Web Navigation Analysis and Simulation using Ant Colony Optimization

Ekachai Jinhirunkul
Advanced Virtual and Intelligent Computing (AVIC) Center
Department of Mathematics, Faculty of Science
Chulalongkorn University
Bangkok, Thailand
Ekachai.J@Student.chula.ac.th

Peraphon Sophatsathit
Advanced Virtual and Intelligent Computing (AVIC) Center
Department of Mathematics, Faculty of Science
Chulalongkorn University
Bangkok, Thailand
Peraphon.S@chula.ac.th

Abstract—**This paper utilizes the Ant Colony Optimization algorithm to explore an unknown web site, mapping its structure and navigation routing so that accessibility and performance information can be attained. The investigation will also unveil changing structure of the web site adaptively such as new links, removed links, and unreachable links. As a consequence, coverage of all reachable nodes within the designated web site can be obtained, along with essential performance statistics to reflect near optimal accessible paths to any given node in the web site. By virtue of the simplicity of the Ant Colony Optimization algorithm, some straightforward mapping techniques were employed to entail opportunistic commercialization of the proposed algorithm.**

Keywords-Web navigation; Web structure analysis; Web agent; Ant Colony Optimization.

I. INTRODUCTION AND LITERATURE REVIEW

The Internet online service has become an integral part of our daily life. Many web site technologies precipitated from such insatiable demands have been implemented and deployed at an escalating pace. Web 2.0 is one example that has been developed and integrated into many online services and applications. Many people are able to develop their own web site or participate in various web blog services provided by the host owner. Better yet, general public can edit or add new web contents dynamically any time and anywhere. These capabilities facilitate several web sites to transform their web configuration from static to dynamic. Such an undertaking becomes an enormous responsibility for the administrator or webmaster to maintain satisfactory performance of their service as the size of web site increases. To assist the amount of innumerable development efforts expended by web developers, a simple, concise, yet highly efficient web site analysis method is proposed. The approach rests primarily on the notions of Swarm of Autonomous Agents [1] and Ant Colony Optimization (ACO) [2] algorithm to analyze the structure of a given web site, where ants (or crawler agents) are dispatched to map the site "terrain." In doing so, various site configurations can be discovered, whereby performance tuning, follow-up design modifications, reconfiguration, and the likes can also be carried out to improve web structure, namely, dead link removal, placement of newly added pages, unreachable page fixes, and excessively depth of page location, etc.

This paper is organized as follows. Section II states some prevalent issues pertaining to the dynamic nature of web navigation where the ACO algorithm can be applied. Section III encompasses the analysis of web navigation structure using the modified ACO algorithm. Various simulation parameters and pertinent information are also established as the results of the analysis. A number of experimental outcomes are illustrated and discussed in Section IV. Section V presents some final thoughts and future enhancement.

II. WEB NAVIGATION AND ANT COLONY OPTIMIZATION

Latest news and up-to-date information are vital edges in today's highly competitive businesses. Various technological approaches have been employed to carry out the tasks. The dynamic web page technique is one popular approach embraced by many online businesses. The technique helps the companies easily maintain the latest updated information of their web sites. As the number of web sites and their corresponding size grow exponentially, navigation through such a labyrinth becomes a formidable task. Some prevalent issues that influence web sites accessibility are (1) navigation paths cannot be straightforwardly established as web sites expand dynamically; (2) the number of unknown dead links and unreachable pages is difficult to determine as a direct consequence of (1); and (3) access and processing time are computationally difficult as complexity increases.

Bearing the above problematic issues in mind, an automated system seems to be a viable consideration. Studies show that the ACO is an efficient proposed technique to explore unknown systems and unknown environments offering three advantages, i.e., autonomy, self-organizing, and resilience. In order to apply the ACO algorithm to web navigation, three provisions need to be addressed, namely, (1) bound the ant agents to explore within the designated area; (2) modify the ACO algorithm to ensure that ant agents will cover all pages of the unknown web structure; and (3) establish a technique to translate all information gathered from the ant agents in a presentable form. In the sections that follow, the

modified ACO algorithm will be further elaborated to resolve the above provisions.

