



Learning Automata-based Solutions to the Nonlinear Fractional Bin Packing Problem with Application to distributed web crawling

by

Anis Yazidi

Supervisors: Morten Goodwin Olsen
Ole-Christoffer Granmo

Project report for IKT 407 in Spring 2007

Agder University College
Faculty of Engineering and Science

Grimstad, 14 May 2007

Status: Final

Keywords: Distributed crawler, Learning automata, Bin packing problem, Stochastic environment, Knapsack problem.

Abstract:

In this project, we study the use of learning automata to solve the nonlinear fractional bin packing problem and its application to distributed web crawling.

Recent studies has been performed to solve the the so called nonlinear fractional knapsack problem. The results were really fascinating and demonstrate the strength of learning automata when applied to centralized resource allocation problems. Due to the ever growing deployment of distributed systems, it becomes imperative to extend the later studies to cope with a distributed context. The obvious and direct extension of the non linear fractional knapsack problem is the non linear fractional bin packing problem. In this paper, we present two novel approaches for solving the non linear bin packing problem. We relate this problem to resource allocation problems and we show that it has applications within the field of distributed crawling . An appropriate mapping of the nonlinear fractional bin packing problem to distributed polling frequency determination problem is considered in this paper.

Comprehensive experimental results have showed the superiority of our proposed schemes when applied to distributed web crawling.

[This work is licensed under the Creative Commons Attribution-ShareAlike License \(http://creativecommons.org/licenses/by-sa/2.5/\).](http://creativecommons.org/licenses/by-sa/2.5/)

Table of Contents

1	Introduction.....	3
2	Problem description.....	4
3	Background.....	5
4	Solution.....	8
	1.1 Requirements	8
	1.2 Design Specification	8
	1.3 Implementation.....	13
	1.4 Validation and Testing.....	16
	Discussion.....	20
5	Conclusion.....	21
	References.....	22

1 Introduction

Crawlers are core elements of many services running on the Internet. In fact, by keeping a local copy of the web data, they facilitate the search of relevant information.

One of the key challenges of a web crawler is to maintain its repository up to date. However, due to limitations of bandwidth and computer resources, this task is not feasible in a timely manner. Therefore the crawler should decide carefully which pages to download at what time. Indeed, downloading an unchanged page leads to a waste of valuable crawling resources. Furthermore, the dynamicity of the web put additional burden on the crawling process. In fact, the web pages are often updated with different periods, which makes extremely challenging to choose the right time to download.

In [13], Narendra and Thathachar presented the learning automaton as a solution to finding the optimal action in a random environment. Since this time, learning automata has gained a significant attention. Learning automata seems promising for our project because it was shown highly effective when operating in dynamic environments.

In this paper, we consider the problem of optimizing the allocation of distributed web crawler resources when the polling capacity is limited. Our proposed paradigm underlies on mapping the problem to a novel version of the Bin packing problem where the characteristics of items are unknown. This assumption seems to be realistic when operating in dynamic environments such as the Web. Our scheme involves connecting every item with a team of learning automata performing a random controlled walk on a discredited solution space.

To the best of our knowledge, this problem has not been addressed before in the literature. In the field of distributed crawlers, most of the work [11, 12] in the area mainly focused on the architecture design rather than specific resource allocation approaches. In [10], Cho and Garcia-Molina investigated policies to assign URLs to crawlers in a manner that avoids repeated downloads of the same page.

A similar resource allocation approach to our work has been studied in [8]. The authors of this publication modelled the problem of resource allocation in web as a knapsack problem with unknown properties of the materials. However, they addressed only the centralized version of the problem and their study did not embrace the distributed version. In this project we join the claim of [8, 9] stating that learning automata can efficiently estimate the update frequencies of web pages.

The rest of the report is organized as follows.

In section 2: we present the problem description of our project.

In section 3: we survey a necessary background for the understanding of the project.

In section 4: a detailed description of the solution, its implementation and the conducted experiments are presented.

In section 5: we discuss and evaluate the proposed solution. A further work is also considered in this section.

In section 6: we briefly draw conclusions from our project.

2 Problem description

In this project, we address the problem of optimizing the allocation of distributed web crawler resources when the polling capacity is limited.

The problem seems captivating and intriguing since a possible solution to it can provide benefits to a wide range of resource allocation applications.

Our work is mainly motivated by the need of a distributed crawling policy to manage large scale web data. In fact, the problem has arisen remarkably these last years with the dramatic growth of the internet. However, few studies have been performed in the area of distributed crawlers.

The proposed approach centres on mapping the problem to a novel version of the Bin packing problem where the characteristics of items are unknown. We should underline that the items here are the web pages to monitor, and their weights are guesses of the amount of updated pages.

In order to estimate the appropriate polling frequency of the web pages we make use of learning automata.

3 Background

3.1 Web Crawling Policies

A survey of the literature discerns two types of web crawlers namely batch and incremental crawlers. A batch crawler polls the monitored web pages with the same period regardless of how often they change [1, 2, 3]. This blind scheme is shown to be sub optimal. In fact, the proposed scheme wastes the monitoring capacity by downloading unchanged data. More work was invested in developing more efficient crawling policies leading to the advent of incremental crawlers. In contrast to a batch crawler, an incremental crawler updates its repository based on the frequencies at which the pages change. In regards to performance, incremental crawlers attain higher freshness of the local copy of the monitored data sources and optimize significantly the use of the available crawling resources.

The notion of incremental crawler was first introduced by Junghoo Cho and Hector Garcia in [4]. Their claim is that the web pages updates can be modelled as Poisson processes. Furthermore, in [5] they build estimators for the web pages changes underlying on this assumption. However such estimation of web pages changes requires having a history of the updates which is not practical in many cases. In addition, in [5] it is stated that their Poisson model does not handle a large set of web pages, namely highly and slowly changing web pages. Moreover, they assumed that the change frequency is static which may not be valid in certain cases.

The reliability of the Poisson model has been the subject of critical analysis conducted by Brewington and Cybernenko in [6]. In fact, they showed that a large amount of tested web pages does not follow the Poisson Process.

