

# Distributing Resource Allocation based on Simulated User Interaction

TRABELSI WALID & MORTEN GOODWIN OLSEN

*Faculty of Engineering and science, Agder University College*

*Serviceboks 509, N-4898 Grimstad, Norway*

[Walidt06@student.hia.no](mailto:Walidt06@student.hia.no), [Morten.G.Olsen@hia.no](mailto:Morten.G.Olsen@hia.no)

## **Abstract:**

Due to the explosive growth of the Internet, web search engines are becoming increasingly important as the primary means of locating relevant information. This rapidly growing amount of data on the web leads to new challenges in order to allocated resources in distributed web. This is especially true if we keep in mind that any content should available as quickly as possible for the end users. This can be seen as a distributed resource allocation problem, as the crawler should minimize the time spent downloading of the web sites with limited resources when resources available vary between access points. In the same way, the search engine can lead to minimize the duration of accessing cached pages from the end users. In this paper, we study the use of the Object Migration Automaton (OMA) for Distributed Resource Allocation Problem. The objective of our approach is to position web sites downloaded utilizing the available resources as best as possible and thus minimizing the total duration of download and especially the duration of each requests from the perspective of the end users.

**Keywords:** Distributed Resource Allocation Problem, Object Migration Automaton OMA, Duration of accessing cached pages, Simulated environment.

## **1. Introduction**

**Problem Background:** The main part of a mechanism used for large scale web assessment machinery, e.g. as a web accessibility measurement tool [8] or a search engine, will be a web crawler (also known as web spider or web robot), possibly distributed. The web crawler can download as many updated web sites as possible compared to the locally stored copies and cached web pages and that the time used on any evaluation of the complete set of web sites is minimized as small as possible.

This can be seen as a Distributed Resource Allocation Problem, as the crawler should minimize the time spent downloading and evaluation of the web sites with limited resources available when the resources available vary between access points. As consequence the duration of accessing cached pages from the end users is minimized using a new partition pages in the best way within a distributed crawler.

In [1, 2], B.J. Oommen and D.C.Y Ma presented the Object Migration Automaton (OMA) as one of two theoretical stochastic learning Automaton solutions for the Object Partitioning Problem (OPP). OMA seems particularly promising for distributed resource allocation problem because it demonstrated an excellent partitioning capability that we can assume that resources should be partitioned on the best efficient.

**Goal:** Our claim here is that the Object Migration Automaton (OMA) can be used for

**License note:** This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Distributed Resource Allocation Algorithm to minimize the duration of accessing cached pages by measuring the duration of each access and in the same way to improve the web crawler to get the highly efficient crawling systems. Then, we suggest allocating the resources basing on the Object Migration Automaton (OMA) approach. Our solution is tested through a Java implementation and an evaluation by developing an algorithm for a simulated environment.

**Distributed Web Crawling:** With the explosive growth of the Internet, web search engines are becoming increasingly important as the primary means of locating relevant information. Thus, highly efficient crawling systems are needed in order to download the hundreds of millions of web pages indexed by the major search engines. In fact, search engines compete against each other primarily based on the size and currency of their underlying database, in addition to the quality and response time of their ranking function. Even the largest search engines, such as Google or AltaVista, currently cover only limited parts of the web, and much of their data is several months out of date. (We note, however, that crawling speed is not the only obstacle to increased search engine size, and that the scaling of query throughput and response time to larger collections is also a major issue [11]).

Two issues are addressed for a good crawler for large scale web assessment in [11]. 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 optimized 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.

**Format of the Paper:** In Section II we present our motivation by some requirements for the Automaton's functionality. We present a review of relevant literature in section III. The proposed design of a variant of the Object Migration Automaton (OMA) as a solution to the distributed resource allocation problem is then presented in section IV. In section V, we test and evaluate our solution. The discussion is found in section VI. We conclude the paper in Section VII and offer prospects for further work.

## 2. Motivation

The solution under development needs some requirements to work satisfactory, we have predefined a few requirements for the Automaton's functionality.