III. WEB NAVIGATION STRUCTURE SIMULATION BY ACO

This study utilizes a few known and accessible web sites as a preliminary experimental analysis to verify the viability of proposed approach. A number of relevant statistics are subsequently collected. We resorted to set up a simulation based on these preliminary statistics to procedurally explore the structure of any given web sites. The procedures can be performed step-by-step as follows:

A. Define some prerequisite theoretical fundaments based on ant's behavior;

B. Devise navigational path traversals using ACO, whereby related information can be orderly gathered;

C. Analyze the web navigation routing parameters; and

D. Create a site map, along with pertinent statistics, as a visual representation of the web site under investigation.

Figure 1 illustrates a conceptual view of the proposed web structure analysis approach. The ant agents hereafter will be simply referred to as ants. Details on analysis procedure are elucidated in the sub-sections that follow.

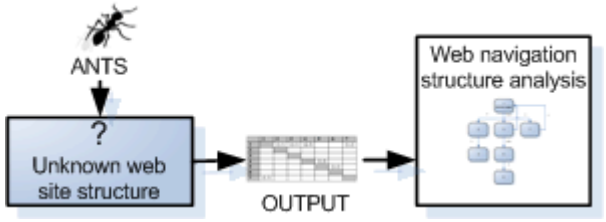

Figure 1 Web navigation structure analysis.

A. Define prerequisite fundaments based on ant's behavior

The ACO algorithm is derived from observations of real ant's behavior. An ant releases a chemical residue called *pheromone* along the path as it passes [3]. This behavior facilitates all following ants instinctively guess the appropriate path to their food based on the density of pheromone. Such a behavior was adopted by the ACO algorithm to successfully solve the popular travelling salesman problem [4].

Basically, ants can travel around their habitat in search for food. They simply react in accordance with the information obtained from the ants ahead along that path. In web navigation case, the above provision is inadequate to accommodate navigation through such complex interconnections, where ants can move to any web pages either inside or outside the web site under investigation. A common encounter for most commercial websites and online services is that they carry many advertisements, banners, or exchange links to the external websites. These are usually out of the administrator's or webmaster's responsibility to monitor and maintain efficient access and quality of service. To avoid indefinite cascading external links, the modified ACO algorithm will confine the URL access limit via ant's traversal based only on internal links. The proposed solution can be accomplished by simply creating an IP address table that keeps track of all connecting domain servers to be analyzed. As such, the ants can always access and verify their destination pages from this IP address table. Figure 2 shows an example of the IP address table.

ID	Internal IP addresses
1	161.100.100.0
2	162.100.100.0

Figure 2 A sample IP address table.

B. Web navigation and information gathering by the modified ACO algorithm

The original ACO algorithm was designed to discover the shortest path on a graph using the pheromone update technique based on the path that has the largest density of pheromone [5]. Our modified ACO algorithm reverts the ants' travel to the opposite direction that has the lowest density of pheromone first. This will help ants explore new paths, whereby increasing more coverage for web navigation structure analysis. The following Ant package structure and Ant algorithm are proposed to explore the web navigation paths by means of ant agents.

Ant package

An Ant package entry encompasses 4 elements shown in Figure 3.

S	T	TS	Age

Figure 3 An Ant package entry.

where *S* denotes the set of ant's visited pages; *T* is the set of time the designating ants spent moving from source page to target page, or equivalently speaking, it is the time used for loading all required web page contents; *TS* is the recorded timestamp captured after the ants arrive at the page in Julian day format [6]; and *Age* is the variable representing the available life time of the ants. Figure 4 shows the Ant package after the ants visiting to the pages 1 → 2 → 3 → 4 →1. The navigation time from page 1 to 2, 2 to 3, 3 to 4 and 4 to 1 are 5, 4, 4, and 6, respectively. The time that the ants arrive at the current page is 2454647 and the current ant's life time is 50.

{1,2,3,4,1}	{0,5,4,4,6}	2454647	50

Figure 4 A sample entry of the Ant package.

Ant algorithm

1. Update ant's Age

The ant's *Age* is a variable defined to limit the number of move for each ant. This variable will be reduced every time the ant arrives at a page. In other words, the age is decreased for each link followed and an ant's traversal will terminate when its age is zero, thus skipping steps 2-6. This helps prevent potential infinite looping problem.

