is performed up to 10 decimal places by using precision. The number of vertexes, the number of outgoing links from each vertex and the target node of each outgoing links is entered as an input. The program produces as output the Page Rank value of each node after the convergence of iteration and it also shows the out degree of each node. The output of the program is stored in a file to analyze the intermediate page rank value of each page. The experiment is performed by running the
Case 1: Run the web graph on the existing algorithm.
Case 2: Run the web graph on the modified algorithm.

Case 1: Run the web graph on the existing algorithm

The result after running the web graph shown in figure 8 on the existing algorithm is presented in figure 10. From the figure 10 it can be observed that it takes 135 iterations to converge. The last two rows (value in light purple color) show the values at 134th and 135th iteration are same so the algorithm is converged here. Hypothetical node is always the last node because it is added at last in the web graph if there is dangling node present in the web graph. From the Figure 10 it also observed that value of all other node except hypothetical node is same in both the iteration at the 37^{th} and 38^{th} iteration (value in orange color) but because of hypothetical node iteration is went up to 135 iterations. The value of hypothetical node is always very high in comparison to other

nodes at the 135the iteration. The value of hypothetical node is 4.8575989242 which is very high when compare it to all other value it can be ignored as it is discussed in [7].

Case 2: Run the web graph on the proposed modified algorithm.

Figure 11 is the result which is produced after executing the web graph shown in figure 8 on the modified algorithm. Now the number of iteartion is 38. From the Figure 11 it can be observed that the algorithm is now converged at the 38th iteration. The value of all node except hypothetical node is same in both the iteration at the 37th and 38th (value in light purple color). The value of hypothetical node at 37th iteration is 4.8404751923 and value of hypothectical node at 38th iteration is 4.8430437519 both value are not same but the algorithm is converged accoring to modified algorithm.The value of hypothetical node is still high and can be ignored because of having to high value when compare to page rank value of other node in the graph.

1: 0.1499999765	0.1499999765	0.1499999765	0.1499999765	0.1499999765	0.1499999765	0.1499999765
2: 0.1924999702	0.2987499563	0.2349999646	0.2349999646	0.2349999646	0.2137499681	0.5324999271
3: 0.2346457995	0.3709999494	0.2711249612	0.2891874595	0.3012291250	0.2498749640	0.9840623805
4 :0.2551166314	0.4136475157	0.2984194042	0.3215996098	0.3370530809	0.2652280879	1.4548915117
5: 0.2672001092	0.4402311662	0.3134029156	0.3394831568	0.3583200007	0.2768282271	1.8985968032
.....						
.....						
.....						
.....						
36:0.2850075278	0.4764972300	0.3343840189	0.3657596627	0.3886394354	0.2921131876	4.8374533588
37: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8404751923
38: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8430437519
39 :0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8452270279
40: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8470828121
41: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8486602289
.....						
.....						
.....						
.....						
131: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8575989221
132: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8575989228
133: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8575989235
134 :0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8575989242
135: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8575989242

Figure 10: Result after applying the Existing PageRank algorithm on Web Graph shown in Figure 8.

1: 0.1499999765	0.1499999765	0.1499999765	0.1499999765	0.1499999765	0.1499999765	0.1499999765
2: 0.1924999702	0.2987499563	0.2349999646	0.2349999646	0.2349999646	0.2137499681	0.5324999271
3: 0.2346457995	0.3709999494	0.2711249612	0.2891874595	0.3012291250	0.2498749640	0.9840623805
4: 0.2551166314	0.4136475157	0.2984194042	0.3215996098	0.3370530809	0.2652280879	1.4548915117
5: 0.2672001092	0.4402311662	0.3134029156	0.3394831568	0.3583200007	0.2768282271	1.8985968032
.....						
.....						
.....						
.....						
.....						
35: 0.2850075271	0.4764972279	0.3343840167	0.3657596627	0.3886394332	0.2921131869	4.8338982634
36: 0.2850075278	0.4764972307	0.3343840189	0.3657596627	0.3886394354	0.2921131876	4.8374533588
37: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8404751923
38: 0.2850075285	0.4764972307	0.3343840189	0.3657596634	0.3886394361	0.2921131883	4.8430437519

Figure 11: Result after applying the Modified algorithm on Web Graph shown in Figure 8.

