การนำทางและการจำลองเว็บด้วยแอนต์โคโลนีออปติไมเซชัน

นายเอกชัย จินต์หิรัณย์กุล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์ ภาควิชาคณิตศาสตร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2551

WEB NAVIGATION ANALYSIS AND SIMULATION USING ANT COLONY OPTIMIZATION

Mr. Ekachai Jinhirunkul

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Computer Science

Department of Mathematics

Faculty of Science

Chulalongkorn University

Academic Year 2008

Thesis Title	WEB NAVIGATION ANALYSIS AND SIMULATION USING ANT COLONY OPTIMIZATION
By	Mr. Ekachai Jinhirunkul
Field of Study	Computer Science
Advisor	Assistant Professor Peraphon Sophatsathit, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

…….. Dean of the Faculty of Science

(Professor Supot Hannongbua, Ph.D.)

THESIS COMMITTEE

…….. Chairman

(Professor Chidchanok Lursinsap, Ph.D.)

…….. Advisor

(Associate Professor Peraphon Sophatsathit, Ph.D.)

…….. External Examiner

(Associate Professor Arit Thammano, Ph.D.)

เอกชัย จินต์หิรัณย์กุล : การนำทางและการจำลองเว็บด้วยแอนต์โคโลนีออปติไมเซชัน. (WEB NAVIGATION ANALYSIS AND SIMULATION USING ANT COLONY OPTIMIZATION) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ. ดร. พีระพนธ์ โสพัศสถิตย์, 52 หน้า.

การวิจัยนี้ได้นำ Ant Colony Optimization อัลกอริทึมมาใช้ประโยชน์ในการสำรวจ และจำลองโครงสร้างเว็บไซต์ รวมไปถึงการตรวจสอบลำดับการเข้าถึงของหน้าหน้าเว็บไซต์ซึ่งเป็นการทำให้ได้มาของข้อมูลในการเข้าถึง และ ประสิทธิภาพของเว็บไซต์ การสำรวจนี้ยังสามารถทำให้ทราบถึงการเปลี่ยนแปลงโครงสร้างของเว็บไซต์ เช่น การเชื่อมโยงของหน้าเวบไซต์ที่เกิดขึ้นใหม่, การเชื่อมโยงของหน้าเว็บไซต์ที่ถูกนำออก และ การเชื่อมโยงของเว็บไซต์ที่ไม่สามารถเข้าถึงได้ ผลลัพธ์จากการสำรวจนี้ยังทำให้ทราบถึงหน้าเว็บไซต์ที่สามารถเข้าถึงได้ทั้งหมด และ ข้อมูลทางสถิติของประสิทธิภาพในการเข้าถึงหน้าเว็บไซต์ เพื่อช่วยในการปรับปรุงคุณภาพของเวบไซต์ต่อไป ข้อดีของความไม่ซับซ้อนในระเบียบวิธี Ant Colony Optimization ทำให้สามารถใช้เทคนิคที่ง่ายๆ ในการแปลความหมายของข้อมูลให้เป็นข้อมูลที่มีประโยชน์ในการปรับปรุงคุณภาพของเว็บไซต์ และ ประโยชน์เชิงพาณิชย์ต่อไป

ภาควิชา คณิตศาสตร์ ลายมือชื่อนิสิต.....

สาขาวิชา วิทยาการคอมพิวเตอร์ ลายมือชื่ออ.ที่ปรึกษาวิทยานิพนธ์หลัก.....

ปีการศึกษา 2551

507 36228 23 : MAJOR COMPUTER SCIENCE

KEYWORDS : WEB NAVIGATION / WEB STRUCTURE ANALYSIS / WEB AGENT / ANT COLONY OPTIMIZATION

EKACHAI JINHIRUNKUL : WEB NAVIGATION ANALYSIS AND SIMULATION USING ANT COLONY OPTIMIZATION. ADVISOR : ASST. PROF. PERAPHON SOPHATSATHIT, Ph.D., 52 pp.

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.

Department : Mathematics Student's Signature .....

Field of Study : Computer science Advisor's Signature .....

Academic Year : 2008

ACKNOWLEDGEMENTS

I would like to acknowledge my advisor, Associate Professor Peraphon Sophatsathit, at The Advanced Virtual and Intelligent Computing (AVIC) Research Center, for all his great support and patience that tremendously helped me accomplish this thesis. He also suggests the solution for solving many problems occurred while doing an experiment. I would also like to thank all my friends, and most importantly, my father, mother and sister for everything that they have supported me.

May I dedicate this work to all the people as I mentioned above. Without them, this work will never be done. Throughout this thesis, I had encountered many problems, but they were trivial, compared to all the supports given by these people. The encouragement from them was so great and those impressions will always be remembered.

CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

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.

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. 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 so doing, 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.

Studies [2] 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.

1.1 Problem Identification

From the new website technologies describe above, numerous problems arise in the development of web application. Some of the problems are

1. Web sites structure and Web navigation change everyday and need more effort and maintenance cost;
2. Historical changes of web site have not been kept and hard to reference in the future;
3. A lot of pages have been added, modified or removed without verification from administrators;
4. The number of unexpected unreachable pages increases;
5. It is hard to find the root cause of all unreachable pages;
6. Performance is reduced; and
7. More staffs are needed to maintain the web site.

1.2 Research Objectives

In order to address and solve some of the above problem, an algorithm is proposed to investigate and analyze web site structure that will serve the following aspects:

1. Explore any designated web site, mapping its structure and navigation routing;
2. Unveil changing structure of the web site such as new links, removed links, and unreachable links; and
3. Obtain accessibility and performance information of the website.

1.3 Scope

Due to the dynamicity and the sheer volume of the World Wide Web, the propose reach will confine scope of study within the following domain of application:

1. The websites under researched are not prevented by user's authentication page.
2. Researchers initially do the research and experiment on a limited website environment. Extended studies on web page and web site will be focused in future work.
3. Web search is limited to internal links only. All external references are considered unreachable.

1.4 Research methodology

In order to achieve the defined objectives above, the following tasks will be stated by means of appropriate theoretical work described below:

1. Investigate related technologies.
   - Study Website technology, advantage and disadvantage of Web2.0 compared with Web 1.0
   - Study basic concept of Swam intelligence
   - Research and study the Ant Colony Optimization (ACO) algorithm
   - Research the concept of sitemap
2. Devise the solutions to the research objective
3. Establish an algorithm to construct the web site analysis model
4. Conduct an experiment to verify the viability of the proposed model
5. Write the thesis

Below is a time table covered all of the above tasks.

Table 1.1 : Research methodology time table

No	Tasks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	Investigate related technologies																		
2	Devise the solutions to the research objectives																		
3	An algorithm to construct the web site analysis model																		
4	Conduct an experiment to verify the viability of the proposed model																		
5	Write the thesis																		

1.5 Benefits

The proposed approach will permit automated website investigation, analysis, unveil changing web site structure, and measure the web site navigation performance.

CHAPTER II

THEORETICAL BACKGROUND

2.1 Literature review

2.1.1 Ant Colony Optimization

Marco Dorigo and Christian [2] introduced Ant Colony Optimization in the early 1990s. It is as a nature-inspired metaheuristic for the solution of hard combinatorial optimization (CO) problems. This algorithm used to obtain good enough solutions to hard CO problems in a reasonable amount of computation time. The inspiring source of ACO is the foraging behavior of real ants. When searching for food, ants initially explore the area surrounding their nest in a random manner. As soon as an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. During the return trip, the ant deposits a chemical pheromone trail on the ground. The quantity of pheromone deposited, which may depend on the quantity and quality of the food, will guide other ants to the food source. This characteristic of real ant colonies is exploited in artificial ant colonies in order to solve CO problems. In general, the ACO approach attempts to solve an optimization problem by repeating the following two steps:

- Candidate solutions are constructed using pheromone model
- The candidate solutions are used to modify the pheromone values in a way that is deemed to bias future sampling toward high quality solutions.

2.1.2 Messor load balancing

Alberto Montresor, Hein Meling, and Ozalp Babao glu [1] presented Messor application that used for control P2P Load-Balancing using a Swarm of Autonomous Agents. Messor is a grid computing system aimed at supporting the concurrent execution of highly-parallel, time-intensive computations, in which the workload may be

decomposed into a large number of independent jobs. Figure 2.1 illustrates the architecture of the Messor system. A Messor system is composed of a collection of interconnected Anthill nests configured to run the Messor software. Every such nest can submit jobs to the nest network, where each job is composed of some input data and the algorithm to be computed over these data. Messor offers a very simple API to its users, enabling them to submit new jobs to be computed and collecting results once the jobs have been computed. The originator nest of a job is the nest where the job has been submitted. Once submitted, jobs may remain in the originator, or may be transferred to other nests in order to exploit their unused computational power. When a job is completed, the result is sent back to the originator. Once there, the user is either notified of the job result, or the result is locally stored.