The *Age* value will be initialized to *CONST_AGE* before sending an ant in the web site. A proper value of *CONST_AGE* should be more than the expected number of pages contained in the web sites so that the ants will not prematurely stop before reaching to the leaf pages.

2. *Update the accessing time T*

Access time can be calculated as follows:

T = Current Timestamp – latest value of TS

3. *Update the Ant package information*

Add visited page ID, access time, and current timestamp to Ant package entry.

4. *Discover all HTML links*

At this stage, the ants will find all html links connecting to the current page α. A new node ID and the page's URL will thus be inserted in the pheromone table. Let $P(\alpha)$ be the pheromone density value having set to MIN_P if the arrived web page is an internal web page, and MAX_P if the arrived web page is an external web page. MIN_P and MAX_P are constants denoting the lower-bound and upper-bound of possible pheromone density values, respectively. The ant uses the formula below to assign the pheromone value for the new page to be added to the pheromone table.

$$P(\alpha) = MIN\_P \quad , if\ EL(\alpha) = 0$$
$$= MAX\_P \quad , if\ EL(\alpha) = 1$$

where $EL(\alpha)$ is a function which determines whether the arrived page is an external web page by comparing with the IP addresses in the IP address table, i.e.,

$EL(\alpha) = 1$, if page α is not in the IP address table.

$= 0$, if page α is in the IP address table.

This will prevent the ants' traversal to any unexpected external links.

5. *Update pheromone value P(α) in the pheromone table*

The pheromone value of the visited page is updated and kept in the pheromone table following the rules below.

(a) If the current page α is the page that the ants have never visited ($P(\alpha)$ is equal to MIN_P), the ants will find all html links on the current page and update the pheromone $P(\alpha)$ in pheromone table to the negative value of number of outgoing links from that page.

(b) If the current page α is the page that the ants have visited before ($P(\alpha)$ is greater than MIN_P), the ants will increase the pheromone value $P(\alpha)$ by Δ.

ID	Pages α	P(α)
1	161.100.100.0/index.htm	-5
2	161.100.100.0/page2.htm	MIN_P
3	165.100.100.0/page2.htm	MAX_P

Figure 5 Sample data kept in pheromone table.

As the ants arrive, they will update the pheromone value of that page according to the formula shown below.

$$P(\alpha) = P(\alpha) + \Delta \quad , if\ P(\alpha) > MIN\_P$$
$$= DEF\_P(\alpha) \quad , if\ P(\alpha) = MIN\_P$$

where Δ denotes the pheromone increment upon arriving at page α. $DEF\_P(\alpha)$ is the starting value of pheromone at node α. Thus,

$DEF\_P(\alpha)$ = - (number of internal links on page α)

The results are stored in the pheromone table as illustrated in Figure 5.

6. *Determine the destination page β*

The pheromone information kept in the pheromone table is used to determine next destination page to which the ants will move. The ants will choose the destination page β according to the following rules:

- Select page β with the lowest pheromone level $P(\beta)$
- If there are pages that share the same lowest pheromone level $P(\beta)$, select one of those pages arbitrarily.
- If the lowest pheromone reaches MAX_P, the ant will terminate itself.

The proposed Ant algorithm can be procedurally described by the pseudocode below.

```
PRODEDURE TRAVERSAL (WebPage WP)
BEGIN
   IF Ant.Age > 0 AND WP is not Nothing THEN
            MIN = MIN_P, Destination = Nothing
            Timestamps = GetTimeStamp ()
            T = Timestamps – Ant.TS
            UpdateAntPackage (WP, T, Timestamps, Ant.Age-1)
            LINKS = DicoveryLink (WP)
            InsertNewPagesToPheromoneTable (LINKS)
            UpdatePheromoneAtPage (WP)
            FOR EACH I IN LINKS
               IF Pheromone (I) < MIN THEN
                  Clear RandomList
                  MIN = Pheromone (I)
               ELSE
                  IF Pheromone (I) = MIN THEN
                        Clear RandomList
                        MIN = Pheromone (I)
                        Put I in RandomList
                  END IF
               END IF
            END FOR
            IF MIN ≠ MAX_P THEN
               Destination = RANDOM (RandomList)
            END IF
            TRAVERSAL (Destination)
   END IF
END
```