We need to create a simulated environment to test and evaluate our solution basing on the Object Migration Automaton (OMA). This environment is designed to represent the behavior of the real web as best as possible, both with regards of the behavior of web sites and access points. Further more, in order to simulate the duration of accessing cached web sites, we need to fix some parameters to be used. In this way, we choose to merge the state of the web site in the Automaton, the number of users accessed to the web site and finally we predefine for each web page, which is seen as an object in our environment, a perfect Arm in the Automaton as perfect distance that pages located close to a user (described later). Finally, The Automaton should be able to organize the objects basing on its actions: both objects are in the same Arm and concerning the time of accessing.

## 3. Object Migration Automaton (OMA)

The standard for evaluating whether the resource object is the most optimal situation can be different, such like time, distance and so on. Using the Object Migration Automaton (OMA) towards partitioning the resource objects to receive the most viable solution seems like a viable approach. In [1, 2], B.J. Oommen and D.C.Y Ma presented the Object Migration Automaton (OMA) as one of two theoretical stochastic learning Automaton solutions for the Object Partitioning Problem (OPP). 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 penalized. 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.

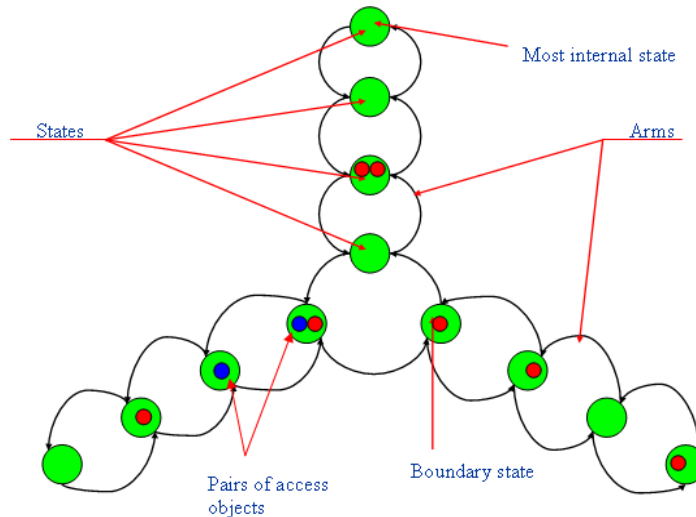


Figure 1: A visual representation of an OMA that have 3 arms with 4 states each

The Automaton can be seen as a set of  $R$  actions, each action representing a certain class named as an arm. Further for each action, there is a fixed number of a states  $N$ . Additionally, a set of abstract objects  $\Omega = \{O_1, \dots, O_w\}$  move around in the automaton. The state is an integer representing the place of the object  $O$  in its current arm. Figure 1 shows an example of the Object Migration Automaton (OMA) with three arms, nine objects and four states in each arm.

The Automaton as described below:

- Each object  $O_i$  is all the time located in one of the predefined  $N$  states and respectively connected to one of the predefined  $R$  actions.
- Each object  $O_i$  has an action  $a_i$ . This action is dependent on which of the  $R$  arms the object is located in.
- All objects are accessed in pairs  $(O_i, O_j)$  and whenever two  $O_i$  and  $O_j$  are accessed they will always be penalized 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 penalized, it moves one state closer to the boundary state in its corresponding arm. If the object 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 penalized if  $O_i$  and  $O_j$  are accessed together and are located in different arms.
- If  $O_i$  and  $O_j$  penalized and at least of of them are located boundary states of different arms, they switch arms and thus change their action.

#### 4. Design of our solution to distributed Resource Allocation Problem

The heart of the algorithm uses then the Object Migration Automaton (OMA) as a solution to the Distributed Resource Allocation Problem. We have extended this approach and we have used the duration of accessing of each object as basic for partitioning in addition to using the order of which the objects are accessed. The objective is in general to position the downloaded cache such that the total duration of accessing any cached page is minimized. In our experiments, we consider duration of accessing any page as a combination of the time spent to download the web page the distance that separates the end user from cached pages.

Before we describe our assumption of the choice of actions, we present the mapping between the developed algorithm and a real web crawler in the table 1.

*Table 1: Mapping between a simulated environment and a real web crawler*

Symbol	Simulated Environment	Web Crawler
$\Omega = \{O_1, \dots, O_w\}$	Objects	Web pages
$R = \{A_1, \dots, A_n\}$	Arms	Access points
N	Internals states	Evaluation duration
T	Time	Duration of accessing

First of all, the objects are randomly placed in the innermost states of the Automaton. The Automaton has a predefined set of starting arms, each with a predefined length. After the corpus has been inserted in the different states, the Automaton can start its process.

