

# E-Business's Page Ranking with Ant Colony Algorithm

Asst. Prof. Chonawat Srisa-an, Ph.D.

Faculty of Information Technology, Rangsit University  
52/347 Phaholyothin Rd. Lakok Pathumthani, 12000  
chonawat@rangsit.rsu.ac.th, chonawat@yahoo.com

## **Abstract**

*Almost all E-Business transactions use classical keyword-based methods for searching commercial goods. One of the main problems that plague modern search engines is poor relevance. In many situations, search engines retrieve thousands of pages at least partially satisfying input query. Of course, human attention remains more or less constant, and most humans are able to cope only with about 100-200 retrieved documents.*

*PageRank index method has been successfully used for search engine output sorting. Two such experimental search engines have been constructed at Stanford University [1]. There seems to be quite high correlation between high PageRank index, and general page importance judged by human users. This is especially spectacular for very general queries, for which a lot of relevant pages exist in the Internet.*

*This work proposes an algorithm for page rank called Ant Colony Ranking (ACR) algorithm. The goal of this paper is to investigate the Ant Colony Ranking (ACR) algorithm in the context of such self-modifying systems to solve two drawbacks of this method. The experiments show that the algorithm has good convergence properties over hypertext structure of the Internet.*

Keywords: web mining, text mining, information retrieval, search engines, Expert Systems and AI in e-Business

## **1. Introduction**

In many situations, search engines retrieve thousands of pages at least partially satisfying input query. In E-business world, people have been overwhelmed with those non-relevant pages every day. PageRank succeeds in such situations by separating highly respected sites from junk pages that only happened to contain words from the query. PageRank is therefore a method that would allow us to approximate page importance for the user, regardless of its relevance to the query measured by classical methods. Having such approximation -in fact a ranking score -we can either sort the pages, presenting these with higher rank first, or even entirely remove these which have very poor rank thus allowing user to concentrate on a set of pages which has reasonable quantity.

There are two drawbacks of this classical method. First, one of the most important drawbacks of this methodology is the necessity to have access to (ideally) entire Web structure to perform proper computation. This is possible only in large

systems that maintain entire Web hyperlink databases [4] such as general-purpose search engines and web crawlers. The first one emphasizes the research effort on Web topology and structural dependencies between links and documents, without examining contents of hypertext documents. It proves that the structure of Web itself carries lots of useful information that should be retrieved.

Secondly, the other drawback of this classical methodology is the difficulty to find out how many times we need to repeat the calculation for big networks. That's a difficult question; for a network as large as the World Wide Web [2] it can be many millions of iterations! The "damping factor" is quite subtle. If it's too high then it takes ages for the numbers to settle, if it's too low then you get repeated over-shoot, both above and below the average - the numbers just swing about the average like a pendulum and never settle down. Also choosing the order of calculations can help. The answer will always come out the same no matter which order you choose, but some orders will get you there quicker than others.

The goal of this paper is to investigate the Ant Colony Ranking (ACR) algorithm in the context of such self-modifying systems to solve two drawbacks of a classical ranking method. The ACR software is to construct a web structure and compute ranking score for each page.

## 2. Standard Pagerank Formula

We assume page A has pages T1...Tn which point to it (i.e., are citations). The parameter d is a damping factor which can be set between 0 and 1. Also C(A) is defined as the number of links going out of page A. The PageRank of a page A is given as follows:

$$PR(A) = (1-d) + d (PR(T1)/C(T1) + \dots + PR(Tn)/C(Tn))$$

1. **PR(Tn)** - Each page has a notion of its own self-importance. That's "PR(T1)" for the first page in the web all the way up to "PR(Tn)" for the last page
2. **C(Tn)** - Each page spreads its vote out evenly amongst all of it's outgoing links. The count, or number, of outgoing links for page 1 is "C(T1)", "C(Tn)" for page n, and so on for all pages.
3. **PR(Tn)/C(Tn)** - so if our page (page A) has a backlink from page "n" the share of the vote page A will get is "PR(Tn)/C(Tn)"
4. **d(...** - All these fractions of votes are added together but, to stop the other pages having too much influence, this total vote is "damped down" by multiplying it by 0.85 (the factor "d")
5. **(1 - d)** - The (1 - d) bit at the beginning is a bit of probability math magic so the "sum of all web pages' PageRanks will be one": it adds in the bit lost by the **d(...** It also means that if a page has no links to it (no backlinks) even then it will still get a small PR of 0.15 (i.e. 1 - 0.85).