J. Cho and A. Ntoulas proposed a sampling-based policy in [7] for the incremental crawler. The approach relies on estimating the number of web pages updated in a web site based on using random samples. Hence, this approach draw conclusions exclusively from the samples available in the current cycle and does not make use of the update pages history, although it represents a relevant information susceptible of yielding better accuracy.

Recent studies [8, 9] have regarded incremental crawling as a continuous learning process. They mapped the problem of optimizing the frequencies of revisiting the monitored web pages under restricted capacity constraints to a version of knapsack problem where the volume of items are variable, following unknown distribution. These works [8, 9] showed that learning automata seem particularly promising when applied in dynamic environments such as the Web.

3.2 Bin Packing Problem

The bin packing problem is one of the famous NP hard problems that arises in a variety of practical problems. The bin packing problem has been described as a set of items of different volume must be packed into a finite number of bins in a way that minimizes the number of used bins [20].

Numerous heuristic solutions have been proposed to the bin packing problems. [21] presents various algorithms to solve the problem in a near optimal way. Empirical and analytical results showed the superiority of the Best Fit Decreasing compared to other packing algorithm.

Because of this, we make use of the Best Fit Decreasing algorithm in our first approach. The Best Fit decreasing algorithm is performed by first sorting the items into decreasing order then packing one at a time in the bin that have the least amount of space left. If no bin matches, we open a new bin and put the item into this.

Several variants of the bin packing problem have been studied in the literature [22, 23, 25, 26]. In the interest of posing our problem in the right perspective, we *briefly* catalogue them below.

The Linear Fractional Bin Packing Problem:

The linear FB packing problem is a classical continuous optimization problem which also has applications within the field of resource allocation. The problem involves n materials of different value v_i per unit volume, $1 \leq i \leq n$ and m bins of different volume C_j , $1 \leq j \leq m$. Each material is available in a certain amount x_i such as $x_i \leq b_i$. Let $f_i(x_i)$ denote the value of the amount x_i of

material i , i.e., $f_i(x_i) = v_i x_i$. The problem is to fill the m bins with the material mix $x = [x_1, \dots, x_n]$ of maximal value $\sum_{i=1}^n f(x_i)$, without violating the capacity constraint of each bin.

The Nonlinear Equality Fractional Bin Packing Problem:

One important extension of the above classical problem is the *Nonlinear Equality FB* problem with a separable and concave objective function. The problem can be stated as follows :

$$\begin{aligned} & \text{Maximize} && \sum_{i=1}^n f(x_i) \\ & \text{subject to} && \sum_{j=1}^m y_{ij} = 1, \quad 1 \leq i \leq n \\ & && y_{ij} \in \{0,1\}, \quad 1 \leq i \leq n, \quad 1 \leq j \leq m \\ & && \sum_{i=1}^n y_{ij} x_i \leq C_j, \quad 1 \leq j \leq m \\ & && x_i \geq 0, \quad 1 \leq i \leq n \end{aligned}$$

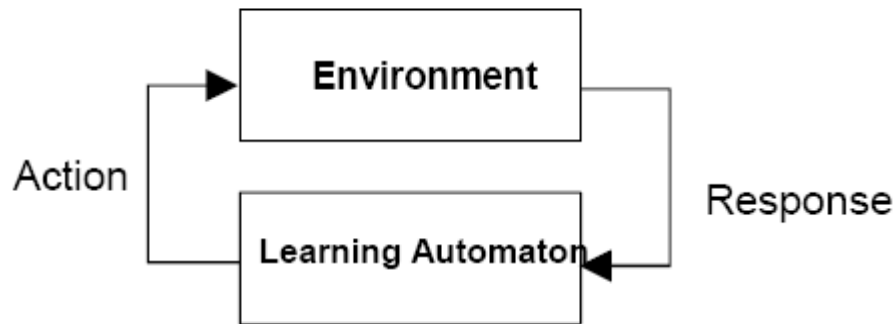
Where y_{ij} is a binary variable such as $y_{ij} = 1$ if the amount of material x_i is assigned to the Bin j , and $y_{ij} = 0$ otherwise. The condition $\sum_{j=1}^m y_{ij} = 1$ assures that every amount of material x_i is assigned to exactly one bin.

Since the objective function is considered to be concave, the value function $f_i(x_i)$ of each material is also concave. This means that the derivatives of the material value functions $f_i(x_i)$ with respect to x_i are non-increasing. In other words, the material value *per unit volume* is no longer constant as in the linear case, but decreases with the material amount.

3.3 Learning automata

The notion of learning automata has been first introduced by M.L Tsetlin [15]. Since that time, a significant amount of work has been reported in the area of learning automata. The appeal of learning automata is its ability to build schemes that mimic human behaviour when acting in a non stationary environment.

Learning automaton is defined as a finite state machine that interacts with a random environment and learns continuously the optimal actions based on the feedback of the environment.



Learning automata acting in an environment

To formally describe a learning automaton we will use the definition stated in [24]. This definition is also commonly used in other literature. The automaton is defined mathematically by the quintuple $\{ \Phi, \alpha, \beta, F(\cdot, \cdot), G(\cdot, \cdot) \}$

1. The state of an automaton at any instant n , denoted by $\varphi(n)$, is an element of the finite set $\Phi = \{ \varphi_1, \varphi_2, \dots, \varphi_s \}$

2. The output or action of an automaton at the instant n , denoted by $\alpha(n)$, is an element of the finite set $\alpha = \{ \alpha_1, \alpha_2, \dots, \alpha_r \}$

3. The input of an automaton at the instant n , denoted by $\beta(n)$, is an element of the set β . This set could either be a finite set or an infinite set, such as an interval on a real line. $\beta = \{ \beta_1, \beta_2, \dots, \beta_m \}$ or $\beta = \{ (a, b) \}$ where a, b are real numbers.

4. The transition function $F(\cdot, \cdot)$ determines the automaton state at the instant $(n+1)$ in terms of the state and input to the automaton at the instant n , i.e, we get $\varphi(n+1) = F[\varphi(n), \beta(n)]$. F can also be described as a mapping from $\Phi \times \beta \rightarrow \Phi$ and can either be deterministic or stochastic.

5. The output function $H(\cdot, \cdot)$ maps the current state and input into the current output. However, if the current output depends on only the current state, the automaton is referred to as a state output automaton. In such a case we replace $H(\cdot, \cdot)$ with $G(\cdot) \Phi \rightarrow \alpha$ Also expressed as $\alpha(n) = G[\varphi(n)]$ and can also either be stochastic or deterministic.

