# TOWARDS DISTRIBUTION OF WEB SITES IN A CRAWLER USED FOR LARGE SCALE WEB ACCESSIBILITY ASSESSMENT.

LEIMING CHEN, DONGMEI WANG AND MORTEN GOODWIN OLSEN.

Faculty of Technology, Agder University College
Serviceboks 509, N-4898 Grimstad, Norway
dongmw05@student.hia.no, leimic05@student.hia.no, Morten.G.Olsen@hia.no

ABSTRACT.

A mechanisms used for large scale accessibility measuring may involve a distributed web crawler. Furthermore, it makes sense to spread the web sites involved to different access points (crawler locations / crawler nodes) of the distributed crawler. We will in this publication present an algorithm utilising the available resources to a much greater extent than the traditional uniform distribution of web sites.

Our novel algorithm, namely the Time Weighted Object Migration Automaton (TWOMA), is an extension on the Object Migration Automaton (OMA) presented in [1].

The heart of our scheme involves continuously accessing web sites while measuring the duration of each access. Note that accessing a site involves downloading and measuring the accessibility. When a web site is accessed the following happens;

- If the duration of accessing the web site is less than the average duration for all web sites in the corresponding accesspoint, the web site is moved one state closer to the most internal state of this access point.
- If the duration of accessing the web site is more than the average duration in the corresponding accesspoint, the web site is moved one state closer to the boundary state of this access point.
- If the site is already located in the boundary state, the site is moved to another random access point.

The above scheme is repeated as long as the crawling / measurement is ongoing. This ensures that the scheme works in a dynamic environment (as the real web). Furthermore, we will in this publication show that the algorithm is working towards an optimal distribtion of web sites in available access points using experimental data.

## 1. INTRODUCTION

The main part of a mechanism used for large scale web assessment machinery, e.g. as a web accessibility measurement tool [2, 3, 4] or a search engine [5], will be a web crawler (also known as web spider or web robot), possibly distributed. It is highly desirable that both the crawler can download as many updated web sites as possible compared to the locally stored copies and that the time used on any evaluation of the complete set of web sites is as small as possible.

This can be seen as a distributed resource allocation problem, as the crawler should minimise the time spent downloading and evaluation of the web sites with limited resources available when the resources available vary between access points.

The above problem has, to the best of our knowledge, not been adressed in the literature earlier. In contrast non-distrirbuted solutions to resource allocation problems in crawlers often use estimated tracking faces [6, 7]. Other solutions [8, 9] use sampling strategies, applying parts of the available resources for sampling and further determine if the sampled web site should be priorities in the remaining resources available.

In contrast, [10, 11, 12], showd that having an estimation period has a disadvantage in dynamic environments as the optimal solution changes in time. Instead they presented a solution using a game of Learning Automata without the need for a periodically reestimation faces or sampling.

We have in this paper investigated the problem of distributed resource allocation in a web crawling based on an object partitioning algorithm, namely the Object Migration Automaton (OMA) [1, 13]. We have further extended the OMA and developed the Time Weighted Object Migration Automaton (TWOMA) to work as a distributed resource allocation algorithm. All tests has been performed in a simuated environment representing the behavior of real web as best known possible.

## 1.1. **Motivation.**

1.1.1. *Distributed Web Crawling.* Due to the explosive size of Internet, web search engines are becoming increasingly important as the primary means of locating relevant information. Such search engines rely on massive collections of web sites that are acquired with the help of web crawlers, which traverse the web by following hyperlinks and storing downloaded pages in a large database that is later indexed for efficient execution of user queries [14]. Thus, highly efficient crawling systems are needed in order to download thousands of web sites indexed by the major search engines.

Two issues are addressed for a good crawler for large scale web assessment in [14]. First, the crawler needs to have a good crawling strategy, i.e., deciding the order of web sites to download. Secondly, a crawler needs to have a highly optimised system architecture that can download a large number of sites per second while being robust against crashes, manageable, and considerate of resources and web servers.

1.1.2. *Large scale web accessibility benchmarking.* Web accessibility benchmarking is carried out in many European countries today. Addionally, projects like the European Internet Accessibility Observatory (EIAO) [2, 4] are developing a machinery designed to crawl and evaluate 10 000 web sites according to the Unified Web Evaluation Methodology (UWEM) [15]. In order to increase the efficiency of such large scale measurements, an effective distribution of the web crawler resources is desirable.