C. *Web navigation routing parameters analysis*

Any arbitrary number of ants can be sent in the web site simultaneously. The ants will stop further exploration after arriving at the page that does not have an html link (hereafter referred to as leaf page). This information will be kept in the Ant package and the reference database for subsequent analysis and future exploration. The number of ants being sent in the web site depends on the increasing rate of new pages found. This can be calculated from the number of pages kept in the pheromone table using the formula given below.

$$INCREASE\_RATE = [(V_t - V_{t-1})/V_{t-1}] * 100$$

where V_t denotes the number of cumulative pages kept in the pheromone table at time t. For example, given there are 10 pages already kept in the pheromone table. If we find two more pages after sending more ants in the web sites, the $INCREASE\_RATE$ will be equal to (12-10)/10 = 20%

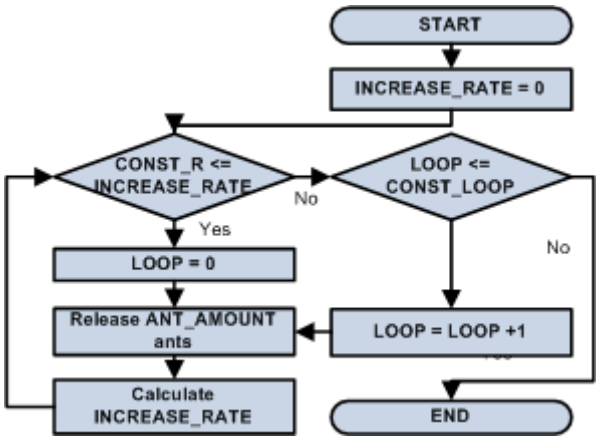

Figure 6 An Ant decision flow chart.

As shown in Figure 6, the algorithm releases *ANT_AMOUNT* ants in the web site in every *INTERVAL_TIME* time, and will stop if the value of number of pages kept in the pheromone table does not change more than *CONST_R* % compared with the previous number of pages found within *CONST_LOOP* times. A number of threshold constants are predefined for simulation propose, namely, *INTERVAL_TIME*, *ANT_AMOUNT*, *CONST_R*, and *CONST_LOOP*.

When *INCREASE_RATE* < *CONST_R* and LOOP > *CONST_LOOP*, the modified ACO algorithm will stop and use the information from the Ant package to generate the adjacent table for use in drawing the web navigation structure diagram.

ANTS	ANT Package
1	S= {1, 2, 3}, T= {0, 1, 1}
2	S= {1, 4, 1, 7, 1, 4, 5, 6},T= {0, 1, 1, 1, 1, 2, 2, 1}

Figure 7 Sample data of the Ant package.

	1	2	3	4	5	6	7
1		(1,1)		(2,3)			(1,1)
2			(1,1)				
3							
4	(1,1)				(1,2)		
5						(1,1)	
6							
7	(1,1)						

Figure 8 Sample inputs of adjacent table for Ant 1 (bold borders) and Ant 2 (light borders).

Figure 7 illustrates samples of information kept in the Ant package after 2 ants have been sent in the sample web site. The visited pages set *(S)* and used time set *(T)* in the Ant package will be added to the adjacent table as a pairwise a_{ij} of the form (N_{ij}, T_{ij}). The first element N_{ij} represents the number of visited ants leaving from page *i* for page *j* and the second element T_{ij} is the cumulative time used by the ants to move from page *i* to page *j*. Figure 8 demonstrates the values kept in the adjacent table after the first ant and the second ant have been sent in the web site, respectively.

D. Site map creation and analysis

We can simply draw the web navigation diagram from the data kept in the adjacent table as shown in Figure 9. Each number shown on each connecting edge is derived from the second element T_{ij} of the adjacent pair divided by the first element N_{ij}. This number denotes the average used time that the ants navigate from page *i* to page *j*. For example, the number 1.5 shown on the connecting edge between page 1 and page 4 represents the average used time (in seconds) that the ants move from page 1 to page 4.