The Automaton runs in one specific loop until the Automaton reach the most optimal position of all objects, with an iteration defined as one test and the following action taken based upon this test. A general iteration is outlined below, in figure 2.

Unlike the basic rules of Object Migration Automaton (OMA) that each Object  $O_i$  in the corresponding Arm  $A_j$  and is located in one of the N states, an object may only move from  $N_i$  to either  $N_{i-1}$  or  $N_{i+1}$ . Our scheme involves accessing web sites while measuring the duration of accessing cached pages.

It is noticeable that in our solution, an object will always have a perfect Arm considered as the distance separated the user from pages located close to a user. If the object is located in the perfect Arm the duration of accessing will in average be less than if the object was located in any other Arm than the perfect Arm. The perfect Arm for all the objects is predefined in the learning process and it is assumed as a distance that can take the web pages between all the different access points. In a large scale search engine/ crawler, it is evident that pages located close to a user, can more quickly be accessed, especially when we consider the limited resources that restricts any crawler. e.g. for a European user, it is more beneficial when it comes to time, to read a

cached page stored in Europe than e.g. the same page cached in a server in Asia.

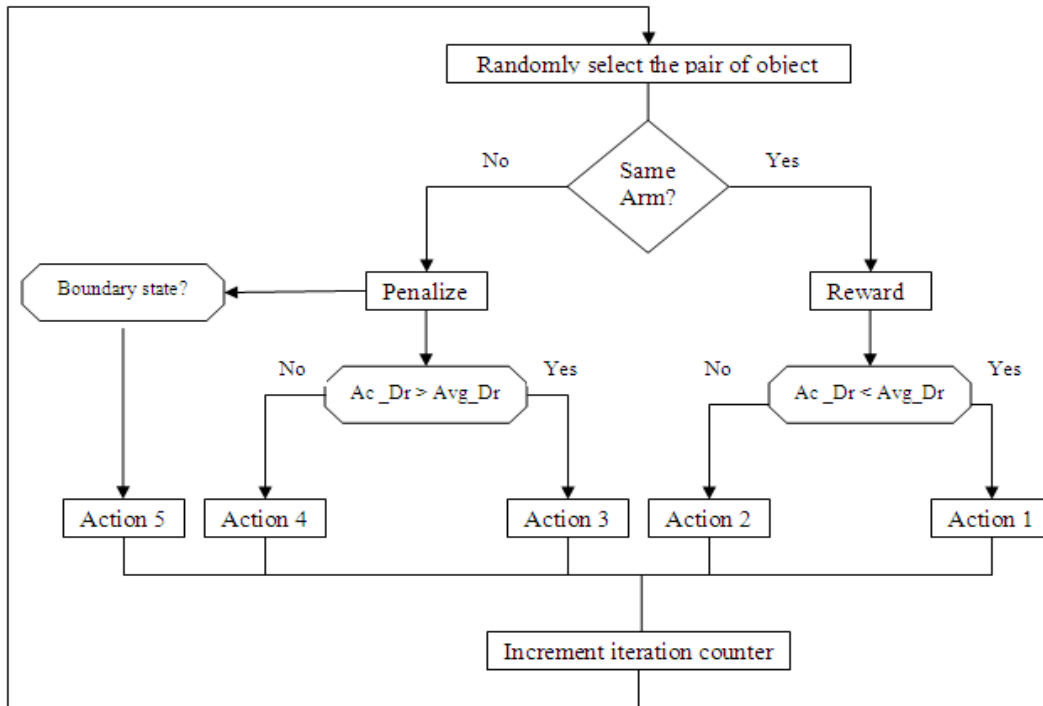


Figure 2: The Automaton Process based on the time of download

In our case, in order to handle the time of download, we had to change the actions which the OMA originally use for the fast object partitioning problem, presented in [1, 2].

Still, we base our actions on the two simple tests:

1-Are the pairs of objects accessed in the same arm?

2-Compare the time of download of the access objects with the average time of all objects in the corresponding Arm.