The automaton is stochastic if either one of the F and G mappings are stochastic. There are two types of stochastic automaton:

- Fixed structure stochastic LA
- Variable structure stochastic LA

In our project we make use of fixed structure automaton. More details about the design of such an automaton will figure in the design chapter.

4 Solution

1.1 Requirements

- We need first to establish the theoretical findings of our Nonlinear Fractional Bin Packing Problem. This includes defining formally the problem.
- We have to design a scheme susceptible to provide a solution to our defined Nonlinear Fractional Bin Packing Problem.
- The solution should be based on learning automata as it is imposed in the project definition.
- Two approaches to solve the problem should be presented: One online and the other offline.
- An appropriate mapping of the Nonlinear Fractional Bin Packing Problem to the distributed polling frequency determination problem should be considered.
- We should define the environment model with which the automata will interact.
- We need to set a simulation environment to test and evaluate the approach. Thus, we should implement the simulation environment using a suitable programming language.
- The computational implementation of the approach must be able to handle large web data sets.
- The algorithm should empirically converge in a reasonable time to a near optimal solution.
- We should compare the efficiency of our crawling solution with the traditional crawling policies.

1.2 Design Specification

a. Overview of the Solution

We assume that the material unit volume values are *random* variables $\{V_1, \dots, V_n\}$ with unknown distributions and we suppose that distribution of the material value per unit volume depends on the amount x_i . Therefore, the value $F_i(x_i)$ of the amount x_i of material i is a stochastic function: $F_i(x_i) = V_i \times x_i$.

Let $F'_i(x_i)$ denote the stochastic function that determines the unit volume value of material i given material amount x_i . $F'_i(x_i)$ takes the value 0 with probability $p_0^i(x_i)$ and the value 1 with probability $p_1^i(x_i)$. Our NEBK problem can be defined as an optimization problem

$$\begin{aligned} & \text{Maximize} \quad \sum_{i=1}^n E(F_i(x_i)) \\ & \text{Subject to} \quad \sum_{j=1}^m y_{ij} = 1, \quad 1 \leq i \leq n \\ & \quad y_{ij} \in \{0, 1\}, \quad 1 \leq i \leq n, \quad 1 \leq j \leq m \\ & \quad \sum_{i=1}^n y_{ij} x_i \leq c_j, \quad 1 \leq j \leq m \end{aligned}$$

$$x_i \geq 0 \quad 1 \leq i \leq n$$

Where y_{ij} is a binary variable such as $y_{ij} = 1$ if the amount of material x_i is assigned to the Bin j , and $y_{ij} = 0$ otherwise.

The condition $\sum_{j=1}^m y_{ij} = 1$ assures that every amount of material x_i is assigned to exactly one bin.

Note that we allow only instantiations of the material value per unit volume $F^i(x_i)$ to be observed. That is, each time an amount x_i of material i is placed in one of the bins, an instantiation of $F^i(x_i)$ at x_i is observed. x_i is assumed a number in the interval $(0; 1]$.

The crucial issue that we have to address is that of determining how to change our current guesses on $x_i, 1 \leq i \leq n$. We shall attempt to do this in a discretized manner by subdividing the

unit interval into N points $\{ \frac{1}{N^\alpha}, \frac{2^\alpha}{N^\alpha}, \dots, \frac{(N-1)}{N^\alpha}, 1 \}$, where N is the resolution of the learning

scheme and $\lambda > 0$ determines the linearity of the discretized solution space. We will see that a larger value of N will ultimately imply a more accurate solution to the knapsack problem.

Approach 1:

This approach is an offline approach. Its key finding is solving the problem as a knapsack problem then inferring this solution to the bin packing problem.

Given a set of m bins of different volume $c_j, 1 \leq j \leq m$ the derivate knapsack problem would involve one knapsack of a total capacity $C = \sum_{i=1}^m c_i$. A solution of this Knapsack problem with

the capacity $C = \sum_{i=1}^m c_i$ is an upper boundary solution for the Bin packing problem. In fact, the Knapsack problem is a relaxed version of the Bin packing problem with less constraints.

The scheme runs for a certain number of epochs. Each epoch involves a fixed numbers of iterations, N_∞ . N_∞ is chosen so as to be reasonably sure that the scheme "locally" converges after it has run for N_∞ iterations.

At each epoch k , we solve the knapsack problem with a capacity C_k accordingly to the the same paradigm defined in [8,9]. Where C_k is defined as follows:

$$C_k = \begin{cases} \sum_{i=1}^m c_i & \text{if } k=0 \\ C_k = \eta * C_{k-1} & \text{if } k>0, \text{ where } \eta \text{ is a parameter approximately equal to } 1, \eta \text{ is typically equal to } 0.99. \end{cases}$$

In simplest terms, for each epoch k , C_k can be deduced using this formula, $C_k = \eta^k * C_0$,

where $C_0 = \sum_{i=1}^m c_i$.

In the end of each epoch k , the material mix $x = [x_1, \dots, x_n]$ will converge to an optimal solution to the knapsack problem with a capacity C_k .

Nevertheless, this solution might be unsuitable to the bin packing problem due to violating the capacity constraints. Therefore, in the end of each epoch, we perform best fit decreasing algorithm in order to pack the material mix $x = [x_1, \dots, x_n]$ in the set of bins $\{b_j, 1 \leq j \leq m\}$.

We should underline that we chose best fit decreasing algorithm because it is merely among the simplest and most efficient heuristic algorithms for solving the bin packing problem.

If the mixture can be packed, then we say that our scheme has converged and the near optimal solution of the bin packing problem is the material mix x obtained after running the last epoch.

In each epoch k , the environment provides a signal ϕ_k indicating whether the equivalent "knapsack" in the epoch k is full.

$$\phi_k = \text{true if } \sum_{i=1}^n x_i \geq C_k$$

$\phi_k = \text{False otherwise}$

We should underline that C_k is updated accordingly to the rules previously mentioned above.

Let LA_i be the automaton connected to the each material i and let $s_i^k(t)$ be its current state in the epoch number k .