Note that the PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one.

PageRank or PR(A) can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web.

## 3. Ant Colony Ranking Algorithm

Ant Colony Ranking (ACR) algorithm is a system based on agents which simulate the natural behavior of ants, including mechanisms of cooperation and adaptation.

ACR algorithms are based on the following RULES:

- Each path followed by an ant is associated with a candidate solution for a given problem.
- When an ant follows a path, the amount of pheromone deposited on that path is proportional to the quality of the corresponding candidate solution for the target problem. Artificial ants have a probabilistic preference for paths with a larger amount of pheromone. Two kinds of pheromones are forward pheromone and backward pheromone on different purposes.
- Each destination has food available for only one artificial ant; therefore, each node ant needs to visit only once. When an ant has to choose between two or more paths, the path(s) with a larger amount of pheromone have a greater probability of being chosen by the ant.
- An appropriate representation of the problem, which allows the ants to incrementally construct/modify solutions through the use of a probabilistic transition rule, based on the amount of pheromone in the trail.
- Artificial Ants can be thought of having two working *forward and backward modes*. They are in forward mode when they are moving from the nest toward the food, and they are in backward mode when they are moving from the food backward mode when they are moving from the food back to their nest. Once an artificial ant in forward mode reaches its destination, it switches to backward mode and starts its travel back to the source on the same route they previously used and switch to its backward pheromone.
- A rule for pheromone updating, which specifies how to modify the pheromone trail (t). Artificial Ants use pheromone trail to guide the route. Assume that there is only available food for one ant; therefore, once an ant reaches the destination (food), it eliminates its pheromone on the way back to its nest to prevent loop.
- Artificial ants implement loop elimination by eliminate all pheromone and deposit backward pheromone to set up a path. The permanent path is created by “backward pheromone”.
- Each ant that following a forward pheromone has to give up its search once it finds backward pheromone on the path.
- In ACR system, the ants memorize the node they visit during the forward path and exchange its journey knowledge with other once it reaches its nest. The permanent path was created by backward pheromone and no longer use. At this point in time, information about incoming and outgoing for each node is store in a database.

Note: For clustering purpose, there are possibly many nests in the ACR system. The ACR software reduces size problems by clustering methods that decompose such problems into smaller ones.

#### 4. Ant Colony Ranking Software

In order to solve Web structure problem, this research replaces web crawler by Artificial Ants for the path construction phase and use web log analysis for better understanding of reference page. Since some reference path is hard to detect in WWW environment since any page can point to the page anytime, web log is used for a better result but not the main part of the system.

ACR software creates enough “Artificial Ants” to explore Web structure. All artificial ants must return their nest once

they reach a food (destination). Each ant detects `<a href="...">` and `</a>` tags in source code of web page and start its journey to construct a permanent path. Once it arrives home, the ACR software destroys the ant and then stores its knowledge in a database. Note: The case of topology heavily change is not in this paper scope.

The page can have high index, if it is pointed by many pages, or it has relatively few back-links, but coming from pages with high index value. This index works well in hypertext environments that can be represented by either non-cyclic, or heavily interconnected graphs. Unfortunately the structure of the World Wide Web is different, and the situations as presented below are not uncommon; therefore, the artificial ants implement loop elimination by depositing backward pheromone to set up a path. The permanent path is created by "backward pheromone". Moreover, normalized index can be interpreted as a probability measure of viewing a certain page by so called "random surfer", who randomly follows hyperlinks between web documents.