The general assumptions of the choice of actions take firstly the localization of the accessed pairs of objects as a basic in the Object Migration Automaton (OMA) approach. In fact, if the objects accessed in pair are in the same Arm then the action should be rewarded, in the other case that the objects accessed in pair are in different Arm then the action should be penalized. To extend the OMA approach, we involve accessing web sites measuring the time of download of each object access and we compare this with the average time of all objects in the corresponding Arm.

Then in each time  $t$  each object  $O_i$  is accessed and the duration of download is measured. The main rules of actions for each crawler run  $t$  are outlined below:

- If the pairs of objects are in the same Arm then the objects are rewarded and in the same way we calculate the time of download of each one of them:

**Action 1:** If the time of download is less than the average duration of all objects in the corresponding Arm, the object jumps two states closer to the most internal state.

**Action 2:** If the time of download is more than the average duration of all objects in the corresponding Arm, the object only moves one state closer to the most internal state.

- If the pairs of objects are in different Arm then the objects are penalized and in the same way we calculate the time of download of each one of them.

**Action 3:** If the time of download is more than the average duration of all objects in the corresponding Arm, the object jumps two states closer to boundary state.

**Action 4:** If the time of download is less than the average duration of all objects in the corresponding Arm, the object moves only one state closer to the boundary state.

**Action 5:** If the pairs of objects are penalized and at least one of them is located in the boundary state, they switch arms and thus change their action.

- Any object that is already located in the most internal state and rewarded then it doesn't move.

Note we are working with an unknown environment. This means that the algorithm can only know the duration of download by actually performing a download. In other word, in order to represent the environment presented above, each web site in our environment is seen as an object with a random stochastic variable. This means that the time of download is represented as a random variable that can only be retrieved after the object has been accessed and the crawler is running.

We can mention that the heart of our scheme involves also the accessing web sites while verifying the web site change in each access. Then we take advantage of the excellence partitioning capability in [1, 2, 6] to estimate the frequency of change of the web sites. When the web sites, assumed as the objects in the Automaton, are partitioned following the Object Migration Automaton (OMA) approach we can profit from this by developing a function to detect in each access if the web site has changed.

## 5. Evaluating and Testing

In this section, we evaluate our solution and the presented algorithm. In all the test results, the iterations to reach the optimal solution are the mean value of 8 test runs per test. Then all results presented are the outcome of averaging 8 runs.

Note that according to the goal's algorithm to minimize the duration of accessing cached pages, the optimal solution is when the time of accessing of each object is less a fixed time. This fixed time is taken as an average time considering that it can reach the sufficient of the users.

In the first test we used ninety objects per arm, three arms and an arm length of 8 states. We present in the figure 3 the performance of the algorithm to minimize the duration of accessing compared to the initial measurement of time. In the figure 3 the behavior of the algorithm in a static environment with 3 arms and 100 objects is shown. Note that the x-axis shows the number of iterations as the total number crawler runs, while the y-axis shows the time of accessing of all objects.

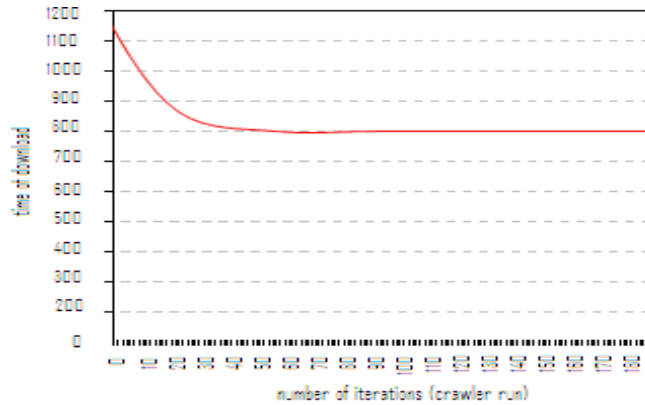


Figure 3: Duration of accessing behavior

This experiment shows that the behavior of the algorithm starts out having an initial time of accessing of the objects, but quickly reduces the time of download of each crawl as the number of iterations increases. It reaches the optimal solution after about 40 iterations.

Note that in this experiment we assume that the time of download is static as we work with an unknown environment.