In regards to these notations the amount $x_i^k(t)$ would be equal to $\frac{s_i^k(t)^\alpha}{N^\alpha}$

Assume that $V_i^k(t)$ and $\phi_k(t)$ are the resulting feedback from the stochastic Environment, where k refers to the epoch and t states for the iteration number within this epoch. Then the state of the automaton is updated as follows:

$$s_i^k(t+1) := s_i^k(t) + 1 \quad \text{If } V_i^k(t) = 1 \quad \text{and} \quad 1 \leq s_i^k(t) < N \quad \text{and not } \phi_k(t)$$

$$s_i^k(t+1) := s_i^k(t) - 1 \quad \text{If } V_i^k(t) = 0 \quad \text{and} \quad 1 < s_i^k(t) \leq N \quad \text{and } \phi_k(t)$$

$$s_i^k(t+1) := s_i^k(t) \quad \text{Otherwise.}$$

At epoch k , the presented LA game creates a stochastic competition between the n automata that directs the automata towards the optimal solution to the fractional knapsack problem with the capacity C_k .

Approach 2:

The Nonlinear Equality FK (NEFK) Problem with a separable and concave objective function defined in oommen publication as

$$\begin{aligned} & \text{Maximize } \sum_{i=1}^n f_i(x_i) \\ & \text{subject to } \sum_{j=1}^m x_j = c, \quad 1 \leq i \leq n \end{aligned}$$

$$x_i \geq 0 \quad 1 \leq i \leq n$$

Under these assumptions, this problem can be solved using the principles of Lagrange multipliers.

The optimal solution for the non linear knapsack problem occurs when the derivatives of the material $f_i(x_i)$ value functions are equal, subject to the knapsack constraints.

$$f_1(x_1) = \dots = f_n(x_n)$$

$$\sum_{j=1}^m x_i = c, 0 \leq x_i, i = 1, \dots, n.$$

In the light of this, given an configuration of the items x_i spread among m bins, an optimal solution to this given configuration implies dividing the bin packing problem into m separate knapsack problems and solving each problem separately using the principle of Lagrange multipliers.

Obviously, the non linear fractional problem is a NP hard problem. In fact, given m bins and n items ($n > m$) there are C_m^n packing combinations (or configurations). A rough solution to the non linear bin packing problem would be to just check all C_m^n possible packing configurations and solving for each packing combination m separate nonlinear fractional problem using the principles of Lagrange multipliers. In other words, an optimal solution to the fractional bin packing problem would explore roughly $m \times C_m^n$ knapsack problems. In our problem using Lagrange multipliers is not viable whenever $f_i(x_i), 1 \leq i \leq n$, are unknown and only instantiations of the material value per unit volume can be observed.

In contrast to the first approach, our proposed second approach is online. This approach runs for a certain number of epochs that should be chosen enough large so that the scheme converges. To start with, we spread randomly the n materials x_i among the m bins. Each bin is regarded as a knapsack and contains a subset of the set of the items $\{x_i, 1 \leq i \leq n\}$. At each epoch, running Oommen algorithm [8,9] in every bin moves the automata towards an optimal solution that depends on the items configurations. Each epoch involves N , N is chosen relatively small to avoid adding burden on the convergence speed. We should highlight that running Oommen algorithm [8,9] for a given configuration during an large enough number of iterations achieves an optimal solution for the current items configuration that is not necessarily an optimal solution to the bin packing problem due to the fact that there are C_m^n possible packing configuration. Therefore, we should define a scheme that improves incrementally the global solution by changing the items configuration after each epoch.

In order to attain this purpose, we compute the expected value contained in the Bin at the end of each epoch. Formally this expected value in Bin j is $\sum_{i=1}^n y_{ij} E(F_i(x_i))$.

One greedy scheme for moving the web site that intuitively improves the solution is moving the item with the smallest expected value from the bin with the largest expected value to the bin with the lowest expected value. The idea is to utilize the capacity in all bins better and thus increase the sum of the estimated values for all bins. Therefore, this scheme will incrementally improve the global solution for an epoch to another moving towards the optimal solution.

We attach to every bin j , at the each epoch k , the feedback from the environment called ϕ_j^k indicating whether the "bin j " is full.

$$\phi_j^k = \text{True if } \sum_{i=1}^n y_{ij} x_i \geq c_j$$

$\phi_j^k = \text{False}$ otherwise

Let LA_i be the automaton connected to the Each material i and let $s_i^k(t)$ be its current state in the epoch number k .

In regards to these notations the amount $x_i^k(t)$ would be equal to $\frac{s_i^k(t)^\alpha}{N^\alpha}$

Assume that $V_i^k(t)$ and ϕ_j^k , $1 \leq j \leq m$, are the resulting feedback from the stochastic Environment, where k refers to the the epoch and t states for the iteration number within this epoch. Then the state of the automaton is updated as follows:

We suppose that x_i is stored in the bin of index p ,

$$s_i^k(t+1) := s_i^k(t) + 1 \quad \text{If } V_i^k(t) = 1 \text{ and } 1 \leq s_i^k(t) < N \text{ and not } \phi_p^k(t)$$

$$s_i^k(t+1) := s_i^k(t) - 1 \quad \text{If } V_i^k(t) = 0 \text{ and } 1 < s_i^k(t) \leq N \text{ and } \phi_p^k(t)$$

$$s_i^k(t+1) := s_i^k(t) \quad \text{Otherwise.}$$

b. Application to distributed web crawling:

In this paper, we use the update detection probability as the token of interest in this paper. We denote the update detection probability of a web page i as d_i . Under the above conditions, d_i depends on the frequency, x_i , that the page is polled with, and is modeled using the following expression:

$$d_i(x_i) = 1 - q_i^{\frac{1}{x_i}}$$

Increasing the polling frequency reduces the probability of discovering new information on each polling.

Our problem can be defined as . Given m crawlers with capacities c_1, c_2, \dots, c_m , allocate the available capacity among web pages so that expected number of pollings uncovering new information per time step is maximized.

