ANT PAGERANK ALGORITHM

Mahmoud Zaki Abdo, Manal Ahmed Ismail
Communication Dept. Faculty of Engineering, Helwan University, Cairo, Egypt

Mohamed Ibraheem Eladawy
Communication Dept. Faculty of Engineering, Helwan University, Cairo, Egypt

Keywords: Search engine, PageRank algorithm, Ranking algorithms, Ant algorithm.

Abstract: The amount of global information in the World Wide Web is growing at an incredible rate. Millions of results are returned from search engines. The rank of pages in the search engines is very important. One of the basic rank algorithms is PageRank algorithm. This paper proposes an enhancement of PageRank algorithm to speed up the computational process. The enhancement of PageRank algorithm depends on using the Ant algorithm. On average, this technique yields about 7.5 out of ten relevant pages to the query topic, and the total time reduced by 19.9%.

1 INTRODUCTION

The amount of information on the web is growing rapidly, as well as the number of new users inexperienced in the art of web search engine. World Wide Web search engines have become the most heavily used online services, with millions of searches performed each day (B. Davison, 2000). The web search engines use the structure present in hypertext to provide much higher quality search results (S. Brain, 1998).

The rank of pages is an integral component of any search engine. In the context of the web search engines, the role of ranking becomes even more important. The most important researches proposing Link Analysis Ranking (A. Borodin, 2005) are found in Kleinberg (J. Kleinberg, 1999), and Brin and Page (S. Brain, 1998).

Link Analysis Ranking can be described as the use of hyperlink structure for the purpose of ranking web documents. Link Analysis Ranking operates on the graph representation of hyperlinked web documents. The hyperlink graph is based on the representation of a web page as a node and hyperlink between pages as a directed edge. The goal of Link Analysis Ranking is to extract this information, and use it to determine a particular weight for every page, and use these weights to rank the web documents. This paper proposes an enhancement to speed up the computation of PageRank algorithm (S. Brain, 1998) by using Ant algorithm.

The rest of the paper is organized as follows. Section two presents an overview on the Link Analysis Ranking Algorithms. Section three explains Ant System. Section four explains the proposed Ant PageRank algorithm. Section five presents the experimental data preparation. Section six presents experimental results. Finally, Section seven concludes the paper.

2 LINK ANALYSIS RANKING ALGORITHMS

All the Link Analysis Ranking algorithms start with a collection of Web pages to be ranked. These algorithms proceed as follows:

- **Extraction**: extracting the hyperlinks between the pages
- **Construction**: constructing the underlying hyperlink graph. The hyperlink graph is constructed by creating a node for every Web page, and a directed edge for every hyperlink between two pages.
- **Calculation of Node Weight**: The graph is given as input to the Link Analysis Ranking algorithms. The algorithms operate on the
graph, and calculate a weight for each Web page. This weight is used to rank the pages.

In the following two subsections; the **In-degree algorithm** (P. Tasparas, 2004) and **PageRank algorithm** (S. Brain, 1998) will be briefly described as examples of Link Analysis Ranking algorithms.

### 2.1 In-degree Algorithm

The In-degree algorithm depends on the popularity of pages. The number of pages that link to this page measures the popularity of a page. It ranks pages according to their **in-degree** pages (the number of pages that link to the page).

### 2.2 The PageRank Algorithm

The algorithm depends on the computation of the **PageRank weight** of all pages in the graph by Eq. (1). The PageRank weight of a page A is given as follows:

\[
PR(A) = (1 - d) + d \left( \frac{PR(T_1)}{C(T_1)} + \frac{PR(T_2)}{C(T_2)} + \ldots + \frac{PR(T_n)}{C(T_n)} \right) \tag{1}
\]

where:
- PR (A) is the PageRank weight of the page A.
- Page A has pages T₁, T₂ ... Tₙ which point to it, as shown in Figure 1.
- The parameter \(d\) is a damping factor which can be set between \((0, 1)\); usually set \(d\) to 0.85.
- \(C(T)\) is the number of links going out of page T.
- \(B(A)\) the set of nodes that point to node A (Backwards links).

