

Solving the Bin packing problem

Student : **Anis Yazidi**

Supervisor: **Mr Morten Goodwin Olsen**

Presentation outline

Introduction

Web crawling and Learning automaton

The Bin packing Problem

Proposed scheme

Validation and tests

Introduction

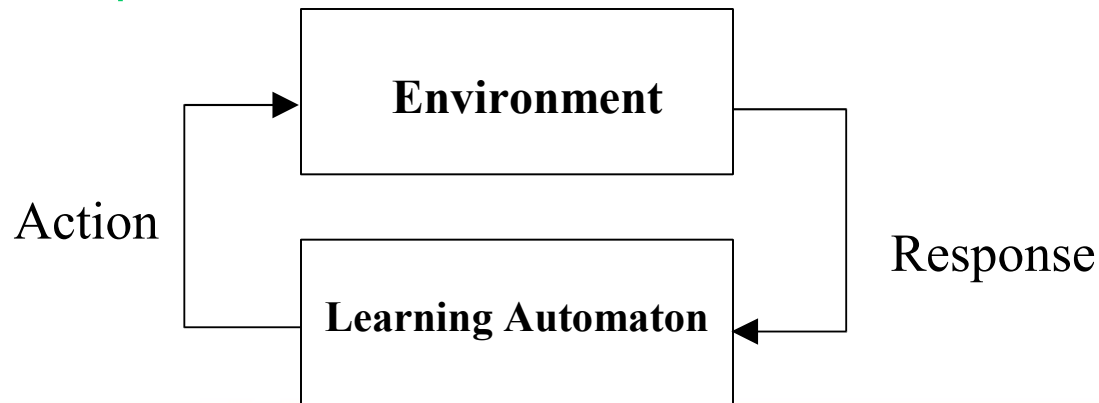
- **Need :** Crawlers download and accumulate web pages in order to provide quick responses to users' queries.
- **Problem :** The web is too big and changes too fast
 - ➔ As the size of the web grows , it becomes imperative to use distributed crawler
 - ➔ Detect as many changes as we can using the fixed download capacity that we have.
 - ➔ Optimize the number of used crawlers

Web crawling

- Batch crawler : shown to be sub-optimal
- Incremental crawler :
 - ⊕ based on a mathematical model of the update patterns of web pages
 - ⊕ based on **learning automata**

Learning automaton

- Given a finite number of actions that can be performed in a random environment, when a specific action is taken place the environment provides a random response which is either favorable or unfavorable.
- The choice of the action at any stage should be guided by **past actions** and **responses**.



Mapping the problem

The Bin packing mapping

Bin size	Crawler capacity
Item	Web site
Item size	the cumulative sum of the polling frequencies of all web pages of the web site in question
Number of items	Number of web sites to crawl

Learning algorithm

- ⊕ All pages are downloaded with a given **probability**.
- ⊕ The polling frequencies of every monitored web page is connected with a learning automaton which :
 - Increases the polling frequency when an update is detected
 - Decreases the frequency when an assumed update did not occur.

Scheme overview

- ⊕ **Time step 0** : Assign the web sites to the crawlers using BFD .
- ⊕ **Time step > 0** : If a crawler capacity is exceeded
 - ▣ Transfer a web site (s) to other crawler (s)
 - ▣ Reducing polling frequencies

Transfer a web site (s) to other crawler (s)

Hypothes : all the nodes of the distributed crawler are interconnected with a network backbone

- ◆ Exclude some web sites from the overloaded bin
- ◆ the choice of the web sites to exclude should also result in a maximization of use of the bin.
 - Use of a variant of the Knapsack problem : Cumulative Subset Sum problem

Reducing polling frequencies

- ◆ reduce the polling frequencies of some web pages to make all the items fit in the bin
- ◆ Greedy Algorithm :reduce just the polling frequencies of the most penalized pages
- ◆ We should have a history of rewards and penalties

But which alternative to choose?