Given the above considerations, our problem can be formalized as follows:

$$\begin{aligned} & \text{Maximize } \sum_{i=1}^n x_i \times d_i(x_i) \\ & \text{subject to } \sum_{j=1}^m y_{ij} = 1, \quad 1 \leq i \leq n \\ & \quad y_{ij} \in \{0, 1\}, \quad 1 \leq i \leq n, \quad 1 \leq j \leq m \\ & \quad \sum_{i=1}^n y_{ij} x_i \leq c_j, \quad 1 \leq j \leq m \\ & \quad x_i \geq 0 \quad 1 \leq i \leq n \end{aligned}$$

Note that in the general web monitoring case, we are not able to observe $d_i(x_i)$ or q_i directly polling a web page only reveals whether the web page has been updated *at least once* since our last poll.

We want $F_i(x_i)$ to instantiate to the value 0 with probability $1 - d_i(x_i)$ and to the value 1 with probability $d_i(x_i)$. Accordingly, if the web page i is polled and i has been updated since our last polling, then we consider $F_i(x_i)$ to have been instantiated to 1. And, if the web page i is unchanged, we consider $F_i(x_i)$ to have been instantiated to 0.

Application of approach 1 to distribute web crawling:

Given m crawlers with different capacities c_1, c_2, \dots, c_m , we assume a single crawler with capacity $C = \sum_{i=1}^m c_i$. The solution to the single crawler problem will form an upper performance bound for our crawling problem.

At each epoch k , we run the scheme presented in Oommen paper [8,9] with a knapsack of capacity C_k , where $C_k = \eta^k * C_0$, $C_0 = \sum_{i=1}^m c_i$ and $\eta = 0.99$. If the items can be packed in the bins in the end of the epoch k , we say that we have attained a near optimal solution otherwise we start a new epoch $k+1$ with knapsack capacity C_{k+1} .

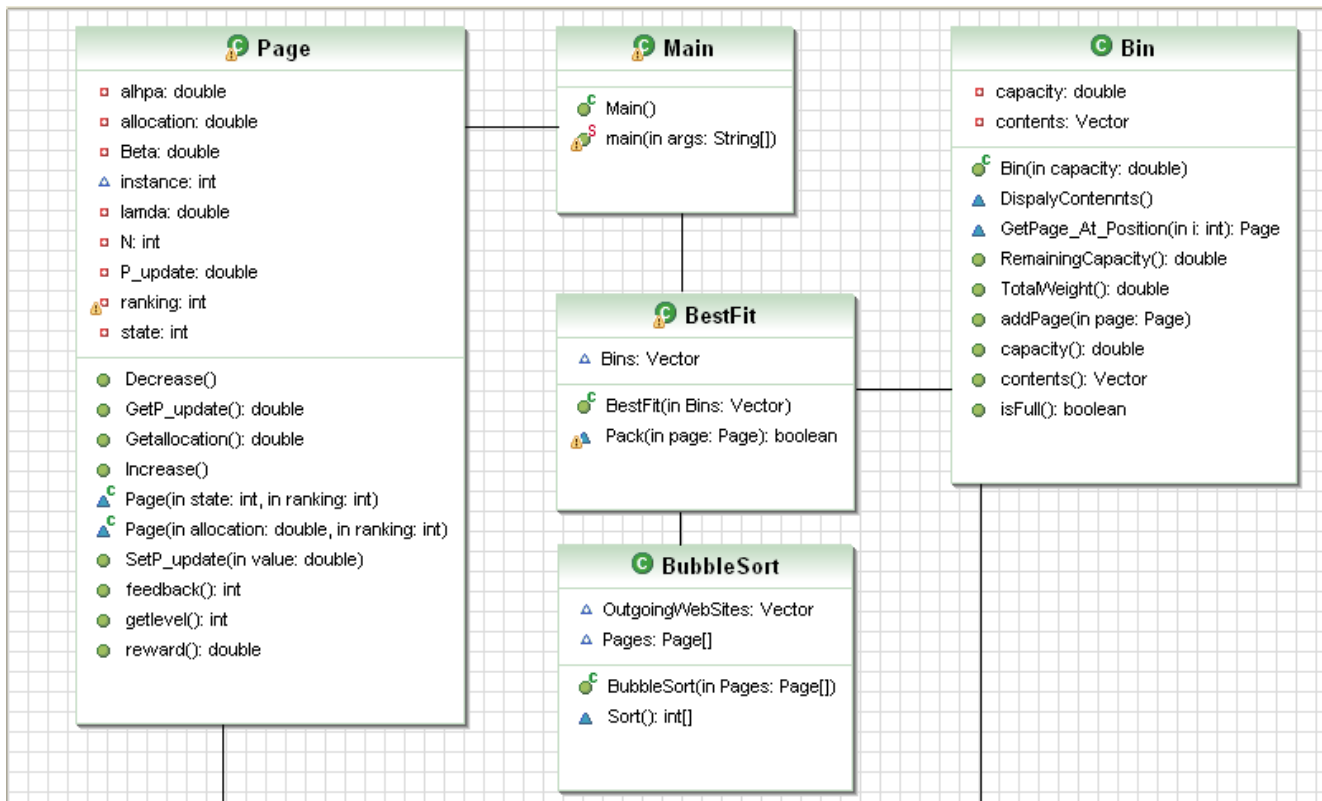
Application of approach 2 to distribute web crawling:

In this approach, we start by spreading randomly the web pages among the m crawlers. Each crawler is regarded as a knapsack and contains a subset of the set of the pages characterized by their polling frequencies $\{x_i, 1 \leq i \leq n\}$. We run Oommen algorithm in every epoch for a large number of iterations. At the end of each epoch we compute for every crawler j the expected value of pollings uncovering new information $\sum_{i=1}^n y_{ij}(x_i \times d_i(x_i))$. In order to improve the global solution we move the web page with the smallest expected value from the crawler having the largest expected value to the crawler having the lowest expected value.

1.3 Implementation

Our proposed solution was implemented using Java. The code includes common functions and classes that present the skeleton of the implementation of the two approaches. For the sake of brevity and clarity, we introduce just the basic common functionalities without getting deeply in the details of each approach.

The figure below illustrates the UML class diagram that describes our program structure.



To start with, the user is prompted to specify the main parameters of the simulation namely the number of crawlers, their capacities, the number of web pages, the distribution of the update probability and the initial values of the polling frequencies. We should note that the distribution of the update probabilities web page is simulated as a Zipf distribution. Therefore the user is invited to enter the parameter β that determines the skewed-ness of the distribution, and the parameter α that represents the magnitude of the update probabilities.