![PageRank graph](image)

The intuition underlying the In-degree algorithm is that a page has a good weight is a page that is pointed to by many nodes in the graph. Brin and Page (S. Brain, 1998) extended this idea further by observing that not all pages have the same weight. Links from pages of high quality should confer more weight. It is not only important how many pages point to a page, but also what the quality (value of the page weight) of these pages is.

Therefore, Brin and Page (S. Brain, 1998) had proposed a one-level weight propagation scheme, where a page has a good weight is the one that is pointed by many pages have a good weights. Figure 2 shows the PageRank algorithm.

![PageRank Algorithm](image)

Due to the huge size of actual web, an approximate iterative computation is usually applied to calculate the PageRank weight. This means that each page is assigned an initial starting value and the PageRank weights of all pages are then calculated in several computation circles based on Eq. (1). The minimum PageRank of a page is given by \((1 - d)\); while the maximum PageRank is determined as \(dN + (1 - d)\), where \(N\) is the number of pages. This maximum PageRank weight can theoretically achieved, only when all web pages solely link to one page, and this page also solely links to itself (Nan Ma, 2008).
3 ANT SYSTEMS

Ant System introduced by Dorigo (Dorigo, 1991). Dorigo's artificial ants (called ants) have some major differences with real (natural). First, ants have a memory. Second, ants are not completely blind. While natural ants rely on chemical signals to navigate. Finally, due to obvious constraints imposed by the architecture of current computers, artificial ants live in a world where time is discrete (Islam, 2005). In the case of Travel salesman Problem (TSP) - Ants have a memory; this memory is used to store a list of previously visited cities. - Ants are not completely blind. Ants are aware of the distance between cites.

To examine the ant algorithms, apply them to the well-known travelling salesman problem (TSP) (E. L. Lawler, 1985)

3.1 Ant System and Travel salesman Problem (TSP)

Given a set of n towns, the TSP problem can be stated as the problem of finding a minimal length closed tour that visits each town once. The distance between town i and town j is calculated by Euclidean distance equation (2).

\[ d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \]  

An instance of the TSP problem is given by a weighted graph (N,E), where N is the set of towns and E is the set of edges between towns, weighted by the distances. Assume \( b_i(t) \) is the number of ants in town i at time t. Then the total number of ants is calculated by equation (3)

\[ m = \sum_{i=1}^{n} b_i(t) \]  

Each ant is a simple agent with the following characteristics:

- When going from town i to town j it lays a substance, called trail, on edge (i,j);
- It chooses the town to go to with a probability that is a function of the town distance and of the amount of trail present on the connecting edge.
- Each ant has a data structure, called a tabu list. That memorizes the towns already visited up to time t and forbids the ant to visit them again before a tour has been completed. When a tour is completed the tabu list is emptied and the ant is free again to choose its way. The vector containing the tabu list of the k-th ant is tabu_k and tabu_k(s) is the s-th element of the tabu list of the k-th ant. Let \( \tau_{ij}(t) \) be the intensity of trail on edge (i,j) at time t. At each iteration of the algorithm trail intensity becomes

\[ \tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta \tau_{ij}(t+1) \]  

where \( \rho \) is a coefficient such that \((1 - \rho)\) represents the evaporation of trail. The coefficient \( \rho \) must be set to a value <1 to avoid unlimited accumulation of trail

\[ \Delta \tau_{ij}(t+1) = \sum_{k=1}^{m} \Delta \tau^k_{ij}(t+1) \]  

where \( \Delta \tau^k_{ij}(t+1) \) is the quantity per unit of length of trail substance (pheromone in real ants) laid on edge (i,j) by the k-th ant between time t and t+1.