The next test is done to explain and justify why in the Automaton's actions move sometimes 1 state and sometimes 2 states. First of all, all the abstract objects moves around the automaton at least one state, the automaton is either rewarded or penalized. This action is dependent on which of the pair of objects are in the same arm as the base of the Object Migration Automaton (OMA) approach. To extend this approach, the Automaton's actions depend also on the time of download of each object. That is why it should move another state in our solution. However, the question that we can put it is why we choose to move 2 states?

To justify our proposition, we evaluate the same test used in the figure 4 when the Automaton's actions move 3 or 4 states instead of moving 2 states. We combine the result in the same figure 4 to show the different behavior of each one of them. The figure 4 shows that our proposition to move only 2 states has a behavior better than moving 3 or 4 states because it reaches the optimal solution in less number of iterations. This can be explained by the limited number of states used by the Automaton, 8 or 10 in each arm.

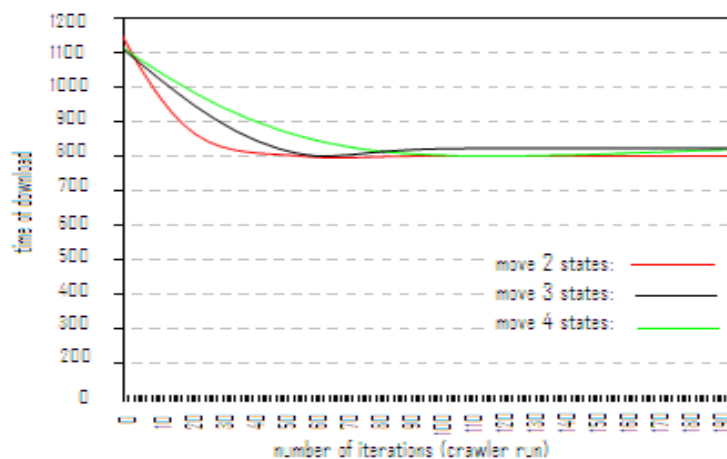


Figure 4: Behavior for different moving states

The figure 5 shows the number of cached pages (the objects) plays an important role to reach the optimal solution. Thus, the Automaton needs more iteration to attend the optimal solution

when the numbers of pages (the objects) increases. That's why we think to increase also the number of access points (arms).

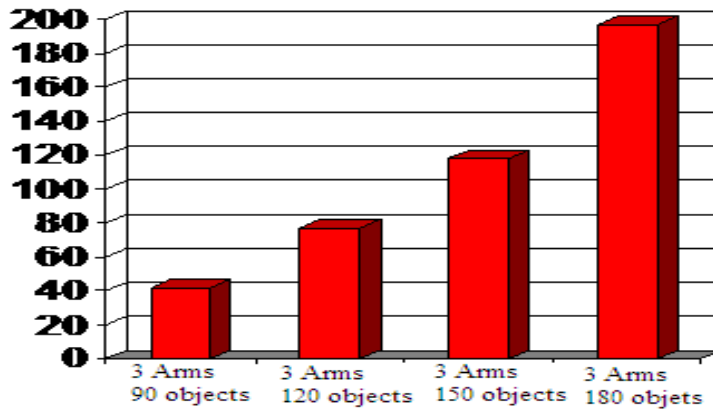


Figure 5: Role of number of objects

Another interesting result shown in figure 6 is that the numbers of access points (arms) plays a significant role to reach the optimal solution compared to the numbers of cached web (the objects). The figure 6 shows the different behavior and the relationship between the objects and the corresponding arms.

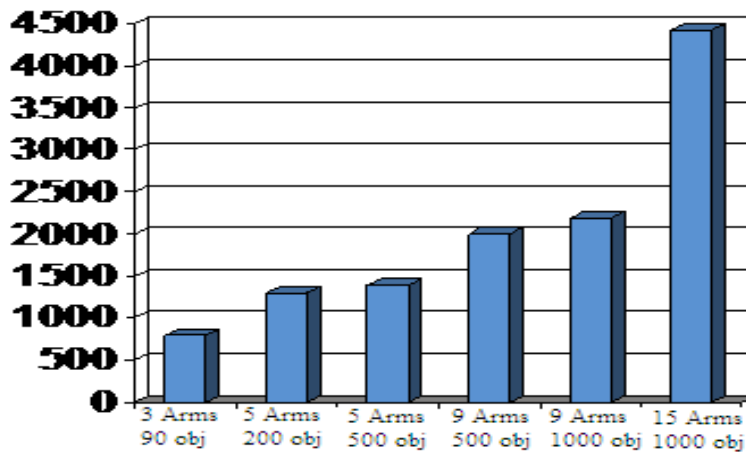


Figure 6: Role of the Access points

## 6. Discussion and Further work

Our initial task was to see whether the OMA was a good alternative to use in Distributed Resource Allocation Problem. We have found that it works very well to minimize the duration of accessing with small number of cached web (objects), but not so good with a large number of cached web. It needs a big number of iterations and more access points (arm) to reach the optimal solution.