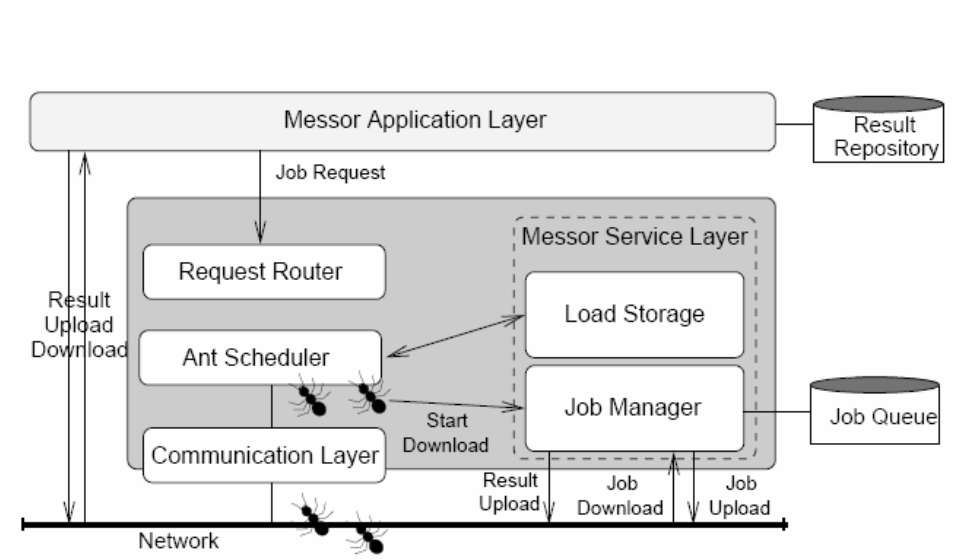

Figure 2.1 : Example of Messor Architecture

2.1.3 Graph searching with Ant

Israel A. Wagner, Michael Lindenbaum and Alfred M. Bruckstein [5][6][8] described the solution to efficient exploration of large networks using ACO. They presented the Vertex Ant Walk that is a trace oriented process to cover a dynamic graph in a bounded time which is depicted in Figure 2.2. This concept utilizes ACO to be used on graph, subgraphs, trees and network searching.

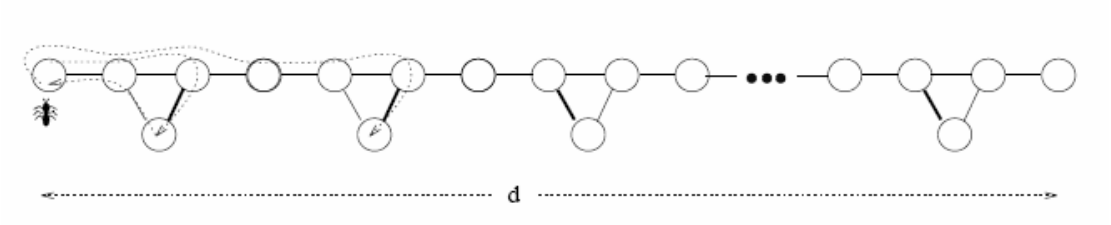

Figure 2.2 : Example of using Ant Colony Optimization to search on graph.

2.1.4 Software Test Data Generation using Ant Colony Optimization

Huaizhong Li and C. Peng Lam [3][4][7] presented the Software Test Data Generation using Ant Colony Optimization. They used ACO as a supplementary optimization stage for finding sequences of transitional statements in generating test data for evolutionary testing. This proposal uses UML Statechart diagrams and ACO for test data generation. The advantages of the proposed approach are:

1) the approach directly uses the standard UML artifacts created in software design processes; and

2) the automatically generated test sequence is always feasible, non-redundant and achieves the all state coverage criterion.

A group of ants can effectively explore the graph and generate optimal test data to achieve test coverage requirement, as illustrated in Figure 2.3

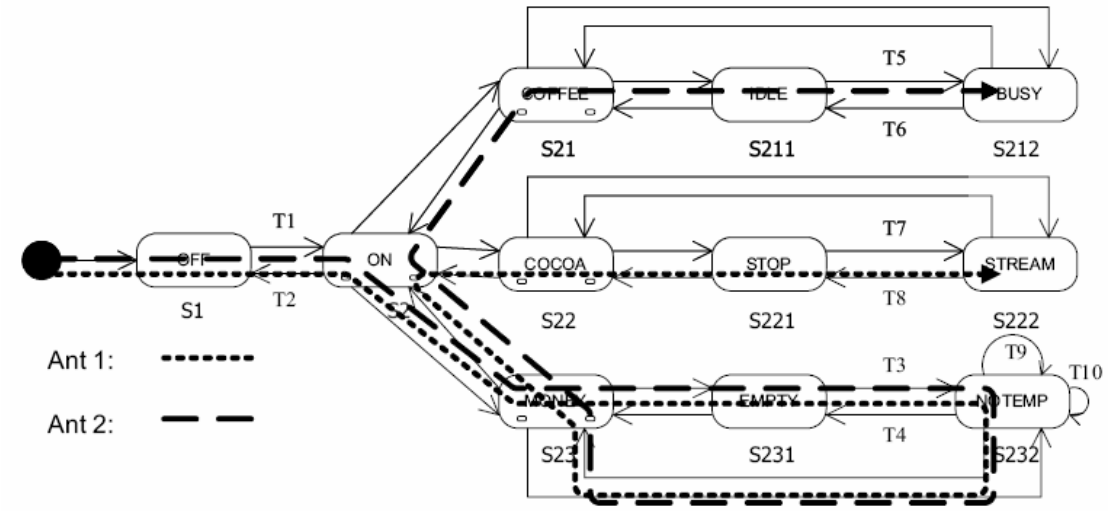

Figure 2.3 : Example of using Ant Colony Optimization to generate the test case data.

2.2 Website technology

2.2.1 Website

A **website** [12] (alternatively, **web site** or **Web site**, from the proper noun World Wide Web) is a collection of Web pages, images, videos or other digital assets that are hosted on one or more web servers, usually accessible via the Internet. All publicly accessible websites on the Internet are seen collectively as constituting the "World Wide Web."

Websites are written in, or dynamically converted to, HTML (HyperText Markup Language) and are accessed using a software interface classified as user agent. Web pages can be viewed or otherwise accessed from a range of computer-based and Internet-enabled devices of various sizes, including desktop computers, laptops, PDAs and, cell phones.

A website is hosted on a computer system known as a web server, also called an HTTP server. These terms can also refer to the software that runs on these systems that retrieves and delivers the Web pages in response to requests from the website users.

Some websites require a subscription to access some or all of their contents. Examples of subscription sites include many business sites, news sites, academic journal sites, gaming sites, message boards, Web-based e-mail, services, social networking websites, and sites providing real-time stock market data. Because they require authentication to view the content, they are technically an Intranet site.

There are several categories of websites, for example,

- a personal website
- a commercial website
- a government website
- a non-profit organization website

Any website can contain a hyperlink to any other websites, so the distinction between individual sites, as perceived by the user, may sometimes be blurred.

A Web page is a document, typically written in (X)HTML, that is almost always accessible via HTTP which is a protocol used to transfers information from the Web server to be displayed in the user's Web browser.

The pages of a website can usually be accessed from a common root URL called the homepage which resides on the same physical server. The URLs of all pages are structurally organized in a hierarchy. Nevertheless, the hyperlinks between them control how the reader perceives the overall structure and how the traffic flows between the different parts of the site.

Static content of a web page may also be dynamically generated either periodically, or if certain conditions for regeneration occur (cached) in order to avoid the performance loss of initiating the dynamic engine on a per-user or per-connection basis.

2.2.2 Static and Dynamic websites

Static Website

A Static Website is one that has web pages stored on the server in the same form as the user will view them. It is primarily coded in HTML (Hypertext Markup Language).

A static website is also called a **Classic website**, a **5-page website** or a **Brochure website** because it simply presents pre-defined information to the user. It may include information about a company and its products and services via text, photos, flash animation, audio/video, and interactive menus and navigation.

Static website usually displays the same information to all visitors, thus the information is static. Similar to handing out a printed brochure to customers or clients, a static website will generally provide consistent, standard information for an extended period of time. Although the website owner may make updates periodically, it is a

manual process to edit the text, photos, and other contents and may require basic website design skills and software.

Visitors are not able to control what information they receive via a static website, and must instead settle for whatever content the website owner has decided to offer at that time.

Dynamic website

A Dynamic Website is one that does not have web pages stored on the server in the same form as the user will view them. Instead, the web page content changes automatically and/or frequently based on certain criteria. It generally collates information on the hop each time a page is requested.

A dynamic website also describes its construction or how it is built, and more specifically refers to the code used to create a single web page. A **Dynamic Web Page** is generated on the fly by piecing together certain blocks of code, procedures or routines. A dynamically-generated web page would call various bits of information from a database and put them together in a pre-defined format to present the reader with a coherent page. It interacts with users in a variety of ways including by reading cookies recognizing users' previous history, session variables, server side variables etc., or by using direct interaction (form elements, mouseovers, etc.) A site can display the current state of a dialogue between users, monitor a changing situation, or provide information in some way personalized to the requirements of the individual user.