- Use Learning Automaton to decide.
- The key finding of the automaton
 - ◆ Prioritizing the alternative suggesting to decrease the weights
 - ◆ Adopting the “excluding web sites” alternative when the previous alternative fails.
- Connect a learning automaton to every bin.
- Each automaton increases the probability of choosing to exclude some web sites when an overload occurs and decrease the same probability when an overload did not occur.

Test Environments

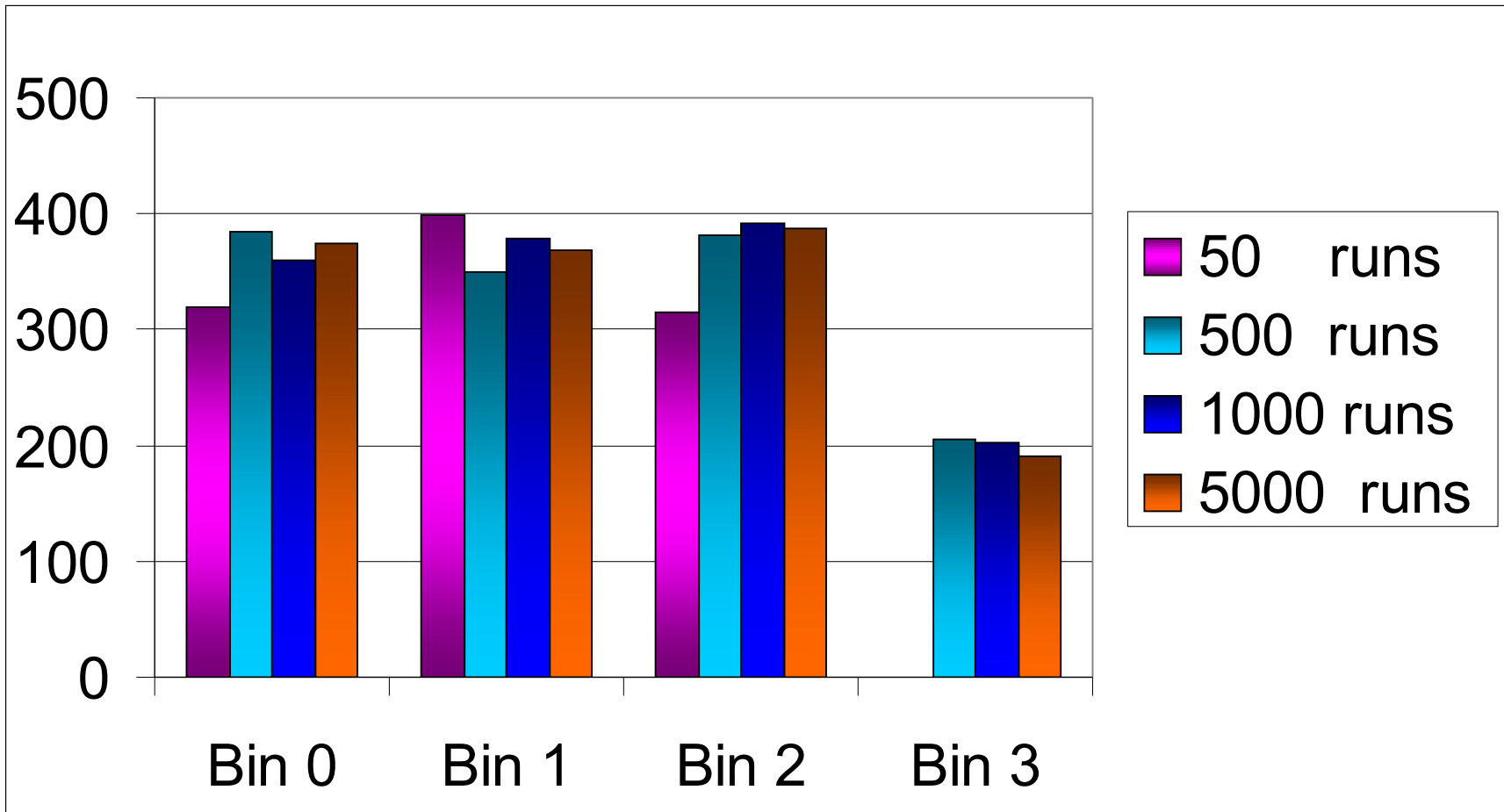
■ Static environment

- Fixed probability according to Zipf function:
- Fixed probability according to top k levels domains