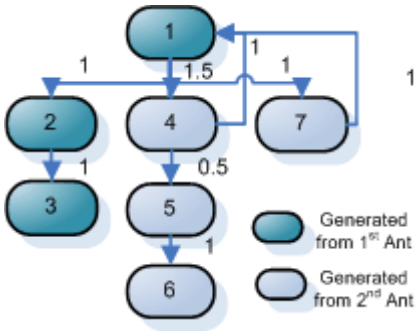

Figure 9 A web navigation diagram resulting from Ant package.

All of these data are stored in the database use for subsequent comparison and reference. We can periodically keep track of web update on a daily, weekly, or monthly basis. The results will help detect additional links, dead links, and unreachable links in the web site to be explained subsequently.

	1	2	3	4	5	6	7	8
1		(1,1)		(1,2)			(1,1)	
2			(1,1)					
3								
4								
5								
6								
7	(1,1)							(1,1)
8								

Figure 10 Samples inputs of adjacent table after web pages are being added or removed.

Upon completion of site data and site map creation, analysis of web structure navigation begins. The adjacent table is used as a baseline to compare with the current web navigation structure. In the real world, as the web structure changes everyday, many pages are added or removed. The proposed method in this section will serve to identify those pages. Figure 10 shows sample data in the adjacent table retrieved from the ants that have been sent in the system on various occasions, where some pages have been added or removed. The page ID and IP address kept in the original IP address table have been reused, so each page found by the ants can refer to existing page ID contained in the pheromone table from the previous visit. This simplifies identification of pages to be removed or added. In this example, the link between pages 1 and 4 has been changed to point to wrong URL that results in the "page not found" error and blocks all following web page accesses. Consequently, pages 5 and 6 become unreachable. Figure 11 depicts the new web navigation diagram based on data from the new adjacent table.

By comparing updated data in the adjacent table with the original one, we can determine the added pages by locating all new columns being added to the IP address table from the set operation below.

$$S_{\text{pages added}} = S_{\text{actual}} - S_{\text{baseline}}$$

where S_{actual} is a set of all page IDs kept the current adjacent table, and S_{baseline} is a set of all baseline page ID in the baseline adjacent table. The baseline's adjacent table can be chosen from a specific point of time from the reference database. For this example, page "8" becomes the newly added page to the web site based on the above set operation.

$$S_{\text{pages added}} = \{1, 2, 3, 4, 7, 8\} - \{1, 2, 3, 4, 5, 6, 7\} = \{8\}$$

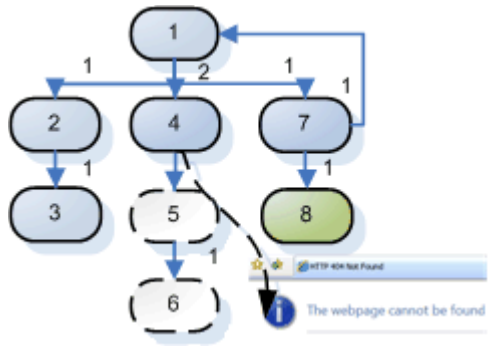

Figure 11 A web navigation error found after changing the web structure.

	1	2	3	4	5	6	7	8
1		(1,1)		(1,2)			(1,1)	
2			(1,1)					
3								
4								
5								
6								
7	(1,1)							(1,1)
8								

Figure 12 Sample methods to identify the unreachable pages.

To identify the removed pages, dead links, and the pages that might be the cause of other unreachable pages, a simple mapping technique is used. From Figure 12, the highlighted cells represent all dead links found in the new web structure. All navigation information is lost compared with the original one. The dead link set has been described as follows:

$$S_{\text{dead links}} = \{link_{ij}, link_{ij} \text{ is a link from page } i \text{ to page } j \mid a_{ij} \in \text{baseline adjacent table} \wedge a_{ij} \notin \text{current adjacent table}\}$$

The unreachable pages can be determined by the columns in the adjacent table that do not contain any data of adjacent pairs.