The transition probability from town i to town j for the k-th ant is

\[ p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \eta_{ij}^\beta}{\sum_{j \in \text{allowed}} [\tau_{ij}(t)]^\alpha \eta_{ij}^\beta} & \text{if } j \text{ is allowed} \\ 0 & \text{otherwise} \end{cases} \]  

Different choices about how to compute \( \Delta \tau^k_{ij}(t+1) \) and when to update the \( \tau_{ij}(t) \) cause different Instantiations of the ant algorithm. In the next two sections Dorigo (Dorigo, 1991) present the three approaches. Dorigo (Dorigo, 1991) used as experimental test-bed for ideas, namely Ant-density, Ant-quantity, and Ant-cycle.

3.2 The Ant-density and Ant-quantity Approaches

Initially, two approaches were developed by Dorigo (Dorigo, 2000) to exploit his Ant System heuristic named: Ant-density and Ant-quantity. In the Ant-density approach a quantity Q1 of trail for every unit of length is left on edge (i,j) and when to update the \( \tau_{ij}(t) \) cause different Instantiations of the ant algorithm. In the next two sections Dorigo (Dorigo, 1991) present the three approaches. Dorigo (Dorigo, 1991) used as experimental test-bed for ideas, namely Ant-density, Ant-quantity, and Ant-cycle.
Therefore, in the Ant-density approach
\[ \Delta \tau_{ij}(t, t+1) = \begin{cases} Q_1 & \text{if } k - \text{th ant goes from } i \text{ to } j \text{ between } t \text{ and } t+1 \\ 0 & \text{Otherwise} \end{cases} \quad (7) \]

And in the Ant-quantity approach
\[ \Delta \tau_{ij}(t, t+1) = \begin{cases} \frac{Q_2}{L_k} & \text{if } k - \text{th ant goes from } i \text{ to } j \text{ between } t \text{ and } t+1 \\ 0 & \text{Otherwise} \end{cases} \quad (8) \]

where: \( Q_1 \) and \( Q_2 \) are the same quantity of pheromone. Thus, an increase in trail intensity on edge \((i,j)\) when an ant goes from \(i\) to \(j\) is independent of \(d_{ij}\) in the Ant-density approach and is inversely proportional to \(d_{ij}\) in the Ant-quantity approach (i.e., shorter edges are made more desirable by ants in the Ant-quantity approach, thus further reinforcing the visibility factor in equation (6)).

### 3.3 The Ant-cycle Approach

The approach introduced a major difference with respect to the two previous systems. Here \( \Delta \tau_{ij} \) is not computed at every step, but after a complete tour (n steps). The value of \( \Delta \tau_{ij}(t, t+n) \) is given by
\[ \Delta \tau_{ij}(t, t+n) = \begin{cases} \frac{Q_3}{L_k} & \text{if } k - \text{th ant uses edge } (i,j) \text{ in its tour} \\ 0 & \text{Otherwise} \end{cases} \quad (9) \]

where \( Q_3 \) is a constant and \( L_k \) is the tour length of the \( k \)-th ant. The entire Ant-Cycle approach is demonstrated in (Islam, 2005)

![Figure 3: The Path of one ant with high in-degree selected from out-degree pages.](image)

### 4 PROPOSED ANT PAGERANK ALGORITHM

The main objective behind Ant PageRank algorithm is reducing the execution time for computing the PageRank weight of the pages. The PageRank weight of the page depends on the weight of the in-degree pages as shown in equation (1). The proposed algorithm locates ants in the pages without in-degree pages, and release the ant move on the graph (as Figure 3) and calculate the PageRank weight of the pages in the tabu of the ant.

The ant moves to one page from out-degree page. This paper introduces novel PageRank approaches to be used the conjunction with the Ant algorithm.

1. **Ant PageRank approach 1**: Locate one ant in every page without in-degree pages. The ant moves to one page from out-degree pages as random selected.
2. **Ant PageRank approach 2**: Locate one ant in every page without in-degree pages. The ant moves to one page from out-degree page as high in-degree page selected.
3. **Ant PageRank approach 3**: Locate two ants in every page without in-degree pages. Every ant move to one page from out-degree page as random selected.