The main purpose behind a dynamic site is that it is much simpler to maintain a few web pages plus a database than it is to build and update hundreds or thousands of individual web pages and links. In one way, a data-driven website is similar to a static site because the information that is presented on the site is still limited to what the website owner has allowed to be stored in the database (data entered by the owner and/or input by users and approved by the owner). The advantage is that there is usually more information stored in a database and made available to users.

This type of website usually displays different information depending on the visitor, thus the information is dynamic. Similar to talking to a customer service representative on the telephone, a dynamic website will provide personalized, real-time information and take the appropriate action intended to serve the customer's needs immediately. The website usually requires advanced programming and a database. It often includes admin tools for the website owner to update the website content frequently.

Visitors are able to control what information they wish to receive via a dynamic website, instead of settling for only static content that the website owner has decided to offer. In addition, a visitor may be able to manipulate the content of the website and perform a multitude of tasks.

2.2.3 Web 1.0 and Web 2.0

Web 1.0 is a retronym which refers to the state of the World Wide Web and any website design style used before the advent of the Web 2.0 phenomenon. It is the general term that has been created to describe the Web before the 'bursting of the dot-com bubble' in 2001, which is seen by many as a turning point for the internet.

Some typical design elements of a **Web 1.0** site include:

- Static pages instead of dynamic user-generated content.
- The use of framesets.
- Proprietary HTML extensions such as the <blink> and <marquee> tags introduced during the first browser war.
- Online guestbooks.
- GIF buttons, typically 88x31 pixels in size promoting web browsers and other products.
- HTML forms sent via email. A user would fill in a form, and upon clicking submit their email client would attempt to send an email containing the form's details.

The shift from **Web 1.0** to Web 2.0 can be seen as a result of technological refinements, which included such adaptations as "broadband, improved browsers, and Ajax, to the rise of Flash application platforms and the mass development of wigetization, such as Flickr and YouTube badges".

As well as such adjustments to the internet, the shift from **Web 1.0** to Web 2.0 is a direct result of the change in the behaviour of those who use the World Wide Web. Web 1.0 trends included worries over privacy concerns resulting in a one-way flow of information, through websites which contained 'read-only' material. Widespread computer illiteracy and slow internet connections added to the restrictions of the internet, which characterised **Web 1.0**. Now, during Web 2.0, the use of the Web can be characterised as the decentralisation of website content, which is now generated from the 'bottom-up', with many users being contributors and producers of information, as well as the traditional consumers.

Web 2.0

Web 2.0 [13] describes the changing trends in the use of World Wide Web technology and web design that aim to enhance creativity, communications, secure information sharing, collaboration and functionality of the web. Web 2.0 concepts have led to the development and evolution of web culture communities and hosted services, such as social-networking sites, video sharing sites, wikis, blogs, and folksonomies. The term became notable after the first O'Reilly Media Web 2.0 conference in 2004. Although the term suggests a new version of the World Wide Web, it does not refer to an update to any technical specifications, but rather to changes in the ways software developers and end-users utilize the Web.

Web 2.0 websites allow users to do more than just retrieve information. They can build on the interactive facilities of "Web 1.0" to provide "Network as platform" computing, allowing users to run software-applications entirely through a browser. Users can own the data on a Web 2.0 site and exercise control over that data. These sites may have an "Architecture of participation" that encourages users to add value to the application as they use it. This stands in contrast to very old traditional websites, the sort

which limited visitors to viewing and whose content only the site's owner could modify. Web 2.0 sites often feature a rich, user-friendly interface based on Ajax, OpenLaszlo, Flex or similar rich media. Figure 2.4 exemplifies some commercial values obtained from Web 2.0.

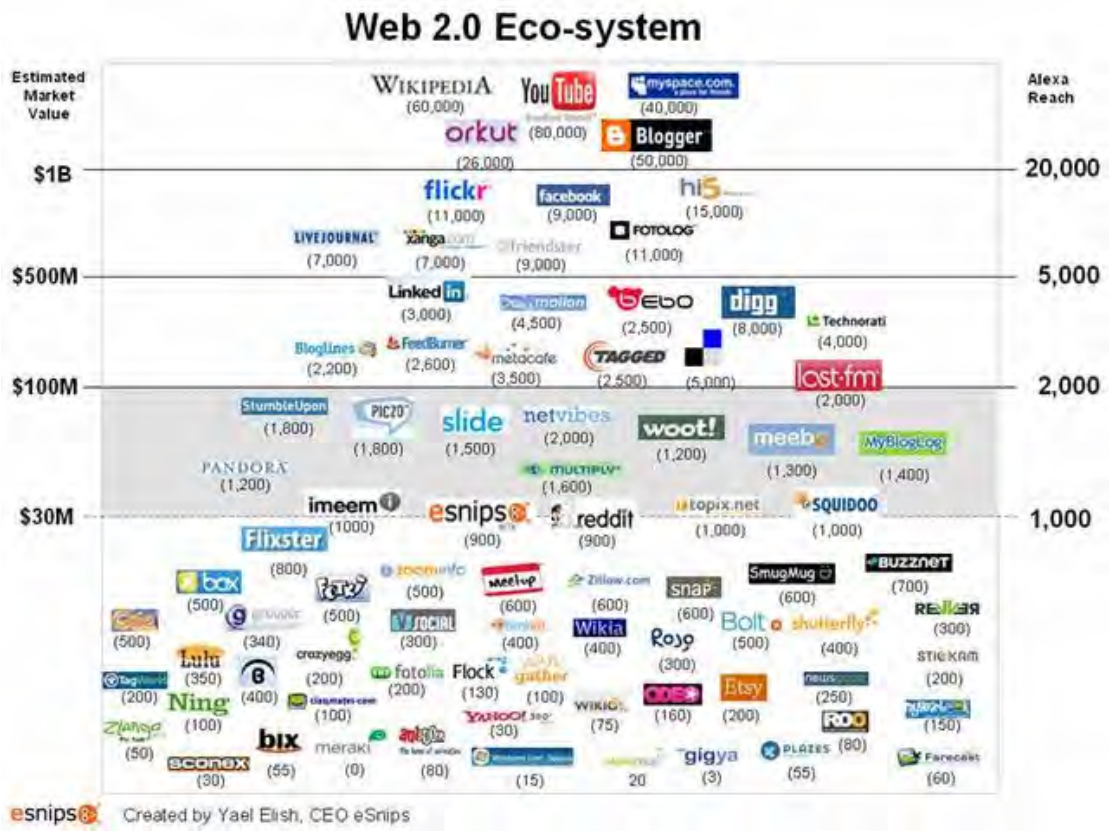

Figure 2.4 : The market value earned from Web 2.0 technology

The characteristics of Web 2.0 are rich user experience, user participation, dynamic content, metadata, web standards and scalability. Further characteristics, such as openness, freedom and collective intelligence by way of user participation, can also be viewed as essential attributes of Web 2.0.

Some typical design elements of a **Web 2.0** site include:

- Search: the ease of finding information through keyword search which makes the platform valuable.
- Links: guides to important pieces of information. The best pages are the most frequently linked to.

- Authoring: the ability to create constantly updating content over a platform that is shifted from being the creation of a few to being the constantly updated, interlinked work. In wikis, the content is iterative in the sense that the people undo and redo each other's work. While in blogs is cumulative that posts and comments of individuals are accumulated over time.

- Tags: categorization of content by creating tags that are simple, one-word descriptions to facilitate searching and avoid rigid, pre-made categories.

- Extensions: automation some of the work and pattern matching by using algorithms e.g. amazon.com recommendations.

- Signals: the use of RSS (Really Simple Syndication) technology to notify users with any changes of the content by sending e-mails to them."