Web Servers' logs can be analyzed for information about popularity of certain web pages among surfers. Such data can help searching and browsing processes in a variety of ways. Self modifying hypertext environments, Hyperlinks have been devised as a means of facilitating navigation among huge number of documents. User navigation patterns and habits are however variable, and hence hyperlink structure is determined only by Web service creators, it can sometimes be ineffective and lead to confusion. There is even special name for such hyperlink structure inefficiency effect, predating emergence of the World Wide Web: the "lost in hyperspace phenomenon". ACR algorithm would reduce this problem because each ant memorizes the node they visit during the forward path and exchange its journey knowledge with other once it reaches its nest. The permanent path was created by

backward pheromone and was no longer used. With this knowledge and web log analysis a clear topology can be constructed.

In the case of large system, The ACR software reduces size problems by clustering methods that decomposes such problems into smaller ones. Separating large number of pages into many nest helps artificial ant to identify quickly which groups are relevant and important for it. Each artificial ant can live in one nest. Once all paths are explored, the sub structure is created. To compose into one big web structure is the task of ACR software. The ants memorize the node they visit during the forward path and exchange its journey knowledge with other once it reaches its nest. The permanent path was created by backward pheromone and no longer use. Once an ant reaches its nest, the ACR software destroys the ant and stores its knowledge in a database. Incoming and outgoing information of each node is store in the database for ranking computation using a standard formula above. Once all artificial ants arrive home, the whole structure is built. Page rank calculation can compute from information stored in the database.

## 5. Conclusion

The ACR algorithm and software was built to enhance ranking method by eliminating two main drawbacks including web structure construction and large iteration. Once all artificial ants arrived their nests, the whole structure was built. Page rank calculation can compute from information stored in the database.

The experiments show that the algorithm has good convergence properties over hypertext structure of the Internet.

## Acknowledgements

We would like to thank Prof. Chom Kimpan for discussions and invaluable advice given during preparation of this paper.

## 6. Bibliography

- [1] Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd, (1998), "The Page Rank Citation Ranking: Bringing order to Web", Stanford University Digital Libraries papers.
- [2] Sergey Brin, Lawrence Page (1998), "The Anatomy of a Large-Scale Hypertextual Web Search Engine", *Proc. of 7<sup>th</sup> International Web Conference*.
- [3] Jon M. Kleinberg (1998), "Authoritative Sources in Hyperlinked Environment", *Proc. of Ninth Annual ACM SIAM Symposium on Discrete Algorithms*.
- [4] D. Gibson, J. Kleinberg, P. Raghavan (1998), "Structural Analysis of the World Wide Web", IBM Almaden Research Center.
- [5] S. Chakrabati, Byron E. Dom, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, (1998) "Experiments in Topic Distillation", IBM Almaden Research Center
- [6] Botafogo R. A., Schneiderman B. (1991), "Identifying aggregates in hypertext structures", *Third ACM Conference on Hypertext*
- [7] P. Kazienko (1998), "Grupowanie stron WWW na podstawie odsy. aczy hipertekstowych", *MISSI'98 conf. proceedings*, Wroc.aw,
- [8] D. Gibson, J. Kleinberg, P. Raghavan, (1998) "Inferring Web Communities from Link Topology", *ACM Conference on Hypertext and Hypermedia proceedings, 9th*
- [9] P. Raghavan (1998), "Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text", *Proc. of 7th World Wide Web conference*,
- [10] S. Chakrabati, B. Dom, P. Indyk (1998), "Enhanced hypertext categorisation using hyperlinks", *ACM SIGMOD'98 proceedings*.
- [11] Matthew Merzbacher (1999), "Discovering Semantic Proximity for Web Pages", *ISMIS'99 conference proceedings*
- [12] P. Pirolli, J. Pitkow, R. Rao (1996), "Silk from a Sow's Ear: Extracting Usable Structures from Web", *Proc. of the Conference on Human Factors in Computing Systems: Common Ground*, pages 118-125, New York,