$$S_{\text{unreachable pages}} = \{j \mid \forall i, a_{ij} = \varnothing\}$$

As shown in Figure 12, the columns of page 5 and 6 do not contain any information for web navigation. Thus, they are unreachable pages. In addition, the cause of unreachable page can be traced by locating candidate rows whose navigation information is missing from the cells intersecting with the columns that have already been identified to be the unreachable pages. Therefore,

$$S_{\text{unreachable causal pages}} = S_{\text{data missing rows}} - S_{\text{unreachable pages}}$$

where $S_{\text{data missing rows}} = \{i \mid link_{ij} \in S_{\text{dead links}}\}$. Based on the above example, the unreachable causal page becomes $\{4,5\} - \{5,6\} = \{4\}$. The results forewarn the webmaster to first investigate these pages as potential culprits for all the errors incurred.

Access performance can be measured by comparing the weight of the connecting edges with the original value from the web navigation baseline. In Figure 12, there is only one connection edge from page 1 to page 4 whose weight value differs from that of the original structure, thereby access time is reduced by 34%. Other statistics such as external page links can be determined from the pages that have the pheromone value equal to *MAX_P* in the pheromone table. Analysis results identify all changing attributes of the new web navigation structure as summarized below.

Attributes	Analysis Result
Number of pages/links	5/5
Number of new additional pages	1 (page 8)
Number of new additional links	1 (Link from 7 to 8)
Number of removal pages	2 (page 5, 6)
The pages that might be the cause of unreachable pages	4
Number of removal/dead links	3 (Links from 4 to 1, 4 to 5, and 5 to 6)
Number of unreachable pages	2 (page 5, 6)
Performance compared with the original structure	Access time of page 1 to 4 is reduced by 34%
Number of external pages	1
Number of leaf pages	3 (3, 4, 8)
The highest content load time	2 seconds (from 1 to 4)
The lowest content load time	1 second

IV. Experiment Results

From the experimental results, we have sent the ants in a number of web sites, in particular, our own which contains 50 web pages and 69 navigation links. Navigation structure of the web site, along with all prerequisite constants, is depicted in Figure 13 and Figure 14, respectively. All web pages have been discovered after only 15 ants were deployed. Nonetheless, the Ant system still sent additional 10 ants to ensure that no more pages left uncovered.

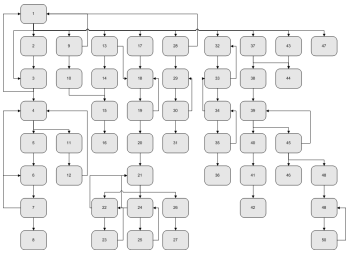

Figure 13 Web site structure.

Figure 15 illustrates the number of pages found in relation to the numbers of ants being deployed. Various test scenarios were conducted to verify and gauge the performance of the

proposed algorithm, namely, non-existing references, dead links, newly added links, and unreachable links. Table entry creation, deletion, and update were also exercised to reaffirm that correct information was maintained.

Constant	Assigned value
MIN_P	-1,000
MAX_P	1,000
INTERVAL_TIME	30 Seconds
ANT_AMOUNT	1
CONST_R	1
CONST_LOOP	10
CONST_AGE	50
Δ	1

Figure 14 Predefined constants.

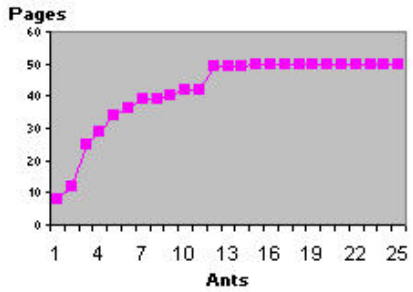

Figure 15 The number of pages found VS the numbers of ants sent.

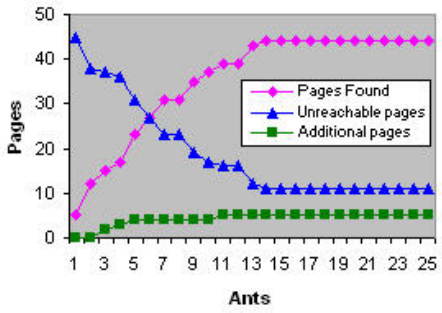

Figure 16 The number of pages found, unreachable pages, and additional pages after modifying web site structure.