Figure 4 shows the Ant PageRank algorithm. The algorithm divided into three approaches:

- **Approach 1**: Locate one ant in every page without in-degree and select the next page as random.
- **Approach 2**: Locate one ant in every page without in-degree and select the next page as random.
higher in-degree page.
- Approach 3: Locate two ants in every page without in-degree and select the page as random.

The change of the number of ants and the method of selecting the next page of the ant are used to get the best time and high top ten pages for matching of the result in the PageRank algorithm.

5 EXPERIMENTAL DATA PREPARATION

This section presents a brief description of experimental results of the proposed algorithm. The experimental results data are text files (P.Tsaparas, 2008). These text files contain information about queries used in the experimental phase. These text files represent the following 33 queries namely:


Each query represented by three text files named "nodes", "adj_list", and "inv_adj_list".

The nodes.txt file is formatted as follows:
- Number of pages
- Information of each page, each page is described as follows [Page ID, http address of the page, page title, number of in-degree pages, number of out-degree pages]

The Adjacency List file contains a list of out-degree page IDs for each page. An example of a page entry can be describes as follow:
1:20 500 6
This means that the page with ID= 1, points to the pages with IDs = {20, 500, 6}.

The Inverted Adjacency List file contains a list of in-degree page IDs for each page. An example of a page entry can be describes as follow:
20:1 5 80 -1
This means that the page with ID = 20, is pointed to by the pages with IDs = {1, 5, 80}.

Table 1: Array of structure to represents the data of the graph.

<table>
<thead>
<tr>
<th>Node 1 ID</th>
<th>Node 1 Indegree Number</th>
<th>Node 1 Outdegree Number</th>
<th>Node 1 Indegree</th>
<th>Node 1 Outdegree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 2 ID</td>
<td>Node 2 Indegree Number</td>
<td>Node 2 Outdegree Number</td>
<td>Node 2 Indegree</td>
<td>Node 2 Outdegree</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

The implementation steps of the proposed algorithm are detailed as follow:
1- Data preprocessing step.
2- Executing proposed algorithm.
3- Comparing the results.

The preprocessing step includes the creation of an array of structure to represent the data. The structure is as follows:
- **NodeID**: a unique identifier for the Page
- **NodeIndegree Number**: Number of in-degree Pages
- **NodeOutdegree Number**: Number of out-degree Pages
- **[ ] NodeIndegree**: Array of NodeID of in-degree pages
- **[ ] NodeOutdegree**: Array of NodeID of out-degree pages

where: Length of the array is number of pages.

Table 1 shows this structure. This structure only needs to store ID of in-degree pages and ID of out-degree pages of each node. This structure replaced by the matrix $W_n^{\omega_n}$ used in the classical PageRank algorithm as explained in subsection 2-2.

6 EXPERIMENTAL RESULTS

This section illustrates experiment results of the proposed algorithm. The experiments done by running the proposed algorithm on the 33 query illustrated in section 5. The comparison between the proposed algorithm and the classical PageRank algorithm depends on the following factors:
- Number of memory cells used.
- Number of iteration for each query.
- Total computation time.
- Number of pages matched in top ten pages with the classical PageRank algorithm.
The main contributions of the proposed algorithm are:

- The percentage in average of the memory used in the proposed Ant PageRank algorithm versus conventional PageRank algorithm is 0.2 %. The memory used in the classical PageRank algorithm is N^2, while the memory used in the proposed algorithm is based on the number of in-degree pages and the number of out-degree pages as shown in Table 1.

- The percentage in average of the number of iteration in all pages in the classical PageRank algorithm versus Ant PageRank algorithms are (as shown in Table 2):
  - Ant PageRank algorithm approach 1 reduces the number of iteration by 95.5 %.
  - Ant PageRank algorithm approach 2 reduces the number of iteration by 96.1 %.
  - Ant PageRank algorithm approach 3 reduces the number of iteration by 93.9 %.