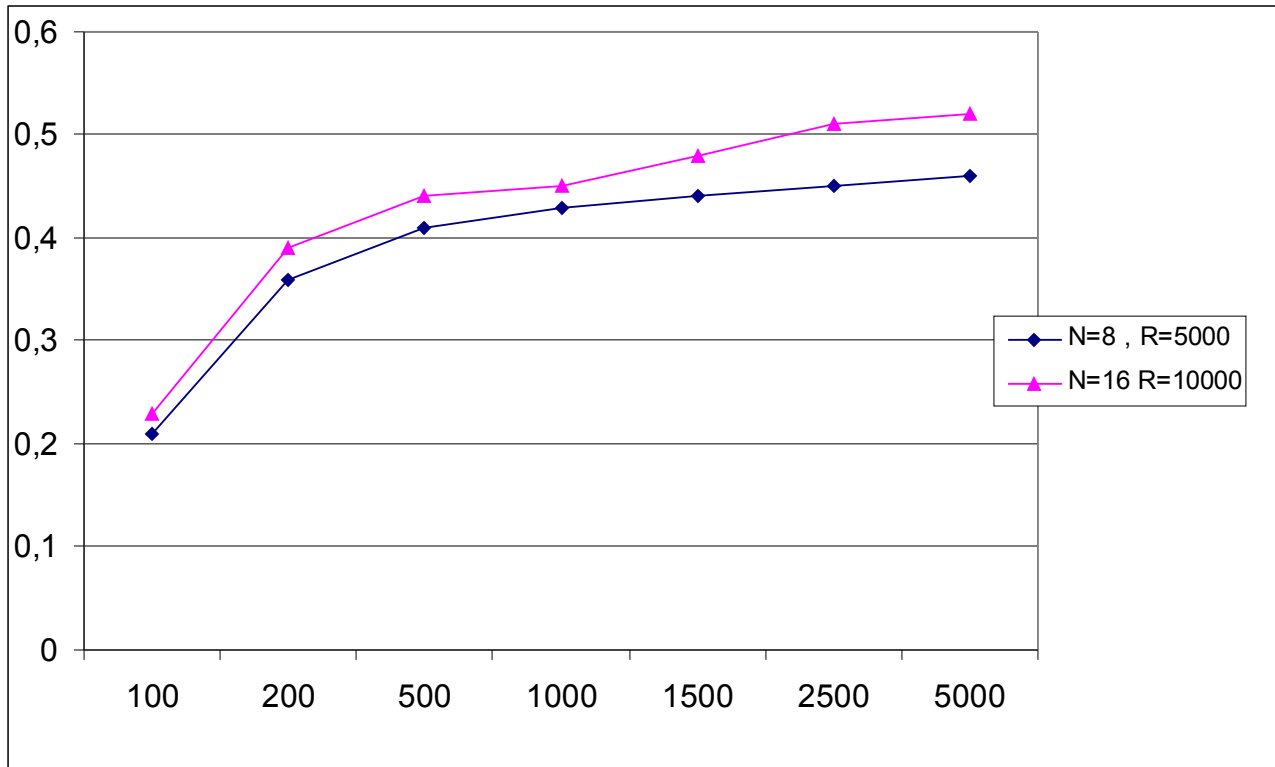
■ Dynamic environment

Switch the Fixed distribution according to top k levels domains

Some Test Results



Some Test Results



Conclusion

- Solution was shown to cope with both dynamic and static environments
- Try to prove analytically the scheme
- Investigate sampling

Thanks for your attention