We believe this can be explained by the explosive size of the Internet and in the same way the growth of users. That is why we think that we can take profit of our solution working very well with small number of web sites (objects). The new idea is to combine our scheme with clustering approach. This makes it a lot easier to collect the large number of web sites in small set of cluster. After that we apply Object Migration Automaton (OMA) approach to the small number of cached web closer to a user.

We have found also that Object Migration Automaton (OMA) is a good solution to improve a web crawler by minimize the duration of accessing cached pages from the end users. According to the big area of Distributed Resource Allocation Problem leads us to think if we can use our solution in Telecoms domain. It is interesting to handle frequencies allocation or channels allocation especially in GPRS and UMTS.

## 7. Conclusion

In this paper, we have presented a working solution for Distributed Resource Allocation problem using the Object Migration Automaton (OMA). The Automaton is able to handle the web sites (objects), storing them in the system and making them moving around the automaton.

We have extended the Object Migration Automaton (OMA) and developed a new algorithm. Evaluating and experimental results have demonstrated that our solution can be used for handling the Resource Allocation Problem. Our aim was in general to position the downloaded cache such that the total duration of accessing any cached page is minimized.

We also learned in the development process that for this type of solution depends of the number of cached web and also depends of the numbers of access points. The solution described has shown to handle a small number of web sites very well, but it has some problems when it comes to large number of web sites. Clustering is a proposed solution described in section VI as a further work.

Our conclusion is that OMA is very suitable for Resource Allocation Problem. As it described in the problem description, OMA is a good solution to handle the limited resources. It optimizes their allocation in web monitoring to be used in the best efficiency. Our successful experiments have shown that it was able to partitioned pages within a distributed crawler so that both download duration of web pages and the duration of accessing the cached pages is minimized.

We believe that OMA may turn out to be a valuable tool when it comes to combining our solution with clustering approach to handle a large number of web sites. The last question that we can put it is: can we use our solution in Telecoms domain to handle frequencies allocation and Channel allocation in GPRS and UMTS?

## 8. References

- [1] B. J. Oommen, D. C. Y. Ma. Fast Object Partitioning Using Stochastic Learning Automaton.
- [2] B. J. Oommen D. C. Y. Ma Deterministic Learning Automata Solutions to the Equipartitioning Problem.
- [3] J.L. Wolf, M.S. Squillante, P.S. Yu, J. Sethuraman, L. Ozsen. Optimal Crawling Strategies for Web Search Engines.
- [4] Junghoo Cho, Alexandros Ntoulas Effective Change Detection Using Sampling.
- [5] Junghoo Cho, Hector Garcia-Molina Estimating Frequency of Change.
- [6] Leiming Chen, Dongmei Wang, Morten Goodwin Olsen. Towards distribution of web sites in a crawler used for large scale web accessibility assessment.
- [7] Morten Goodwin Olsen. Introduction to: Web Crawling Techniques and Resource Allocation
- [8] M. Snarud, M. G. Olsen, and F. Aslaksen. Automatic benchmarking and presentation of the first results from the european Internet accessibility observatory," eChallenges ETSI, 2006.
- [9] Ole-Christoffer Granmo, B. John Oommen, Svein Arild Myrer, Morten Goodwin Olsen. Determining Optimal Polling Frequency Using a Learning Automata-based Solution to the Fractional

Knapsack Problem.

[10] Ole-Christoffer Granmoy, B. John Oommen, Svein Arild Myrer, Morten Goodwin Olsen Learning Automata-based Solutions to the Nonlinear Fractional Knapsack Problem with Applications to Optimal Resource Allocation

[11] Vladislav Shkapenyuk, Torsten Sue. Design and Implementation of a High Performance Distributed Web Crawler.