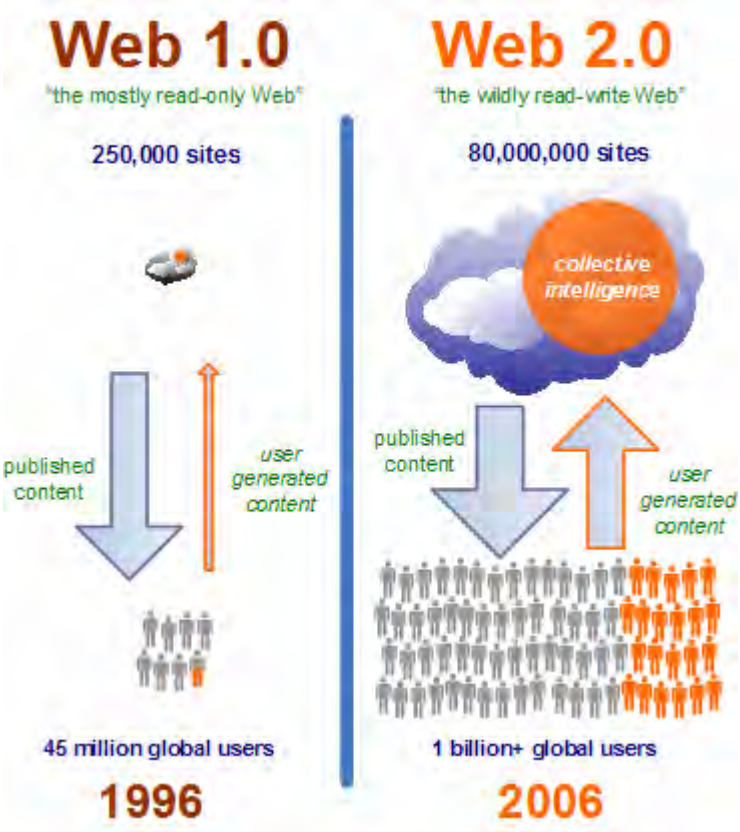

Figure 2.5 : Comparison of Web 1.0 and Web 2.0 usage

2.3 Site map

A **site map** (or **sitemap**) [11][14] is a representation of the architecture of a web site. It can be either a document in any form used as a planning tool for web design, or a web page that lists the pages on a web site, typically organized in hierarchical fashion. This helps visitors and search engine bots find pages on the site.

While some developers argue that **site index** is a more appropriately used term to relay page function, web visitors are used to seeing each term and generally associate both as one and the same. However, a site index is often used to mean an A-Z index that provides access to particular content, while a site map provides a general top-down view of the overall site contents.

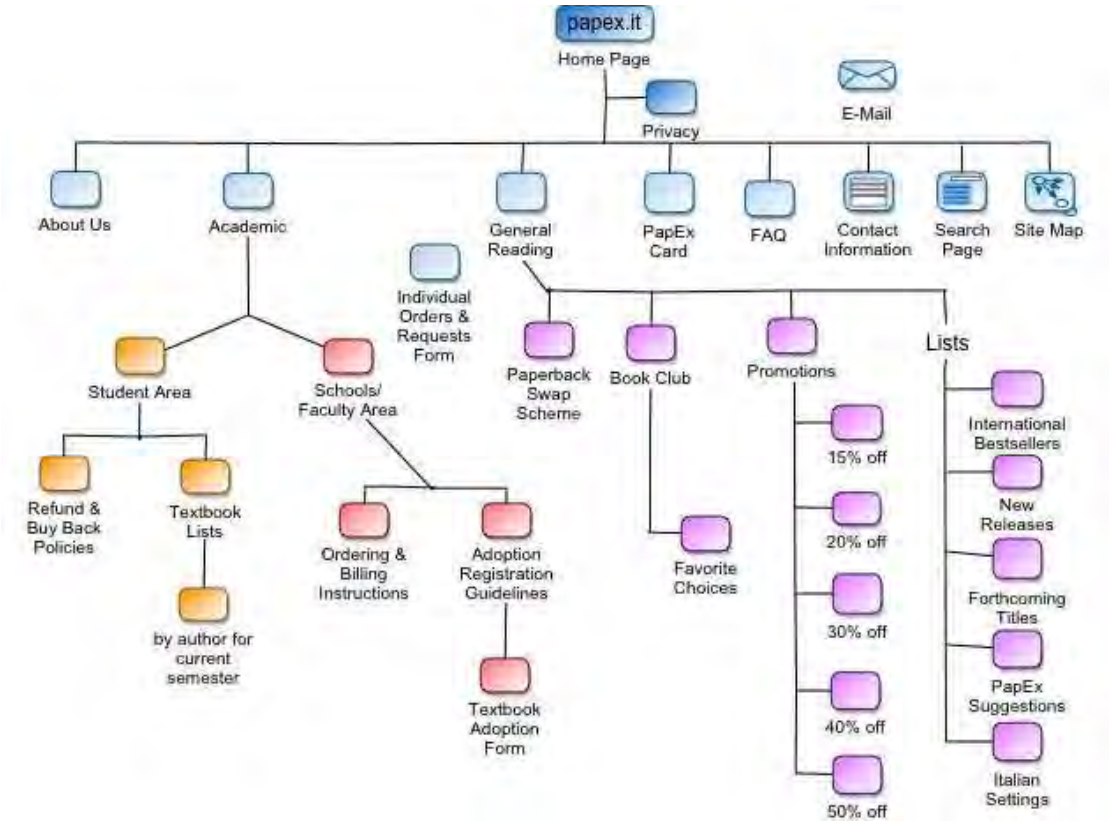

Figure 2.6 : Example of sitemap

Site maps can improve search engine optimization of a site by making sure that all the pages can be found. This is especially important if a site uses Adobe Flash or JavaScript menus that do not include HTML links.

Most search engines will only follow a finite number of links from a page, so if a site is very large, the site map may be required so that search engines and visitors can access all content on the site. The basic premise is that some sites have a large number of dynamic pages that are only available through the use of forms and user entries. The sitemap files can then be used to indicate to a web crawler how such pages can be found. Google, MSN, Yahoo and Ask now jointly support the Sitemaps protocol.

2.4 Swarm Intelligence

Swarm intelligence (SI) [9] is an artificial intelligence based on the collective behavior of decentralized, self-organized systems. The expression was introduced by Gerardo Beni and Jing Wang in 1989 [16], in the context of cellular robotic systems.

SI systems are typically made up of a population of simple agents interacting locally with one another and with their environment. The agents follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local interactions between such agents lead to the emergence of complex global behavior. Natural examples of SI include ant colonies, bird flocking, animal herding, bacterial growth, and fish schooling.

Swarm intelligence has a marked multidisciplinary character since systems with the above mentioned characteristics can be observed in a variety of domains. Research in swarm intelligence can be classified according to different criteria.

Natural vs. Artificial: It is customary to divide swarm intelligence research into two areas according to the nature of the systems under analysis. We speak therefore of *natural* swarm intelligence research, where biological systems are studied; and of *artificial* swarm intelligence, where human artifacts are studied.

Scientific vs. Engineering: An alternative and somehow more informative classification of swarm intelligence research can be given based on the goals that are pursued: we can identify a *scientific* and an *engineering* stream. The goal of the scientific stream is to model swarm intelligence systems and to single out and understand the mechanisms that allow a system as a whole to behave in a coordinated way as a result of local individual-individual and individual-

environment interactions. On the other hand, the goal of the engineering stream is to exploit the understanding developed by the scientific stream in order to design systems that are able to solve problems of practical relevance.

The two dichotomies natural/artificial and scientific/engineering are orthogonal: although the typical scientific investigation concerns natural systems and the typical engineering application concerns the development of an artificial system, a number of swarm intelligence studies have been performed with swarms of robots for validating mathematical models of biological systems. These studies are of a merely speculative nature and definitely belong in the scientific stream of swarm intelligence. On the other hand, one could influence or modify the behavior of the individuals in a biological swarm so that a new swarm-level behavior emerges that is somehow functional to the solution of some task of practical interest. In this case, although the system at hand is a natural one, the goals pursued are definitely those of an engineering application. In the following, an example is given for each of the four possible cases.

Natural/Scientific: Foraging Behavior of Ants

In a now classic experiment done in 1990, Deneubourg and his group showed that, when given the choice between two paths of different length joining the nest to a food source, a colony of ants has a high probability to collectively choose the shorter one. Deneubourg has shown that this behavior can be explained via a simple probabilistic model in which each ant decides where to go by taking random decisions based on the intensity of pheromone perceived on the ground, the pheromone being deposited by the ants while moving from the nest to the food source and back.

Artificial/Scientific: Clustering by a Swarm of Robots

Several ant species cluster corpses to form cemeteries. Deneubourg et al. (1991) were among the first to propose a distributed probabilistic model to explain this clustering behavior. In their model, ants pick up and drop items with probabilities that depend on information on corpse density which is locally available to the ants. Beckers et al. (1994) have programmed a group of robots to implement a similar clustering

behavior demonstrating in this way one of the first swarm intelligence scientific oriented studies in which artificial agents were used.

Natural/Engineering: Exploitation of collective behaviors of animal societies

A possible development of swarm intelligence is the controlled exploitation of the collective behavior of animal societies. No example is available in this area of swarm intelligence although some promising research is currently in progress: For example, in the Leurre project, small insect-like robots are used as lures to influence the behavior of a group of cockroaches. The technology developed within this project could be applied to various domains including agriculture and cattle breeding.

Artificial/Engineering: Swarm-based Data Analysis

Engineers have used the models of the clustering behavior of ants as an inspiration for designing data mining algorithms. A seminal work in this direction was undertaken by Lumer and Faieta in 1994. They defined an artificial environment in which artificial ants pick up and drop data items with probabilities that are governed by the similarities of other data items already present in their neighborhood. The same algorithm has also been used for solving combinatorial optimization problems reformulated as clustering problems (Bonabeau et al. 1999).