## 1.2. **Problem background.** 
The problem of distributing web sites in a web crawler when the duration of accessing each site is unknown and each site behaves differently depending on which part of the crawler (access points) they are addressed from in order to minimise the duration of each crawl has, to the best of out knowledge, not been adressed in literature earlier.

The behaviour web sites and access points is further elaborated in section 3.1.

## 2. Related technologies

## 2.1. **Object Migration Automaton.** 
In [1] Oommen and Ma presented the Object Migration Automation (OMA) as a solution to the Equipartitioning Problem. The automaton is offered a set of actions by which it interacts, and is constrained to choose one of these actions. When an action is chosen, the automaton is either rewarded or penalised. The algorithm works such that the automaton learns the action which has the minimum penalty probability and eventually chooses this action more frequently than the other actions.

In short the automaton is presented as a set of $R$ actions each represented as an arm. Furthermore, for each action, there is a predefined number of states $N$. Additionally, a set

of abstract objects $\Omega = \{O_1, ..., O_w\}$, moves around within the automaton. The automaton functions as described below.

- Each object $O_i$ is all the time located in one of the predefined $N$ states and respectivle connected to one of the predefined $R$ actions.
- Each object $O_i$ has an action $\alpha_i$. This action is dependant on which of the $R$ arms the object is located in.
- All objects are accessed in pairs ($O_i$ and $O_j$) and whenever two $O_i$ and $O_j$ are accessed they will always be penalised or rewarded.
- Whenever an object is rewarded, it moves one state closer to the most internal state in its corresponding arm. If the object is already located in its most internal state, it does not move.
- Whenever an object is penalised, it moves one state closer to the boundary state in its corresponding arm. If the obejct is already located in its boundary state, it is candidate to move to one of the other $R$ arms.
- The objects are rewarded if $O_i$ and $O_j$ are accessed together and are located in the same arm.
- The objects are penalised if $O_i$ and $O_j$ are accessed together and are located in different arms.
- If $O_j$ and $O_i$ penalised and at least of of them are located boundary states of different arms, they switch arms and thus change their action.

For a more in-depth analysis of the OMA we refer the reader to [1, 13], where the OMA was shown to be able to provide an optimal partitioning of the $O\Omega$ objects in the $R$ arms.

## 3. Time Weighted Object Migration Automaton (TWOMA)

In order to solve the distributed resource allocation problem we have extended the OMA described in 2.1. In short we have used the duration of accessing each object as basis for partitioning, in contrast to using the order of which the objects are accessed. Our novel scheme, namely the Time Weighted Object Migration Automaton (TWOMA), is presented below.

| Symbol | TWOMA | Web Crawler |
|---|---|---|
| $\Omega = \{O_1, ..., O_w\}$ | Objects | Web sites |
| $R = \{P_1, ..., P_n\}$ | Arms | Access Points (Crawler locations) |
| $N$ | Internal states of each $P_j$ | Download and evaluation duration |
| $t$ | Time | Crawler run |
| $O_{i,t}$ | Duration of $O_i$ at time $t$ | Download and evaluation of web site $i$ at time $t$ |

TABLE 1. Mapping between TWOMA and a Web Crawler

In table 1 the mapping between TWOMA and a real web crawler is shown. Note that all the time each object $O_i$ is located in a corresponding arm $P_j$. Additionally, each $O_i$ in the corresipinding arm $P_j$ is located in one of the $N$ states. An object may only move from $N_i$ to ether $N_{i-1}$ or $N_{i+1}$ according to the rules presented below. $N_0$ is the boundary state while $N_N$ is the most internal state of each arm in $R$. Figure 1 shows a representation of TWOMA when $N$=3 and $|R|$=3.

3

The heart of our scheme involves continuously accessing web sites while measuring the duration of each access (how duration is simulated is described in 3.1). Note that accessing a site involves downloading and measuring the accessibility.

At each time $t$ each object $O_i$ is accessed and the duration $O_{i,t}$ is measured. All object are accessed in a sequential order, even though this has been shown to be suboptimal in several studies [6, 7, 8, 9, 10, 11, 12]. However, we have chosen only to focus on the distribution of $\Omega$ and thus assume that each access has an equal contribution to the result.