Once all these parameters are defined, our program proceeds by creating objects of the type pages. Clearly, each web page is represented by an object of the type Page. In addition, each crawler is represented by an object of the type Bin. We should notice that in the case of the first approach we are using one single "Bin" (or knapsack) at each epoch which a capacity is that is proportional to the sum of the crawlers capacities.

Every web page is connected with a learning automaton. The two methods Increase () and decrease () implemented in the class Page permit to adjust the polling frequency with regards to the environment responses. The principles of the environment simulations are described below:

p_{update} (Probability of update at a single time step)
 $q=1-p_{update}$ (Probability of remaining *unchanged*)
 x (Polling frequency)
 $d = 1 - q^{\frac{1}{x}}$ (Update detection probability)
 If ($d > \text{Random}(0,1)$)
 then Page is updated
 else Page is unchanged

An interesting part of the implementation is Best fit decreasing algorithm used in the case of the approach 1. As it is known, the best fit decreasing algorithm consists of two phases. First we arrange the items in a decreasing order. To fulfil this requirement we implemented the sorting algorithm

Bubble sort, also known as exchange sort. Then, we perform the best fit algorithm to pack the items into the bins. (Note that Best Fit is distinct from Best Fit Decreasing). Hence, to implement Best Fit Decreasing algorithm we developed the two classes: Best Fit and Bubblesort. The Best Fit Algorithm is explained below:

```

Weight (The weight of the item to pack)
IBF (Index of the best fit bin)
IBF= -1
I=0
Min=Capacity
FOR EACH BIN IN THE VECTOR BINS {
    If (BIN.RemainingCapacity ( ) >= Weight )
        If (BIN.RemainingCapacity ( ) - Weight < Min)
            {
                Min= BIN.RemainingCapacity ( )-Weight ;
                IBF =I;
            }
        I++;
}
If (IBF = -1)
    Then Create a new Bin to pack the item
    Else Pack the Item in the Bin indexed by IBF

```

After the performing the initialization phase as described above, we start iterating. During every time slot, the LA learn the polling frequencies

To compare the throughput of the LA based solution with the optimal solution. The definition of the optimal solution will be defined in the experiments parts. We present here the algorithm to obtain the optimal solution in the case of Knapsack problem:

```

 $x_0$  =Epsilon (Epsilon is a small seeder)
 $q_i$  (Probability of remaining unchanged of page i , )
Optimal=False
While (Not Optimal)
{
     $d_0 = 1 - q_0^{\frac{1}{x_0}}$ 
    sum=0
    for i=1 to n
    {
         $x_i = \frac{\ln(q_i)}{\ln(1-d_0)}$ 
        sum=sum+xi
    }
    if (sum >= capacity)
        then Optimal= True
        else  $x_0 = x_0 + \epsilon$ 
}

```

1.4 Validation and Testing

In this section we evaluate both of our learning schemes by comparing them with three classical policies using synthetic data. We have implemented the following classical policies:

Uniform: The uniform policy allocates monitoring resources uniformly across all web pages. This is the only classical policy of the four that can be applied directly in an unknown environment. Nevertheless, when applied in a distributed allocation problem this definition does not comply directly with the requirements of a distributed allocation problem. Given m crawlers with capacities c_1, c_2, \dots, c_m and n web pages to crawl. The uniform allocation policy can be defined as following :

Let $C = \sum_{i=1}^m c_i$ be the sum of crawlers capacities.

- 1- Allocate to every web page a polling frequency $\rho = \frac{C}{n}$
- 2- Try to pack the web pages in the "bins " using best fit decreasing algorithm
- 3- If the web pages can not be packed then Go to 1 with $C = C \times \eta$, where η is typically equal to 0.99

Proportional: In the proportional policy, the allocation of monitoring resources to web pages is proportional to the update frequencies of the web pages. Accordingly, this policy requires that the web page update frequencies are known. Like the definition of the uniform policy, this definition should be reviewed when applied in a distributed context.

Let $C = \sum_{i=1}^m c_i$ be the sum of crawlers capacities.

Let P_i be the update probability of page number i .

The proportional allocation policy can be defined as following :

- 1- Allocate to every web page number i a polling frequency $\rho_i = \frac{P_i}{\sum_{k=1}^n P_k} \times C$
- 2- Try to pack the web pages in the "bins " using best fit decreasing algorithm
- 3- If the web pages can not be packed then Go to 1 with $C = C \times \eta$, where η is typically equal to 0.99

Optimal :

Obtaining the optimal solution is an NP hard. In fact it requires exploring all possible configurations to pack n items into n bins. For every configuration, we should find an optimal solution for m (m is the number of crawlers) separate knapsack problems based on the principle of Lagrange multipliers for a capacity .The optimal policy requires that web page update frequencies are known, and finds the optimal solution based on the principle of Lagrange multipliers for a capacity .To evaluate web resource allocation policies, recent research advocates Zipf-like distributions to generate realistic web page update frequencies [8, 9, 18, 19]. The Zipf distribution can be stated as follows :

$$Z(k; s, N) = \frac{1/k^s}{\sum_{n=1}^N 1/n^s}$$

where N is the number of elements, k is their rank, and s is a parameter that governs the skewedness of the distribution (e.g., for $s = 0$ the distribution is uniform).

For our experiments, web pages are considered ranked according to their update frequencies, and the update probability of a web page is calculated from its rank. We use the following function to determine the update probability of each web page:

$$q_k(\alpha, \beta) = \frac{\alpha}{k^\beta}$$

In this case, k refers to the web page of rank k and the parameter β determines the skewed-ness of the distribution, while $\alpha \in [0.0, 1.0]$ represents the magnitude of the update probabilities (i.e., the web page of rank 1 is updated with probability α each time step).

Due to the fact that the non linear bin packing problem is NP hard and therefore determining the optimal polling frequencies is not feasible even if the update probabilities of the web pages are given, we define a near optimal policy that is based on solving the bin packing problem as a knapsack problem using the principles of Lagrange multipliers.

Let $C = \sum_{i=1}^m c_i$ be the sum of crawlers capacities.