Properties of a Swarm Intelligence System

A typical swarm intelligence system has the following properties:

- it is composed of many individuals;
- the individuals are relatively homogeneous (i.e., they are either all identical or they belong to a few typologies);
- the interactions among the individuals are based on simple behavioral rules that exploit only local information that the individuals exchange directly or via the environment (stigmergy);

- the overall behaviour of the system results from the interactions of individuals with each other and with their environment, that is, the group behavior self-organizes.

The characterizing property of a swarm intelligence system is its ability to act in a coordinated way without the presence of a coordinator or of an external controller. Many examples can be observed in nature of swarms that perform some collective behavior without any individual controlling the group, or being aware of the overall group behavior. Notwithstanding the lack of individuals in charge of the group, the swarm as a whole can show an intelligent behavior. This is the result of the interaction of spatially neighboring individuals that act on the basis of simple rules. Most often, the behavior of each individual of the swarm is described in probabilistic terms, each individual has a stochastic behavior that depends on his local perception of the neighborhood.

2.5 Ant Colony Optimization Algorithm

The **Ant Colony Optimization** (ACO) [10] algorithm is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs.

This algorithm is a member of **Ant colony algorithms** family that constitutes some metaheuristic optimizations based on Swarm intelligence methods. Initially proposed by Marco Dorigo in 1992 in his PhD thesis, the first algorithm was aiming to search for an optimal path in a graph; based on the behavior of ants seeking a path between their colony and a source of food. The original idea has since diversified to solve a wider class of Numerical problems. As a result, several problems have emerged, drawing on various aspects of the behavior of ants.

In the real world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep traveling at random, but to instead follow the trail, returning and reinforcing it if they eventually find food.

Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The longer it takes for an ant to travel down the path and back again, the more pheromones evaporate. A short path, by comparison, gets marched over faster, and thus the pheromone density remains high as it is laid on the path as fast as it can evaporate. Pheromone evaporation has also the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained.

Thus, when one ant finds a good (i.e., short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads all the ants following a single path. The idea of the ant colony algorithm is to mimic this behavior with "simulated ants" walking around the graph representing the problem to solve.

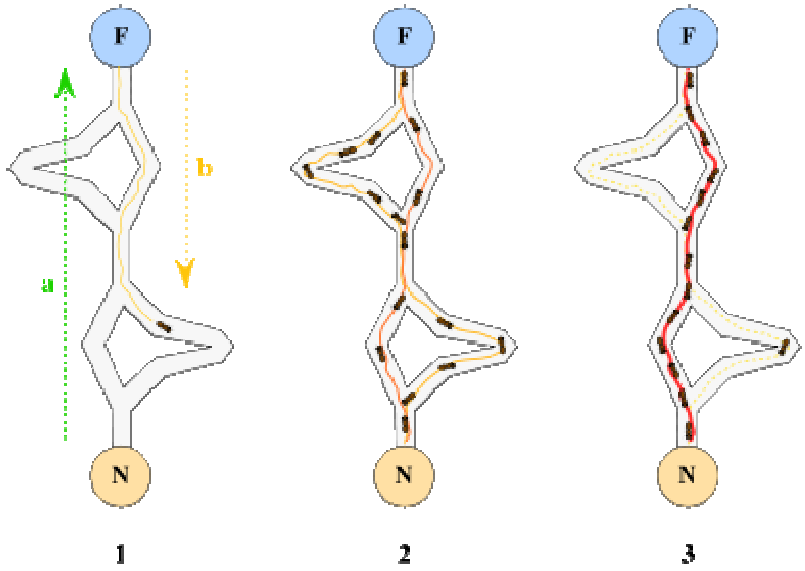

Figure 2.7 : Ant Colony Optimization

The original idea comes from observing the exploitation of food resources among ants, in which ants' individually limited cognitive abilities have collectively been able to find the shortest path between a food source and the nest. This is shown in Figure 2.7.

1. The first ant finds the food source (F), via any way (a), then returns to the nest (N), leaving behind a trail pheromone (b)
2. Ants indiscriminately follow four possible ways, but the strengthening of the runway makes it more attractive the shortest route.
3. Ants take the shortest route, long portions of other ways lose their trail pheromones.

In a series of experiments on a colony of ants with a choice between two unequal length paths leading to a source of food, biologists have observed that ants tended to use the shortest route. A model explaining this behavior is as follows:

1. An ant (called "blitz") runs more or less at random environment around the settlement;
2. If it discovers a food source, it returns more or less directly to the nest, leaving in its path a trail pheromone;
3. These pheromones are attractive, ants from nearby will be inclined to follow, more or less directly, the track;
4. Returning to the nest, these ants will strengthen the runway;
5. If two tracks are possible to achieve the same food source, that being the shorter will be at the same time, traveled by ants over the long runway;
6. The short track will be increasingly enhanced and therefore more attractive;
7. The long runway will eventually disappear as pheromones are volatile; and
8. Eventually, all the ants determine the shortest track.

Ants use the environment as a medium of communication. They exchange information indirectly by depositing pheromones, all detailing the status of their "work." The information exchanged has a local scope, only ants located where the pheromones were filed access. This system is called "Stigmergy" and occurs in many social animals

(it has been studied in the case of the construction of pillars in the nests of termites). The mechanism to solve a problem too complex to be addressed by ants alone is a good example of a self-organized system. This system is based on positive feedback (the deposit of pheromone attracts other ants that will strengthen their turn) and negative (dissipation of the runway by evaporation prevents the system from thrashing). Theoretically, if the quantity of pheromone remained the same over time on all branches, no runway would be chosen. However, because of feedback, a variation on a low branch will be amplified and allow the choice of a branch. The algorithm will move from an unstable state with no branch is stronger than another to a stable state where the route is composed of the strongest branch.

Ant colony optimization algorithms have been applied to many combinatorial optimization problems, ranging from quadratic assignment to fold protein or routing vehicles. A lot of derivate methods have been adapted to dynamic problems in real variables, stochastic problems, multi-targets or parallel implementations. It has also been used to produce near-optimal solutions to the traveling salesman problem. They have an advantage over simulated annealing and genetic algorithm approaches of similar problems, when the graph may change dynamically; the ant colony algorithm can be run continuously and adapt to changes in real time. This is of interest in network routing and urban transportation systems.

An annotated example

The first ACO algorithm was called the Ant system and was aimed to solve the traveling salesman problem, in which the goal is to find the shortest way to link a series of cities. The general algorithm is relatively simple based on a set of ants, each traversing a distance from a series of cities. At each stage, the ant chooses to move from one city to another according to some rules:

1. It can visit each city only once;

2. A far town has less chance of being chosen (or less visible);

3. The more intensity of the pheromone trail lay out on the edge between two cities, the greater the trip will likely be chosen;

4. Once completed its journey, the ant files on all edges traveled having high concentration of pheromones if the journey is short; and

5. After each iteration, trails of pheromones evaporate.

Figure 2.8 shows a step-by-step application of ACO algorithm to the Travelling Salesman Problem.

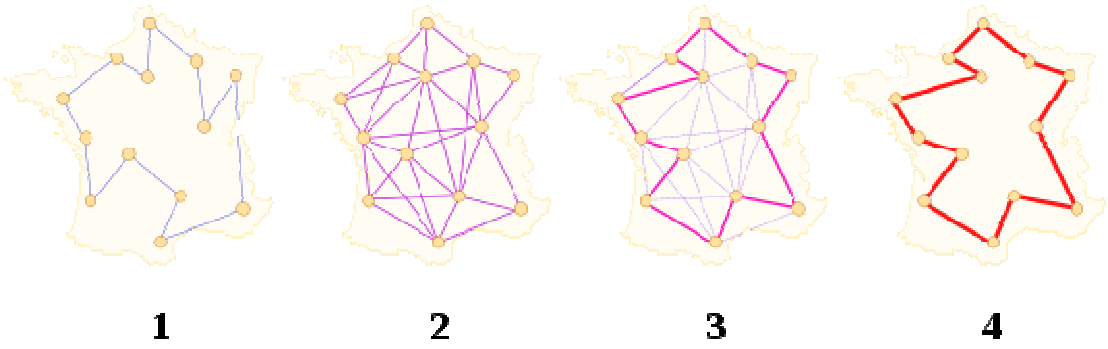

Figure 2.8 : Example of Traveling Salesman problem solving by Ant Colony Optimization

CHAPTER III

WEB NAVIGATION ANALYSIS AND SIMULATION USING ANT COLONY OPTIMIZATION

3.1 Web navigation and ACO

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.

3.2 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.