The rules of the TWOMA is as following for each crawler run $t$:

- Access all $O_i$ in $\Omega$ sequentially and measure the duration $O_{i,t}$.
- If $O_{i,t}$ is less than the average duration of all objects in the corresponding arm $P_j$, $O_i$ is rewarded and moved one state closer to the most internal state of $P_j$.
- If $O_{i,t}$ is more than the average duration of all objects in the corresponding arm $P_j$, $O_i$ is penalised and moved one state closer to the boundary state of $P_j$.
- Any object $O_i$ that is already located in the boundary state and penalised is moved from the arm $P_j$ to a random arm $P_k \epsilon \{P_1, ..., P_{j-1}, P_{j+1}, ..., P_n\}$.
- Any object $O_i$ that is already located in the most internal state and rewarded, remains unchanged.

The above scheme is repeated as long as the crawling / measurement is ongoing. This ensures that the scheme works in a dynamic environment (as the real web).
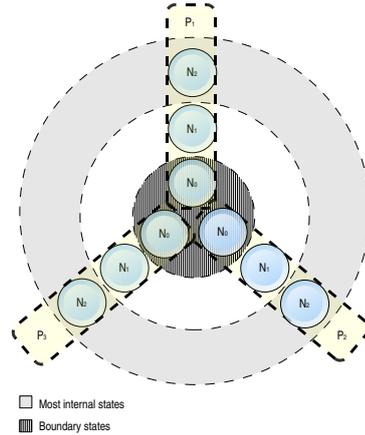


FIGURE 1. The TWOMA represented with $N{=}3, R = \{P_1, P_2, P_3\}$ and $|R| = 3$.

3.1. **Test Environment.** In order to test the presented algorithm we have created a simluated environment. This environment is designed to represent the behaviour of the real web as best as possible, both with regards to the behavior of web sites and access points.

In a real web environment the duration of downloading the complete, or representative sample of, the web sites depends highly on the following;

- The size of the web site.
- The uploading capacity of the web site server.
- The physical location of the web site server.

The duration of evaluating web sites from each access point depends highly on the following;

- The resources available in this access point (CPU, Memory).
- The number of hours the access point is available. (Some acceess points may only be partly be available)
- The downloading capacity from the server.

The environment is designed in such way that there is a number of access point each representing a physical location of the distributed crawler and thus has a predefined resource available. The available resources in the different access points vary, which simulates the

different resources available at each access point. A direct consequence of this is that some access points will use a long time to download and evaluate web sites, while others will be a lot quicker. Additionally, and most noticable, the individual web sites will have different download and evaluation time.

In order to represent the environment presented above, each web site in out environment is seen as an object with an unknown stochastich random duration. This means that any $O_{i,t}$ is represented as a random variable that can only be retrieved after $O_i$ has been accessed at crawler run $t$. This approach is similar to the the approach in [16, 17, 18, 11, 19, 20]. In addition to the environment in these approaces, each $O_i$ in our environment has different behavior when located in different $P_j$.

It is noticable that in this environment, an $O_j$ will always have a perfect arm $P_m$. If $O_i$ is located in $P_m$ the duration $O_{i,t}$ will in average be less than if $O_i$ was located in any other arm than $P_m$. The perfect arm for $O_i$ is however not known by the TWOMA algorithm and only the instasiation of $O_{i,t}$ (duration) can be used in the learning process.

## 4. EMPIRICAL RESULTS

In this section we evaluate the presented algorthm using synthetic data from the environment presented in 3.1. The number of states $N$ in each $P_j$ has been set to 5. All results presented is the outcome of averaging 10 runs. Furthermore, each object is initially uniformly distributed among each arm, if otherwise not stated.

The distributions strategies present in these experiments are:

**Uniform**. All objects in $\Omega$ are distributed uniformly among each access point $R$.

**TWOMA**. All objects in $\Omega$ distributed according to the TWOMA algorithm prestented in 3.

### 4.1. **Comparison.** Here we present the performance of TWOMA compared with the traditional uniform distribution.

In Figure 2 the behavior of both the TWOMA and Uniform in a static environment where $|\Omega| = 10000$ with $|R|=8$ is shown. Note that the y-axix shows the duration of accessing all object (downloading and evaluating the web sites), while the x-axis shows the total number crawler runs ($t$). This experiment shows that the TWOMA starts out having an equal performance as a unfiform distribution, but quickly reduces the total duration of each crawl as $t$ increases. Note also that the $|\Omega| = 10000$ in this experiment is equal to the goal presented at [2].