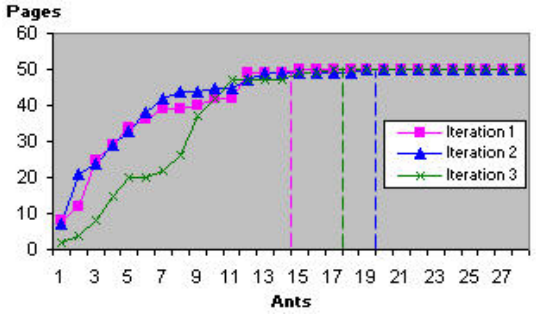

Figure 17 Different number of ants deployed in separate iterations.

The following statistics were complied as the results of the experiment depicted in Figure 16, namely, 44 Web pages discovered by 14 ants, 3 of the above 44 pages were possible culprits of other 7 unreachable pages, 5 newly added pages, and 11 unreachable pages. The number of ants sent in the system to reach all available pages will depend on the random decision by the ants as to which page that has the same lowest pheromone to be chosen. Irrespective of the page chosen, the outcome remains the same upon completion of executing the Ant algorithm. Figure 17 compares different number of ants sent by the Ant algorithm on the same web site, yielding the same result. The Ant algorithm found all the pages contained in the web site after 15, 20, 18 ants being sent in the system for trial number 1, 2, 3, respectively.

Further analyses of the depth-first-search (DFS) and breadth-first-search (BFS) of the original ACO search unveiled infinite looping caused by acyclic references among web pages. This problem was resolved by the use of ant's Age in the modified ACO algorithm (ANT). As a result, many ants could be sent simultaneously to cover the designated web site so as to complete the exploration faster without infinite looping as oppose to the original approach which was executed serially. Some comparative statistics are given below.

	DFS	BFS	ANT
Infinite looping	Yes	Yes	No
Memory resource	Linear O(bm+1)	Exponential $O(b^{d+1})$	Constant O(*Age*)
Concurrency	No	No	Yes

b = branching, d = depth/path length, m = maximum depth

V. Conclusions

This paper proposes a modified Ant Colony Optimization algorithm to analyze web navigation structure and performance monitoring. Implementation of the proposed algorithm lends itself to be a tool identifying dead links, unreachable pages, and new additional pages which result from regular updates. As the system adaptively operates without human intervention, it alleviates daily chores and routine work, whereby increasing the reliability of the web site. Further studies on seemingly recalcitrant issues such as autonomous intelligent web crawler agents within and beyond physical and logical web site boundaries, resolution of acyclic exploration on internal/external references, and minimal agents used will be conducive toward performance improvement of the proposed approach.

References

[1] Alberto Montresor, Hein Meling and Ozalp Babaoglu, "Messor: Load-Balancing through a Swam of Autonomous Agents", Proceedings of the 1st International Workshop on Agents and Peer-to-Peer Computing, Bologna, Italy, July 2002.

[2] Dorigo M., Maniezzo, V., Colorni, A., "The Ant System: Optimization by a Colony of Cooperating Agents", IEEE Transactions on Systems, Man, and Cybernetics-Part B, Vol. 26, No.1, pp.29-41, 1996.

[3] Li, H., Lam, C. P., "Software Test Data Generation using Ant Colony Optimization", International Conference of Computational Intelligence, pp. 1–4, 2004.

[4] B.Golden, W.Stewart, "Empiric analysis of heuristics", in *The Travelling Salesman Problem*, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy-Kan, D. B. Shmoys eds., New York:Wiley, 1985.

[5] Wagner, I. A., Lindenbaum, M., Bruckstein, A. M., "ANTS: Agents, Networks, Trees, and Subgraphs", Special issue on Ant Colony Optimization (M. Dorigo, G. Di Caro, T.Stützle (eds)), *Future Generation Computer Systems*, Vol. 16, No. 8, pp. 915-926, North Holland, June 2000.

[6] Gordon Moyer, "The Origin of the Julian Day System", Sky and Telescope 61 (April 1981) 311−313.