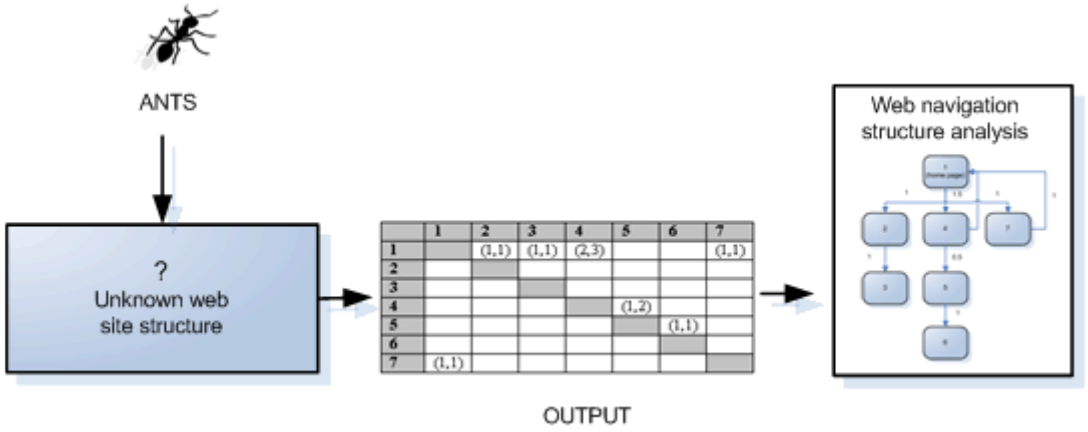

Figure 3.1 : Web navigation structure analysis.

Figure 3.1 illustrates a conceptual view of the proposed web structure analysis approach. The ant agents hereafter will be simply referred to as ants are employed as a means to navigate through the designated web site. Thereby the structure of the web site can be determined. Details on analysis procedure are elucidated in the sub-sections that follow.

3.2.1 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 algorithm can always access and verify their destination pages from this IP address table. Figure 3.2 shows an example of the IP address table.

ID	Internal IP addresses
1	161.100.100.0
2	162.100.100.0

Figure 3.2 : A sample IP address table.

3.2.2 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 [2]. The proposed modified ACO algorithm reverts the ants' travel to the

opposite direction that has the lowest density of pheromone first. This will help the 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.

3.2.2.1 Ant package

An Ant package entry encompasses 4 elements shown in Figure 3.3.

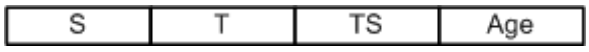

Figure 3.3 : An Ant package entry.

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 [15]; and *Age* is the variable representing the available life time of the ants. Figure 3.4 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.

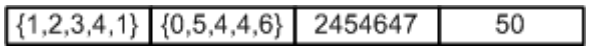

Figure 3.4 : A sample entry of the Ant package.

3.2.2.2 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. An ant's traversal will terminate if the ant's *Age* is reduced to zero. This helps prevent potential infinite loops problem. Updating the ant's *Age* proceeds as follows:

If $Age > 0$, $Age = Age - 1$

If $Age = 0$, terminate itself (skip steps 2-6)

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:

$$T = Current\ Timestamp - latest\ value\ of\ TS \quad (3.1)$$

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.

$$P(\alpha) = MIN\_P \quad , if\ EL(\alpha) = 0 \quad (3.2)$$
$$= MAX\_P \quad , if\ EL(\alpha) = 1$$

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 $P(\alpha)$ 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(**α**)* is greater than *MIN_P*), the ants will increase the pheromone value *P(**α**)* 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 3.5 : Sample data kept in pheromone table.

The original Pheromone update formula shown below simulates the real ant's pheromone behaviour.

$$\tau_i^j \leftarrow (1-\rho)\cdot\tau_i^j + \frac{\rho}{\mathfrak{S}_{\text{upd}}}\cdot\sum_{\{s\in\mathfrak{S}_{\text{upd}}|c_i^j\in s\}} F(s), \tag{3.3}$$

where $\mathfrak{S}_{\text{upd}}$ is the set of solutions that are constructed in the current iteration, and s^{bs} is the best-so-far solution. The parameter $\rho \in (0, 1]$ is called evaporation rate. It has the function of uniformly decreasing all the pheromone values. From a practical point of view, pheromone evaporation is needed to avoid a too rapid convergence of the algorithm toward a sub-optimal region. It implements a useful form of forgetting, favoring the exploration of new areas in the search space. *F* is commonly called the quality *function*. This is a function to determine the feasible possibility of solutions. For the advantage of web navigation, the pheromone value will be kept as long as the ant traveling around the web site. So, the evaporation rate is always equal to 1. Also, $\mathfrak{S}_{\text{upd}}$ is usually not used. This introduces for the mathematical purpose only.

Furthermore, we can consider the last expression (*F* function) of formula as a constant so that the pheromone update formula can be simplified.

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

$$\begin{aligned} P(\alpha) &= P(\alpha) + \Delta \quad , \text{if } P(\alpha) > MIN\_P \\ &= DEF\_P(\alpha) \quad , \text{if } P(\alpha) = MIN\_P \end{aligned} \tag{3.4}$$

where Δ denotes the pheromone increment upon arriving at page α. *DEF_P(α)* is the starting value of pheromone at node α. Thus,

DEF_P(α) = - (number of internal links on page α)

The results are stored in the pheromone table as illustrated in Figure 3.5.

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(β)*
- If there are pages that share the same lowest pheromone level *P(β)*, 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 in Figure 3.6.

```
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
```

Figure 3.6 : Web navigation using Ant Colony Optimization's pseudocode

3.2.3 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.

$$INCREASE\_RATE = [(V_t - V_{t-1})/V_{t-1}] * 100 \quad (3.5)$$

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%

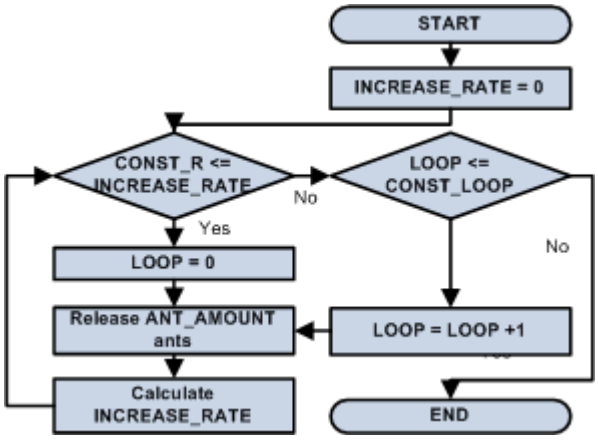

Figure 3.7 : An Ant decision flow chart.

As shown in Figure 3.7, 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.

Figure 3.8 shows samples of information kept in the Ant package after the first ant has been sent in the sample web site. Figure 3.9 shows the adjacent table after retrieving information from the first ant package. Figure 3.10 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*.

ANTS	ANT Package
1	S= {1, 2, 3}, T= {0, 1, 1}

Figure 3.8 : Sample data of the Ant package after first ant sent to the system

	1	2	3
1		(1,1)	
2			(1,1)
3			

Figure 3.9 : Sample inputs of adjacent table for Ant 1

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 3.10 : Sample data of the Ant package after two ants sent to the system

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

Figure 3.11 demonstrates the values kept in the adjacent table after the first ant and the second ant have been sent in the web site, respectively. The adjacent table at row *i* and column *j* will be updated corresponding to the data obtained from current ant; the N_{ij} element will be calculated by counting number of ants leaving from page *i* and visiting at page *j*. In this example, the second ant moves from page *i* to visit page *j* 2 times as highlight in Figure 3.12. The T_{ij} element will be calculated from cumulative time used by the first and second visit that the second ant moves from page *i* to page *j*. In this example, the elapsed times that the second ant moves from page i to page j at the first and second visit are 1 and 2, respectively. The cumulative of both elapsed times is 3. From this calculation, the pairwise (2,3) is updated into the adjacent table.

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, 2, 2, 1}

Figure 3.12 : Sample data of the Ant package after two ants are sent to the system. The highlight shown the elapsed time that the ant uses to move from page 1 to page 4

3.2.4 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 3.14. The elapsed time that ants move from one page to another page might vary depending noise from the internet traffic, so the average value is used instead. 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*. Figure 3.13 shows the calculation of adjacent table.

	1	2	3	4	5	6	7
1		1		1.5			1
2			1				
3							
4	1				1		
5						1	
6							
7	1						

Figure 3.13 : Sample calculation using data from the adjacent table (Tij/Nij)

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.

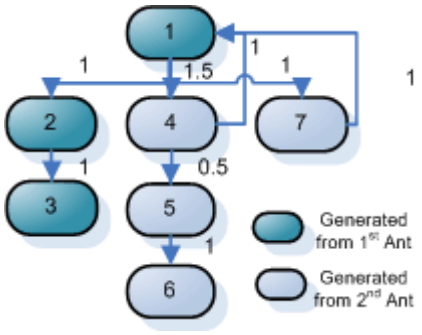

Figure 3.14 : A web navigation diagram resulting from Ant package.

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 3.15 : 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 3.15 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 3.16 depicts the new web navigation diagram based on data from the new adjacent table.