The optimal policy is defined as following

- 1-Determine the optimal polling frequencies for a non linear Knapsack problem with a capacity C
- 2- Try to pack the web pages in the "bins" using best fit decreasing algorithm
- 3-If the web pages can not be packed then Go to 1 with $C = C \times \eta$, where η is typically equal to 0.99

Approach 1:

The results of our experiments in the case of the first approach are truly conclusive and confirm the power of our LA based solution. We report for the sake of brevity only the results for 512 web pages within the following environments:

- $\alpha = 0.3; \beta = 1.5$
- $\alpha = 0.3; \beta = 1.0$
- $\alpha = 0.9; \beta = 1.5$. Because of the high values of both α and β , this environment turns out to be the most challenging one, discriminating clearly between the optimal policy and the proportional policy.

For the experiments, we used 10 Bins with the following capacities $c_1=0.8$, $c_2=0.2$, $c_3=0.7$, $c_4=0.3$, $c_5=0.6$, $c_6=0.4$, $c_7=0.8$, $c_8=0.2$, $c_9=0.7$, $c_{10}=0.3$

Total polling capacity capacity 5. All values of $\sum x_i \times d(x_i)$ are normalized to 1 by division by the total polling capacity.
Each epoch is composed of 150 000 iterations.

The following table resumes the experimental results. (Note that LA stands of the learning automata based solution)

(α, β)	Scheme	$\sum x_i \times d(x_i)$
(0.9, 1.5)	LA	0.9386
	Uniform	0.23
	Proportional	0.9370
	Optimal	0.943
(0.3, 1.5)	LA	0.578
	Uniform	0.122
	Proportional	0.561
	Optimal	0.598
(0.3, 1)	LA	0.826
	Uniform	0.452
	Proportional	0.824
	Optimal	0.837

- $\alpha = 0.9; \beta = 1.5$ The LA off-line solution took 2 epochs to converge.
- $\alpha = 0.3; \beta = 1.5$ The LA off-line solution took 46 epochs to converge.
- $\alpha = 0.3; \beta = 1$ The LA off-line solution took 7 epochs to converge.

The results demonstrate the power of our approach. In fact, it yields better results than the uniform and proportional policies.

Approach 2

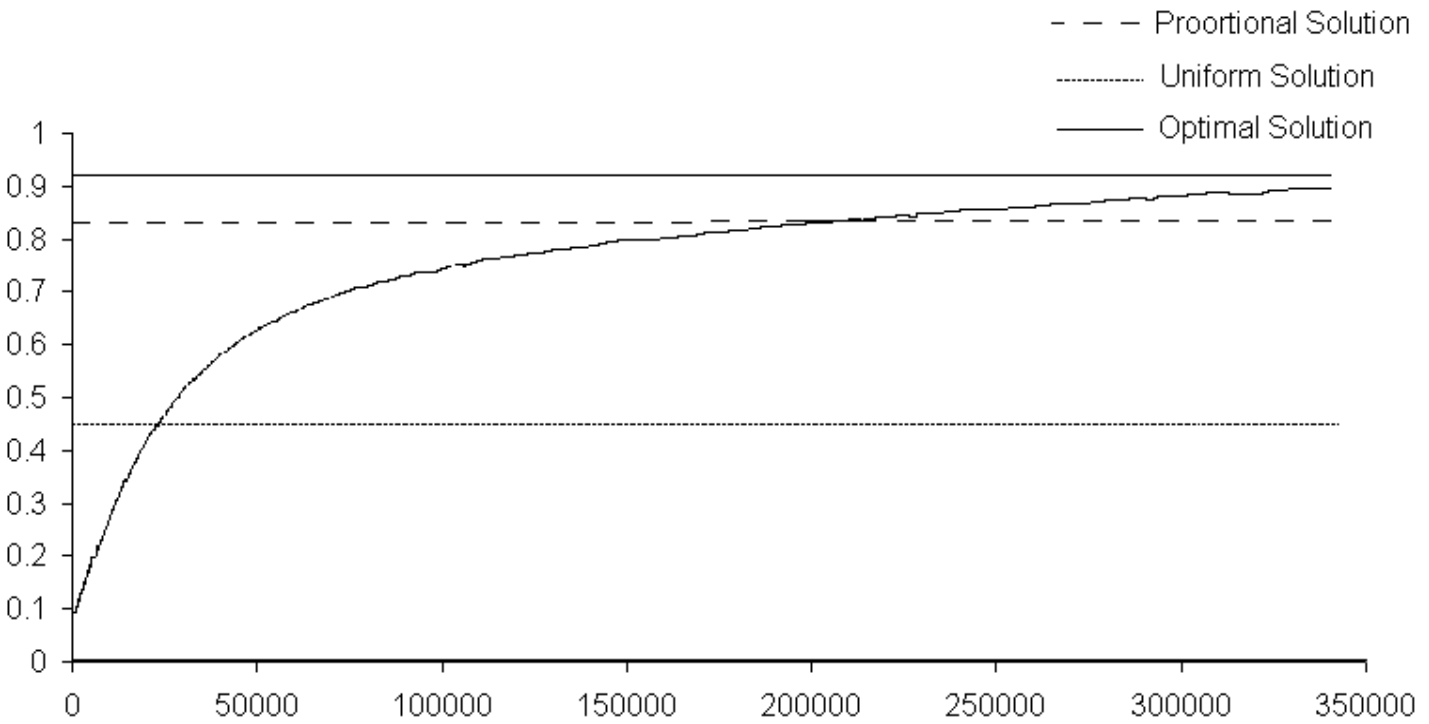
For the experiments, we used 10 Bins with the following capacities $c_1=0.8$, $c_2=0.2$, $c_3=0.7$, $c_4=0.3$, $c_5=0.6$, $c_6=0.4$, $c_7=0.8$, $c_8=0.2$, $c_9=0.7$, $c_{10}=0.3$.

In addition, we used 512 pages. We chose $\alpha = 0.3$ and $\beta = 1$ as parameters of the Zipf distribution.

The total polling capacity capacity 5 . All values of $\sum x_i \times d(x_i)$ are normalizes to 1 by division by the total polling capacity.

Each epoch is composed of 50 iterations. Therefore, every 50 iterations, we move the web page with the smallest expected value from the crawler having the largest expected value to the crawler having the lowest expected value.