The iteration at both the case is shown in table 1. From the table it is clear if the number of iteration in case 2 (when the web graph is run on the modified algorithm) is less than the number of iteration in case 1(when the web graph is run on the existing algorithm.). So by the modified algorithm all the issues related to dangling pages are solved with out increasing the computational cost.

Table 1: Number of Iterations in both the Cases

Case	Number of iteration
Case 1	135
Case 2	38

6. CONCLUSION

This paper contain all the issue associated with dangling Pages present on the web like philosophical and computational issue and also the problem which is raised due to dangling node removal. The paper also covers the some previous work done to handle the dangling page. The hypothetical node solution solves the philosophical issue but computational issue is increased because number of iteration is increased too much. In the paper it is also described how the computational issue is increased by the hypothetical node and modified algorithm for solving this computational problem. The modified algorithm to handle this computational issue is discussed by not comparing the value of hypothetical node to do not change for the algorithm to converge. Experimental results are also shown for better understanding. Our modified algorithm provide accurate value of non-dangling node and dangling node but now it take less number of iteration compare to previous solution. When the algorithm converges we get some value of hypothetical node. Hypothetical node value at this point can be ignored because of having high page rank value. So the modified algorithm provided above reduces the number of iteration when the hypothetical node is included in the web graph. Now the all the issue is solved by the hypothetical node approach without increasing the computational cost.

7. REFERENCES

[1] Brin, S., Page, L., Motwani, R. and Winograd, T. 1999 "The Pagerank Citation Ranking: Bringing order to the Web". Technical Report, Stanford Digital Libraries SIDL-WP-1999-0120.

[2] Eiron, N., McCurley, K. and Tomlin. 2004. J. "Ranking the Web Frontier", In the Proceedings of the the 13th international conference on World Wide Web WWW '04, 309-318.

[3] Norris, J. 1996. "Markov Chains", Cambridge University Press, 1-4.

[4] Lee, L. 2010. "Page Rank Algorithm and Development".

[5] Langville, A.N. and Meyer, C.D. 2005. Deeper Inside PageRank. Internet Mathematics, 1(3), 335-380

[6] Wang, Tao, X. T., Sun, J. T., Shakery, A. and Zhai, C. 2008 "DirichletRank: Solving the Zero-One-Gap Problem of PageRank". In ACM Transactions on Information Systems (TOIS), 26(2), 1-29.

[7] Singh, A.K., Kumar, P.R. and Leng, A. G. K. 2010 "Efficient Algorithm for Handling Dangling Pages Using Hypothetical Node". In the proceedings of 6th International Conference on Digital Content, Multimedia Technology and its Applications (IDC), 44 – 49.

[8] Chakrabarti, S. , Dom, B. , Gibson, D. , Kleinberg, J.M. , Raghavan, P. and Rajagopalan,.S. 1998 "Automatic resource compilation by analyzing hyperlink structure and association test", In Proceedings Of The Seventh International World Wide Web Conference, 65-74,

[9] Langville, A. N. and Meyer, C. D. 2005 "A survey of eigenvector methods of web information retrieval", In SIAM Review, 47(1), 135-161.

[10] Langville, A. N. and Meyer, C.D. 2006 "A Reordering for the PageRank problem". In SIAM Journal on Scientific Computing, 27(6), 2112–2120.

[11] Brin, S., Motwani, R., Page, L. and Winograd, T. 1998 What can you do with a web in your pocket?. Bulletin of the Technical Committee on Data Engineering, 21(2), 37–47.

[12] Lee, C.P., Golub, G.H. and Zenios,S.A. 2003. Partial state space aggregation based on lump ability and its application to pagerank. Technical report, Stanford University.