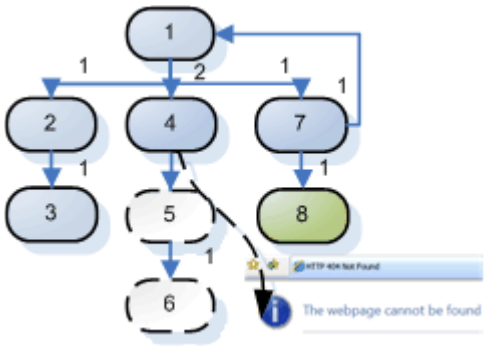

Figure 3.16 : A web navigation error found after changing the web structure.

	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 3.17 : Sample methods to identify the unreachable pages.

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.

$$S_{\text{pages added}} = S_{\text{actual}} - S_{\text{baseline}} \quad (3.6)$$

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.

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 3.17, 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:

$$S_{\text{dead links}} = \{link_{ij}, link_{ij} \text{ is a link from page } i \text{ to page } j \mid a_{ij} \in \text{baseline adjacent table} \wedge a_{ij} \notin \text{current adjacent table}\} \quad (3.7)$$

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

$$S_{\text{unreachable pages}} = \{j \mid \forall i, a_{ij} = \varnothing\} \quad (3.8)$$

As shown in Figure 3.17, 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,

$$S_{\text{unreachable causal pages}} = S_{\text{data missing rows}} - S_{\text{unreachable pages}} \quad (3.9)$$

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 3.16, 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 in Table 3.1.

The number of pages calculated from rows (or columns) of the adjacent table and the number of links calculated from the fields in adjacent table contains parewise a_{ij} information. From this example, the number of pages and links are 5 and 5, respectively. Similarly, the number of new additional pages is calculated by equation (3.5). In this example, page “8” becomes the newly added page to the web site based on the above set operation.

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

Table 3.1 : Example of analysis result.

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

We can use similar set operation to calculate number of new additional links and number of removal pages. From the above example, the number of new additional links and removed pages are 1 (new link from page 7 to page 8 found) and 2 (page 5 and page 6 are removed from the website), respectively. The number of dead links or removal links can be calculated from dead link set operation (3.6). In this example, the pairwise a_{41}, a_{45} and a_{56} in the baseline adjacent table are absent in the current adjacent table. So the number of dead links is 3. Meanwhile, the number of unreachable pages can be simply calculated by seeking the columns that do not contain any pairwise entries in the adjacent table. In the above example, there is no pairwise entry in both column 5 and 6, so they are classified as unreachable pages. The pages that might be the cause of unreachable page can be identified by using unreachable causal page set operation (3.8). The calculation and result are described above. The number of leaf pages can also

be calculated by finding rows that do not contain any such pairwise entries. In this example, row 3, 4 and 8 can be identified as leaf pages.

The highest content load time and the lowest content load time can be retrieved from the maximum and minimum values of *T* set of Ant packages, respectively.

CHAPTER IV

EXPERMENTAL RESULTS AND

MESUREMENT STATISTIC

4.1 Experimental 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 4.1 and Table 4.1, 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.

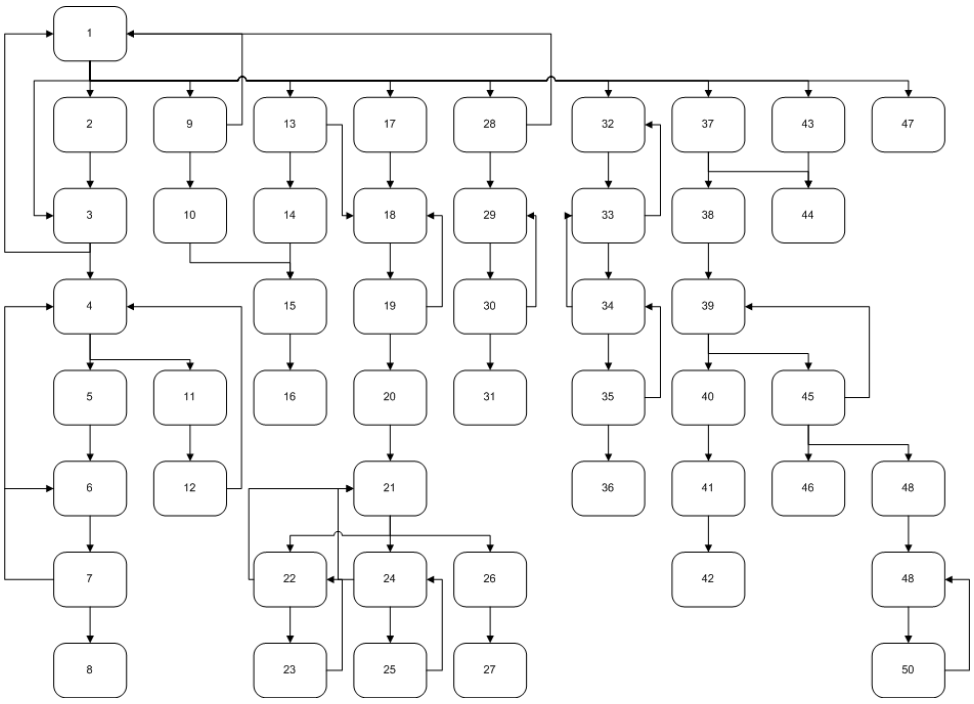

Figure 4.1 : Web site structure.

Table 4.2 and Figure 4.2 illustrate 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.

Table 4.1 : Predefined constants.

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

Table 4.2 : The relationship between the number of ants sent into the system and number of pages found.

ANT	Pages found	ANT	Pages found
1	8	14	49
2	12	15	50
3	25	16	50
4	29	17	50
5	34	18	50
6	36	19	50
7	39	20	50
8	39	21	50
9	40	22	50
10	42	23	50
11	42	24	50
12	49	25	50
13	49	26	50

Table 4.3 : The relationship between the number of ants sent into the system number of pages found, unreachable pages and new additional pages.

ANT	Pages Found	Unreachable pages	Additional pages
1	5	45	0
2	12	38	0
3	15	37	2
4	17	36	3
5	23	31	4
6	27	27	4
7	31	23	4
8	31	23	4
9	35	19	4
10	37	17	4
11	39	16	5
12	39	16	5
13	43	12	5
14	44	11	5
15	44	11	5
16	44	11	5
17	44	11	5
18	44	11	5
20	44	11	5
21	44	11	5
22	44	11	5
23	44	11	5
24	44	11	5
25	44	11	5
26	44	11	5

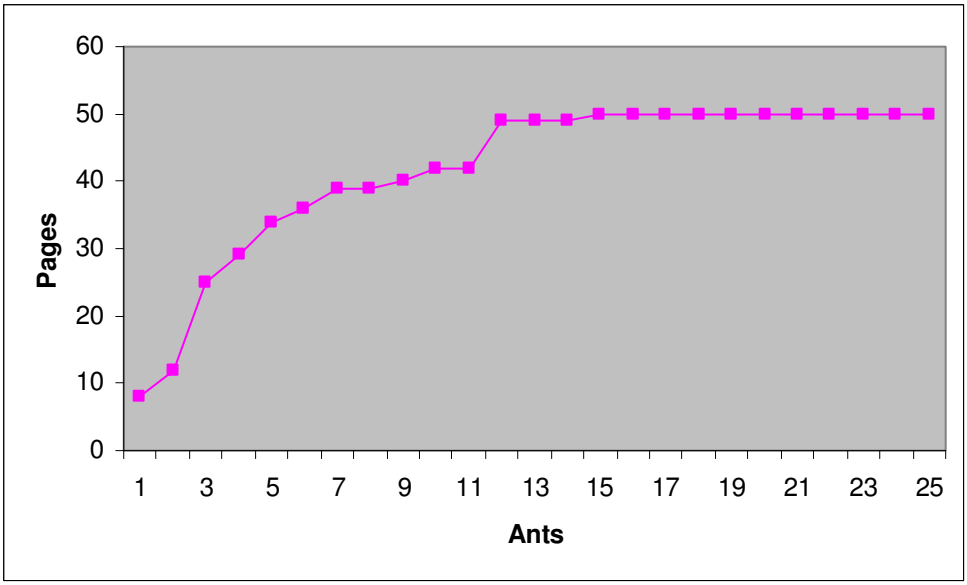

Figure 4.2 : The number of pages found VS the numbers of ants sent.