The below figure describes the performance of the second approach compared to the optimal, uniform and proportional policies.



The results are really conclusive. In fact the learning automata based solution approaches incrementally the optimal policy and performs better than the proportional and uniform policies after a sufficient number of iterations.

Discussion

In this project, established the key findings of a novel bin packing problem, namely the non linear bin packing problem. We made use of fixed structure automaton to determine the optimal polling frequencies of web pages monitored by a distributed crawler. The conducted results seem to be particularly promising. In fact, both of the presented approaches perform better than the best known policies, namely the proportional policy. In this perspective, we defined in the section experiments the uniform, the proportional and the optimal policies when dealing with a distributed context.

We have proposed two approaches for solving this problem. The first approach is offline whereas the second runs online.

Despite the accuracy and efficiency of the first approach, it can not be put into practice in a real web environment. Nevertheless, this approach is appealing because it learns the optimal polling frequencies when the web pages update probabilities are unknown. The main disadvantage of this approach is that it demands a lot of computational resources and time since it runs usually for several epochs composed of large number of iterations.

The second approach is an online approach that can be applied easily in a real web environment. The tests have shown an efficient use of the bins capacity. Most remarkably the items are spread to the bins near optimal way.

Due to the limited time in this project we did not test this approach in a dynamic environment and we contented of reporting the tests done in a static environment. However, we believe that the proposed solution is able to cope with a dynamic environment.

5 Conclusion

In this paper, we introduced a novel variant the bin packing problem problem namely, the nonlinear bin packing problem. Furthermore, we proposed two learning automata based approaches for solving this problem.

We investigated the optimal solution to the problem of resources allocation of a distributed crawler when the polling capacity is restricted by mapping it the the non linear fractional bin packing problem. Comprehensive experimental results have shown the superiority of the two schemes . As future work, we propose to test the second scheme in a real web environment and to prove analytically its convergence and not just empirically. We also intend to find application domains for our algorithm. In addition, we aim to develop alternate LA-based solutions for different classes of bin packing problems.

References

- [1] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In Proc. WWW conf., April 1998.
- [2] A. Heydon and M. Najork. Mercator: A scalable, extensible web crawler. In Proc. WWW, 2(4):219-229, December 1999.
- [3] S. Lawrence and C. L. Giles. Searching the World Wide Web. *Science*, 280(5360):98-100, April 1998.
- [4] J. Cho, H. Garcia-Molina. The evolution of the web and implications for an incremental crawler. In Proc. Of 26th Int. Conf. On VLDB, September 2000.
- [5] Cho, J., Garcia-Molina, H. Estimating frequency of change. *ACM Transactions on Internet Technology*, 3(3): 256-290, 2003.
- [6] B. E. Brewington and G. Cybenko, "Keeping up with the changing Web," *Computer*, vol. 33, no. 5, pp. 52-58, 2000.
- [7] J. Cho and A. Ntoulas. Effective change detection using sampling. In Proc. 28th VLDB Conf., Hong Kong, China, 2002.
- [8] Ole-Christoffer Granmo, B. John Oommen, Svein-Arild Myrer and Morten 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*.
- [9] Ole-Christoffer Granmo, B. John Oommen, Svein-Arild Myrer and Morten G. Olsen. Determining Optimal Polling Frequency Using a Learning Automata-based Solution to the Fractional Knapsack Problem. *IEEE International Conferences on Cybernetics & Intelligent Systems (CIS) and Robotics, Automation & Mechatronics (RAM)*, Bangkok, Thailand, June 2006.
- [10] Cho, J. and Garcia-Molina, H. (2002). Parallel crawlers. In Proceedings of the eleventh international conference on World Wide Web, pages 124-135, Honolulu, Hawaii, USA. ACM Press.
- [11] Boldi, P., Codenotti, B., Santini, M., and Vigna, S. (2004a). UbiCrawler: a scalable fully distributed Web crawler. *Software, Practice and Experience*, 34(8):711-726.
- [12] D. Zeinalipour-Yazti, M. Dikaiakos, "Design and Implementation of a Distributed Crawler and Filtering Processor," *The Fifth Workshop on Next Generation Information Technologies and Systems (NGITS'2002)*, Caesarea, Israel, June 25-27, 2002
- [13] K. S. Narendra and M. A. L. Thathachar, "Learning automata: A survey," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-14, pp. 323-334, 1974.
- [14] Martello S and Toth P (1990). *Knapsack Problems*. Wiley: New York.
- [15] M. A. L. Thathachar and P. S. Sastry, "Varieties of learning automata: an overview." *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 32, no. 6, pp. 711-722, 2002.
- [16] Morten Goodwin Olsen "Adapting to feasible polling rate in an incremental web crawler using learning automata" *Proceedings from the Workshop on Web Accessibility and Met modelling* April 14-16, 2005
- [17] D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener. A large-scale study of the evolution of web pages. In Proceedings of the Twelfth WWW Conference, Budapest, Hungary, 2003.
- [18] 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.
- [19] 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.
- [20] http://en.wikipedia.org/wiki/Bin_packing_problem
- [21] Coffman Jr., E.G., Garey, M.R., and Johnson, D.S., "Approximation Algorithms for Bin-Packing -- An Updated Survey," *Algorithm Design for Computer System Design*, G. AusieUo, M. Lucertini, and P. Serafini Ed., Springer-Verlag, 1984, pp. 49-106.
- [22] Stadje, Wolfgang Bin-packing problems for a renewal process. *Annales de l'institut Henri Poincaré (B) Probabilités et Statistiques*, 26 no. 1 (1990), p. 207-217
- [23] Hoffman U. A class of simple stochastic online bin packing algorithms. *Computing* 1982;29:227-39

- [24] K. S. Narendra and M. A. L. Thathachar, Learning automata: an introduction. Prentice-Hall, Inc., 1989.
- [25] S.P. Fekete and J. Schepers, "New classes of fast lower bounds for bin packing problems", *Mathematical Programming* 91 (2001) 1, 11–31.
- [26] N. Karmarkar and R.M. Karp, "An efficient approximation scheme for the one dimensional bin packing problem", in *Proceedings of the 23rd. Symposium on Foundations of Computer Science*, 312–320, IEEE Computer Society, 1982.