The average of the computation time in all pages in the classical PageRank algorithm versus Ant algorithms are (as shown in Table 2):
  - Ant PageRank algorithm approach 1 reduces the computation time by 19.9 %.
  - Ant PageRank algorithm approach 2 reduces the computation time by 21.5 %.
  - Ant PageRank algorithm approach 3 reduces the computation time by 18.7 %.

The average top ten pages results are matched with the classical PageRank algorithm in each approach are (as shown in Table 2):
  - Ant PageRank algorithm approach 1 is 7.5 pages.
  - Ant PageRank algorithm approach 2 is 6.9 pages.
  - Ant PageRank algorithm approach 3 is 7.5 pages.

### Table 2: Memory used, total number of iterations, computation time, and number of pages matched for classical PageRank algorithm versus Ant PageRank algorithm approaches

<table>
<thead>
<tr>
<th>Query</th>
<th>Number of Pages (N)</th>
<th>Memory used (Cell)</th>
<th>Number of iteration</th>
<th>Computation time (Sec)</th>
<th>Number of pages matched with PR alg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>abortion</td>
<td>3340</td>
<td>11135000</td>
<td>14874</td>
<td>16360</td>
<td>5197</td>
</tr>
<tr>
<td>affirmative action</td>
<td>1233</td>
<td>1365329</td>
<td>131196</td>
<td>13383</td>
<td>3910</td>
</tr>
<tr>
<td>alcohol</td>
<td>4594</td>
<td>21104836</td>
<td>33432</td>
<td>220152</td>
<td>20642</td>
</tr>
<tr>
<td>automobile parks</td>
<td>7399</td>
<td>6474201</td>
<td>72452</td>
<td>369500</td>
<td>12484</td>
</tr>
<tr>
<td>architecture</td>
<td>1136</td>
<td>13450616</td>
<td>6134</td>
<td>52624</td>
<td>1875</td>
</tr>
<tr>
<td>automobile industries</td>
<td>3225</td>
<td>15040623</td>
<td>16318</td>
<td>148350</td>
<td>6255</td>
</tr>
<tr>
<td>basketball</td>
<td>6049</td>
<td>2695401</td>
<td>48818</td>
<td>229862</td>
<td>12205</td>
</tr>
<tr>
<td>bluebird</td>
<td>5554</td>
<td>26663316</td>
<td>48778</td>
<td>256692</td>
<td>10250</td>
</tr>
<tr>
<td>cheese</td>
<td>3266</td>
<td>10666756</td>
<td>23320</td>
<td>150236</td>
<td>6526</td>
</tr>
<tr>
<td>classical guitar</td>
<td>3150</td>
<td>9922500</td>
<td>24088</td>
<td>113400</td>
<td>5114</td>
</tr>
<tr>
<td>complexity</td>
<td>3564</td>
<td>12702096</td>
<td>26962</td>
<td>123804</td>
<td>5139</td>
</tr>
<tr>
<td>computational complexity</td>
<td>1075</td>
<td>1155625</td>
<td>4842</td>
<td>387900</td>
<td>1504</td>
</tr>
<tr>
<td>computational geometry</td>
<td>2250</td>
<td>3253204</td>
<td>63678</td>
<td>389044</td>
<td>3313</td>
</tr>
<tr>
<td>death penalty</td>
<td>4286</td>
<td>18472804</td>
<td>43912</td>
<td>171026</td>
<td>6194</td>
</tr>
<tr>
<td>generic</td>
<td>5295</td>
<td>2808084</td>
<td>38522</td>
<td>227814</td>
<td>10551</td>
</tr>
<tr>
<td>geometry</td>
<td>4526</td>
<td>18714276</td>
<td>26726</td>
<td>211974</td>
<td>7390</td>
</tr>
<tr>
<td>globalisation</td>
<td>4334</td>
<td>18715556</td>
<td>38481</td>
<td>215700</td>
<td>6553</td>
</tr>
<tr>
<td>gun control</td>
<td>2915</td>
<td>8732625</td>
<td>23476</td>
<td>153669</td>
<td>4765</td>
</tr>
<tr>
<td>iraq war</td>
<td>3782</td>
<td>14301524</td>
<td>30746</td>
<td>158484</td>
<td>3509</td>
</tr>
<tr>
<td>jaguar</td>
<td>2820</td>
<td>7952400</td>
<td>16784</td>
<td>141000</td>
<td>5116</td>
</tr>
<tr>
<td>jordan</td>
<td>4099</td>
<td>16072861</td>
<td>21874</td>
<td>200850</td>
<td>7799</td>
</tr>
<tr>
<td>movies</td>
<td>7687</td>
<td>14370389</td>
<td>57563</td>
<td>294779</td>
<td>15754</td>
</tr>
<tr>
<td>national parks</td>
<td>4757</td>
<td>22662049</td>
<td>28312</td>
<td>228336</td>
<td>10148</td>
</tr>
<tr>
<td>net censorship</td>
<td>2598</td>
<td>6749604</td>
<td>13776</td>
<td>124304</td>
<td>3927</td>
</tr>
<tr>
<td>randomised algorithms</td>
<td>742</td>
<td>5001364</td>
<td>2410</td>
<td>6678</td>
<td>1128</td>
</tr>
<tr>
<td>recipes</td>
<td>5243</td>
<td>2490409</td>
<td>65064</td>
<td>272638</td>
<td>10540</td>
</tr>
<tr>
<td>Russell</td>
<td>2790</td>
<td>778443</td>
<td>45403</td>
<td>208670</td>
<td>41060</td>
</tr>
<tr>
<td>search engines</td>
<td>11059</td>
<td>13992821</td>
<td>844722</td>
<td>524655</td>
<td>17025</td>
</tr>
<tr>
<td>shakespeare</td>
<td>4383</td>
<td>92016983</td>
<td>27180</td>
<td>216180</td>
<td>8775</td>
</tr>
<tr>
<td>table tennis</td>
<td>1948</td>
<td>3786704</td>
<td>10930</td>
<td>54654</td>
<td>3840</td>
</tr>
<tr>
<td>vintage cars</td>
<td>3440</td>
<td>11974600</td>
<td>25912</td>
<td>148705</td>
<td>4730</td>
</tr>
<tr>
<td>war</td>
<td>6011</td>
<td>44716120</td>
<td>60344</td>
<td>350693</td>
<td>10947</td>
</tr>
</tbody>
</table>