### 4.2. **Dynamic environment.** Because the real web is dynamic, we show in this section the behavior of the TWOMA in a dynamic environment.

Figure 3 shows the behavior of TWOMA where most optimal arm $P_m$ for each $O_i$ is changed whenever $t$ mod 100 is 0 (for each 100th crawl). This experiment shows that the algorithm is able to run in a dynamic environment such as the real web. Note that the algorithm reaches close to the optimal distribution before the environment changes.

### 4.3. **Number of crawler runs needed to reach optimal distribution.** Figure 4 shows the number of crawls needed to reach an optimal partitioning of the web sites using TWOMA. The optimal distribution is when each object $O_i$ has reached its optimal arm $P_m$.
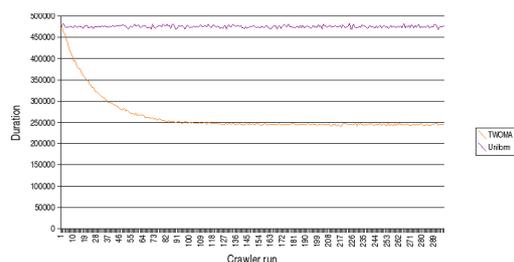
FIGURE 2. The TWOMA and a uniform distribution in a static environment with $|\Omega| = 10000$ and $|R|=8$.
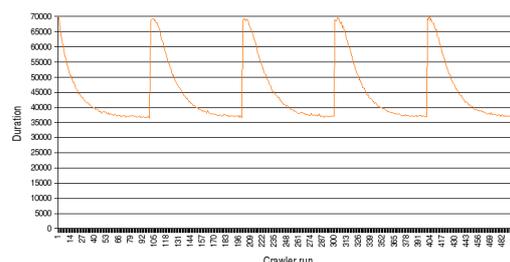


FIGURE 3. The TWOMA in a dynamic environment where the most optimal $P_m$ for each $O_i$ changes after each 100th crawler run.

Note that in this experiment we assume that the duration of accessing the objects (downloading the web sites and performing accessibility measurements) is static, even though the environment gives bias results.

An interesting result shown in figure 4 is that the numbers of the access points $|R|$ plays a significant role when attempting to reach optimal distribution compared to the numbers of the objects $|\Omega|$. This is an indicates that the algorithm scales well with regards to the number of web sites, but has rather limited scaling possibilities with regards to number of access points.
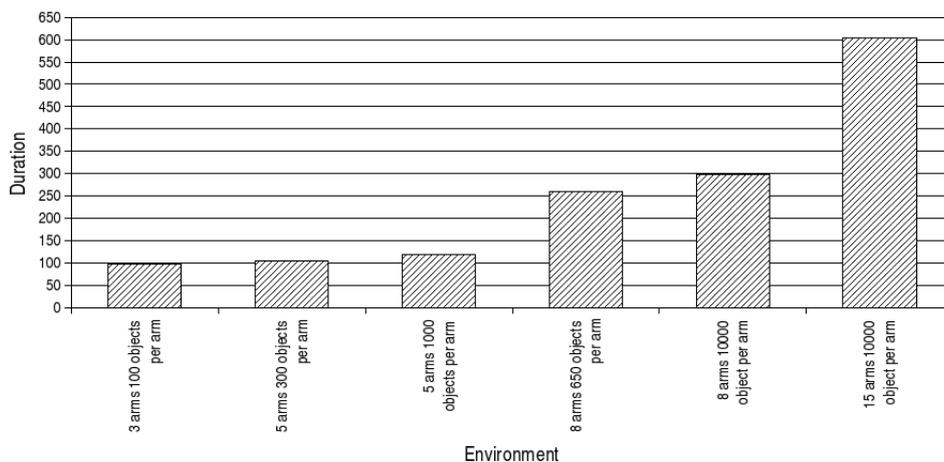


FIGURE 4. The number of crawls $t$ needed for all objects to reach the optimal access point.

## 5. FURTHER WORK

In our further work we aim to test the TWOMA in more complex environments including a smaller real web environment. Additionally, we attempt to extend the TWOMA for a solution that prioritises updated web resources and thus maximsing the number of updates detected in addition to minising the duration of the web crawls. Finally we intend to provide a formal proof of the TWOMA algorithm.