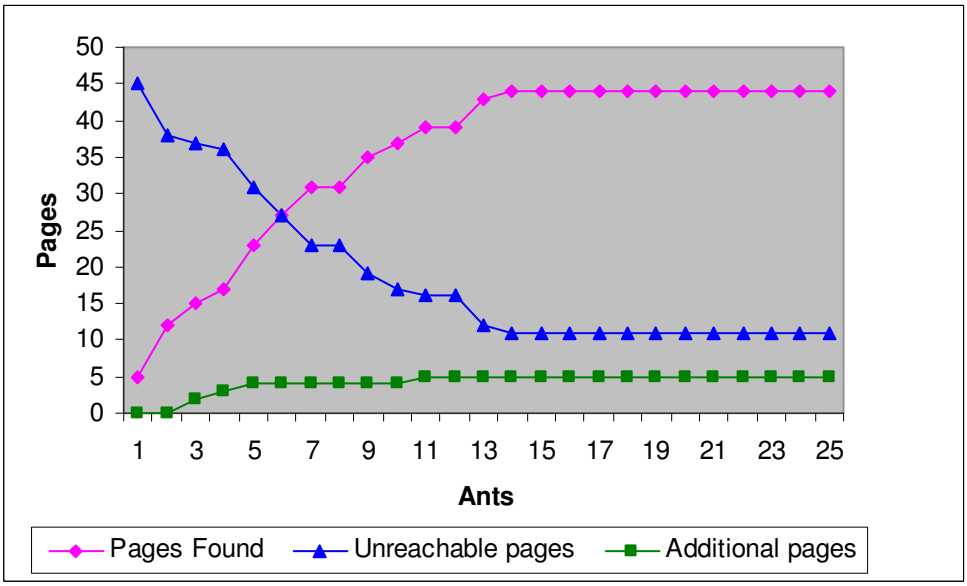

Figure 4.3 : The number of pages found, unreachable pages, and additional pages after modifying web site structure.

The following statistics were complied as the results of the experiment depicted in Table 4.3 and Figure 4.3, 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. Table 4.4 and Figure 4.4 compare 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.

Table 4.4 : The relationship between the number of ants sent into the system number of pages found in each iteration.

ANT	Page Found			ANT	Page Found		
	Iteration 1	Iteration 2	Iteration 3		Iteration 1	Iteration 2	Iteration 3
1	8	7	2	16	50	49	49
2	12	21	4	17	50	49	50
3	25	24	8	18	50	49	50
4	29	29	15	19	50	49	50
5	34	33	20	20	50	50	50
6	36	38	20	21	50	50	50
7	39	42	22	22	50	50	50
8	39	44	26	23	50	50	50
9	40	44	37	24	50	50	50
10	42	45	42	25	50	50	50
11	42	45	47	26	50	50	50
12	49	47	47	27	50	50	50
13	49	49	47	28	50	50	50
14	49	49	47	29	50	50	50
15	50	49	49	30	50	50	50

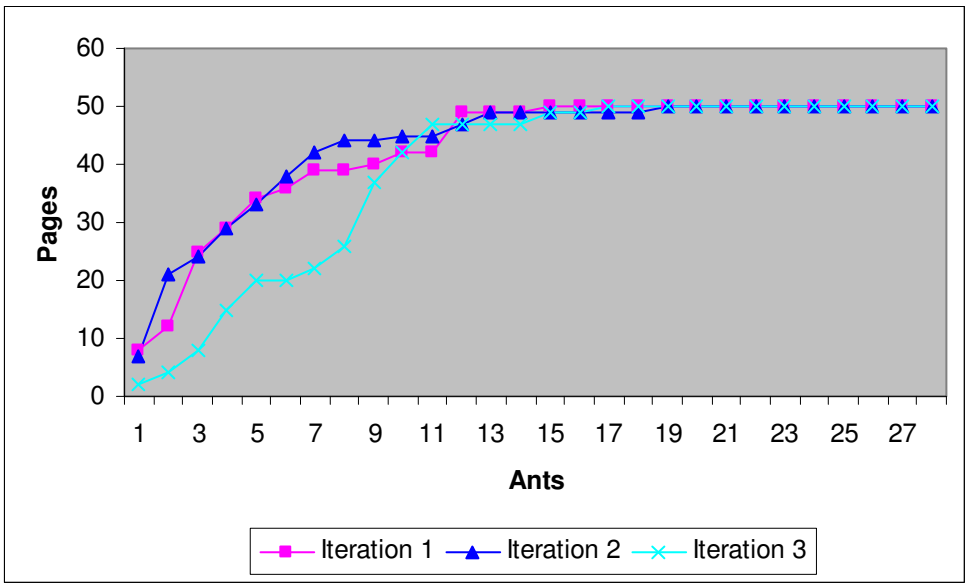

Figure 4.4 : Different number of ants deployed in separate iterations.

4.2 Measurement statistics

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 as oppose to the original approaches, which executed serially. Some comparative statistics are given below.

Table 4.5 : The comparison of DFS, BFS and ANT

	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 and m = maximum depth

CHAPTER V

CONCLUSION

This thesis 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.

From the experimental results shown in the previous chapter, we can conclude that

1. The use of Ant Colony Optimization entails the correct solution with lower memory requirement than the DFS and BFS approaches. Ant deployment to explore the web site can be carried out concurrently, thereby obtaining the results faster.
2. The outcome from the Ant package can be easily translated and kept in the database for future reference. All of this information can be used to reorganize web site structure and improve performance.
3. The infinite loop searching problem that was found in DFS and BFS can be resolved with the help of *Ant's Age* concept of modified Ant Colony Optimization proposed in this thesis.

The simplicity of the original of Ant Colony Optimization can be applied to solve many optimization problems such as Traveling Salesman Problem, Web navigation, and others. Nonetheless, this thesis makes a slight modification to the original concept to use in the opposite way. Instead of moving to the path that has the most pheromone density value, the ants will move to the path that has low pheromone density. This modification can help the ants seek and discover new paths in the system. In addition, many ants can travel around the system simultaneously. Consequence, the coverage of searching area is increasing. This concept can help ants find a number of additional links, removal links, and unreachable links and retrieve a lot more useful information from the system. Using simply table mapping technique, the raw information gathering

from ants can be translated into valued information. All of this information can be kept in the reference database and used to improve the quality of the system, thus minimizing the amount of routine administrative work.

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 reference exploration, and minimal agents used will be conducive toward performance improvement of the proposed approach.

REFERENCES

[1] Montresor, A., Meling, H., and Baboglu, O. (2002). Messor: Load-Balancing through a Swam of Autonomous Agents. Proceedings of the 1st International Workshop on Agents and Peer-to-Peer Computing : 25-137.

[2] Dorigo, M., Maniezzo, V., and Colorni A. 1996. The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B 26 (1): 29-41.

[3] Li, H., and Lam, C. P. (2004). Software Test Data Generation using Ant Colony Optimization. International Conference of Computational Intelligence : 1-4.

[4] Li, H., and Lam, C. P. 2004. Optimization of State-based Test Suites for Software Systems: An Evolutionary Approach. International Journal of Computer & Information Science 5 (3): 212-223.

[5] Wagner, I. A., Lindenbaum, M., and Bruckstein, A. M. 2000. ANTS: Agents, Networks, Trees, and Subgraphs, Special issue on Ant Colony Optimization. In M. Dorigo., G. Di Caro., and T.Stützle (ed.), Future Generation Computer Systems 16 (8): 915-926.

[6] Babaoglu, O., Meling, H., and Montresor, A. (2002). Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems. Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS 02).

[7] Pargas, R. P., Harrold, M. J., and Peck, R. 1999. Test-Data Generation Using Genetic Algorithms. Software Testing, Verification and Reliability 9: 263-282.

[8] Golden, B., and Stewart, W. 1985. Empiric analysis of heuristics. In E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy-Kan, and D. B. Shmoys (ed.), The Travelling Salesman Problem, New York: Wiley.

[9] Bonabeau, E., Dorigo, M., and Theraulaz, Z. 1999. Swarm Intelligence. From Natural to Artificial Systems, New York: Oxford University Press.

[10] Dorigo, M. and Di, Caro, G. (1999). Ant colony optimization: a new meta-heuristic. Proceedings of 1999 Congress on Evolutionary Computation : 1470-1477.

[11] Leung, K. R. P. H., Hui, L. C. K., Yiu, S. M., and Tang, R. W. M. (2000). Modelling Web Navigation by Statechart. Proceedings of the 24th Annual International Computer Software and Applications Conference (COMPSAC 2000) : 41-47.

[12] Wikipedia. Website[online]. (n.d.). Available from: http://en.wikipedia.org/wiki/Website [2009, January 1]

[13] Wikipedia. Web 2.0[online]. (n.d.). Available from : http://en.wikipedia.org/wiki/Web_2.0 [2009, January 1]

[14] Wikipedia. Site map[online]. (n.d.). Available from : http://en.wikipedia.org/wiki/Site_map [2009, January 1]

[15] Moyer, G. 1981. The Origin of the Julian Day System. Sky and Telescope 61 (April): 311-313.

[16] Gerardo, B., and Jing, W. (1989). Swarm intelligence in cellular robotics systems. Proceedings of NATO Advanced Workshop on Robots and Biological System.

CURRICULUM VITAE

Ekachai Jinhirunkul was born in 1982. He received a Bachelor Degree in Science (Majoring Computer Science) with Second Class honor from Chulalongkorn University in 2004. He is working as Team Leader for DST International, Bangkok and is also pursuing a Masters degree in computer science.