| percentage in average | 100 | 0.2 | 100 | 4.5 | 100 | 5.9 | 6.1 | 100 | 80.1 | 78.5 | 77.3 | 7.5 | 6.8 | 7.5 |

- Ant PageRank algorithm approach 2 reduces the number of iteration by 96.1 %.
- Ant PageRank algorithm approach 3 reduces the number of iteration by 93.9 %.

- Ant PageRank algorithm approach 1 reduces the computation time by 19.9 %.
- Ant PageRank algorithm approach 2 reduces the computation time by 21.5 %.
- Ant PageRank algorithm approach 3 reduces the computation time by 18.7 %.

- Ant PageRank algorithm approach 1 is 7.5 pages.
- Ant PageRank algorithm approach 2 is 6.9 pages.
- Ant PageRank algorithm approach 3 is 7.5 pages.
7 CONCLUSIONS

This paper has proposed the Ant PageRank algorithm. This algorithm has a flexible parameters to control the behaviour of the Ant algorithm. These can be achieved through three approaches, which are proposed to enhance the classical PageRank algorithm. The enhancement is based on two factors, computation time and memory required.

As demonstrated, the top ten pages which are achieved by the proposed algorithm are matched with the top ten pages of the classical PageRank algorithm, with lower execution time, number of memory cells, and number of iteration. Presents

The best experimental is Ant PageRank algorithm experimental I. This experimental use one ant with random selected. This experimental reduces the percentage in average execution time by 19.9% and matches with 7.5 pages from top ten pages with classical PageRank algorithm.

REFERENCES