## 6. Conclusion

In this publication we have extended the OMA and developed a Time Weighted version of the algorithm. We have shown that the presented TWOMA may significantly reduce the duration of crawling and evaluating up to 10 000 web sites when the TWOMA interacts with a distributed web crawler compared to more traditional uniform distribution of web sites.

Most noticable the TWOMA is able to reduce the duration of each crawl by half in our experimental environment.

## References

[1] B. J. Oommen and D. C. Y. Ma, "Deterministic learning automata solutions to the equipartitioning problem," *IEEE Trans. Comput.*, vol. 37, no. 1, pp. 2–13, 1988.

[2] "European internet accessibility observatory." [Online]. Available: http://www.eiao.net/

[3] M. Snaprud, N. Ulltveit-Moe, O.-C. Granmo, M. Rafoshei-Klev, A. Wiklund, and A. Sawicka, "Assessment of public web sites accessibility - some early restults," *The Accessibility for All Conference*, 2003.

[4] M. Snaprud, M. G. Olsen, and F. Aslaksen, "Automatic benchmarking and presentation of the first results from the european internet accessibility observatory," *eChallanges ETSI*, 2006.

[5] "Google." [Online]. Available: http://www.google.com/

[6] S. Pandey, K. Ramamritham, and S. Chakrabarti, "Monitoring the dynamic web to respond to continuous queries," in *WWW '03: Proceedings of the twelfth international conference on World Wide Web.* New York, NY, USA: ACM Press, 2003, pp. 659–668.

[7] J. L. Wolf, M. S. Squillante, P. S. Yu, J. Sethuraman, and L. Ozsen, "Optimal crawling strategies for web search engines," in *WWW '02: Proceedings of the eleventh international conference on World Wide Web.* New York, NY, USA: ACM Press, 2002, pp. 136–147.

[8] J. Cho and A. Ntoulas, "Effective change detection using sampling," 2002. [Online]. Available: citeseer.ist.psu.edu/cho02effective.html

[9] J. Cho and H. Garcia-Molina, "Effective page refresh policies for web crawlers," *ACM Trans. Database Syst.*, vol. 28, no. 4, pp. 390–426, 2003.

[10] O.-C. Granmo, B. J. Oommen, S.-A. Myrer, and M. G. Olsen, "Learning automata-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 2006.

[11] ——, "Determining optimal polling frequency using a learning automata-based solution to the fractional knapsack problem," *Proceedings of the 2006 IEEE International Conferences on Cybernetics and Intelligent Systems (CIS) and Robotics, Automation and Mechatronics (RAM)*, 2006.

[12] S. A. Myrer and M. G. Olsen, "Incremental web crawler as a competitive game of learning automata," Master's thesis, Agder University College, 06 2005.

[13] B. J. Oommen and D. C. Y. Ma, "Fast object partitioning using stochastic learning automata," in *Proceedings of the Tenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New Orleans, Louisiana, USA, June 3-5, 1987.* ACM, 1987, pp. 111–122.

[14] V. Shkapenyuk and T. Suel, "Design and implementation of a high-performance distributed web crawler."

[15] "Unified web evaluation methodology version 0.5." [Online]. Available: http://www.wabcluser.org/uwem05/

[16] V. N. Padmanabhan and L. Qui, "The content and access dynamics of a busy web site: findings and implicatins," in *SIGCOMM*, 2000, pp. 111–123. [Online]. Available: citeseer.ist.psu.edu/padmanabhan00content.html

[17] B. E. Brewington and G. Cybenko, "Keeping up with the changing Web," *Computer*, vol. 33, no. 5, pp. 52–58, 2000. [Online]. Available: citeseer.ist.psu.edu/brewington00keeping.html

[18] D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener, "A large-scale study of the evolution of web pages," *Softw. Pract. Exper.*, vol. 34, no. 2, pp. 213–237, 2004.

[19] W. I. Sources, "Wic: A general-purpose algorithm for monitoring." [Online]. Available: citeseer.ist.psu.edu/712654.html

[20] A. Ntoulas, J. Cho, and C. Olston, "What's new on the web?: the evolution of the web from a search engine perspective," in *WWW '04: Proceedings of the 13th international conference on World Wide Web.* New York, NY, USA: ACM Press, 2004, pp. 1–12.