University of Catania
Department of Mathematics and Computer Science
M.Sc. in Computer Science
Compilers - Academic Year 2014-2015

Web Crawler

Introduction

A crawler (aka web crawler, spider or robot) is an automated software that scan a network (or database) and research a series of URLs into the web pages.
This document regards the description of a Web Crawler implemented in C# language with the IDE **Visual Studio 2010**, which is given as input on or more seeds (URL), an integer value used for search depth, a series of keywords and a computation time.
For each correct and available URL will be sought into the web page the presence of one (or more) keywords.
If the search is successfull, the web page will be saved into the filesystem.
Source code and info are released under GNU GPL v3 licence and are avaiable at the following link https://github.com/R3m3r/WebCrawler

Structure of the project

1 Parsing Phase

The parsing phase divided into 3 phases.
Firstly, there is a download of the page, if the URL given as an input is a valid URL and the extension (if present) of the page is supported (list below).
Then, the second phase will look for keywords into the page, and if one of this is found the page will be saved into the filesystem. The name used will be the full URL which the characters ***"/"*** and ***"."*** are replaced by ***"_"*** (so for instance ***"www.google.it"*** becomes ***"www_google_it.html"***).
Every page also will be stacked into a database.
In the end, the third step will be responsible for controlling the depth entered by the user and compare it with page currently parsing. If the depth is less than or equal, are searched URLs on the page and will be stacked into the database to be subsequently checked.

2 Supported extensions

Supported file extensions (and then accepted by the **Parser**) are :

- html
- htm
- xhtml
- xml
- php
- txt
- asp
- aspx
- jsp
- jspx
- do

3 LinkItem

The LinkItem class is specially created to contain some strings : **Protocol** (http or https), **Domain** (ex. www.google.com), **Local_Path** (ex. /index.html), **Depth** and **Parent Url**.
In addition, the method Href will return the complete URL.

4 Algorithm for finding the diameter

The diameter, by definition, is the length of the longest of the shortest path between any two vertices. That means that we can compute the shortest path for any pair of vertices (**Floyd-Warshall algorithm**), and take the longest one of them.

Floyd-Warshall algorithm

```
let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
for each vertex v
    dist[v][v] ← 0
for each edge (u,v)
    dist[u][v] ← w(u,v)  // the weight of the edge (u,v)
for k from 1 to |V|
    for i from 1 to |V|
        for j from 1 to |V|
            if dist[i][j] > dist[i][k] + dist[k][j]
            dist[i][j] ← dist[i][k] + dist[k][j]
            end if
```

Find the longest of the shorter path

```
let diameter be -1
for each vertex u
    for each vertex v
        if diameter < d[u][v]
        diameter = d[u][v];
        endif
```

Behaviour

Software taken in input:

- one or more initial seeds (urls).
- one or more keywords.
- a maximal depth.
- a computation time (in minutes).

with a command like this:

```
./WebCrawler -hostnames [SEED1 SEED2 .. ] -keywords [KEY1 KEY2 .. ] -depth DEPTH -comptime COMPUTATION_TIME
```

Every discovered URL will be put into a database with the following fields:

- **url** (full path of the url founded)
- **filename** (file name of the file saved if some keywords if founded)
- **parent_url** (url of the parent page)
- **depth** (actual depth of the page)
- **discovered_urls** (number of urls discovered into the web page)

Concrete Example

There have been four simulations from 1 hour, 24 hours, 7 days and 1 month.
The results obtained for each simulation were derived from the following transactions:

1. The output from the execution of the crawler was passed to mail parser that made it possible to extract all the emails valid in the various file.txt, which correspond to the various URL analyzed in which there is at least one of the words keys, and were included in a database.
2. Were then carried out the connections to both the database containing all the emails and is the database created by the crawler spiderino. In this way, via the query area, they have been made statistics for the maximum depth (the URLs found and URLs analyzed), the number of URLs found and analyzed for each domain of the 1st level, and the number of emails found for each URL analyzed.

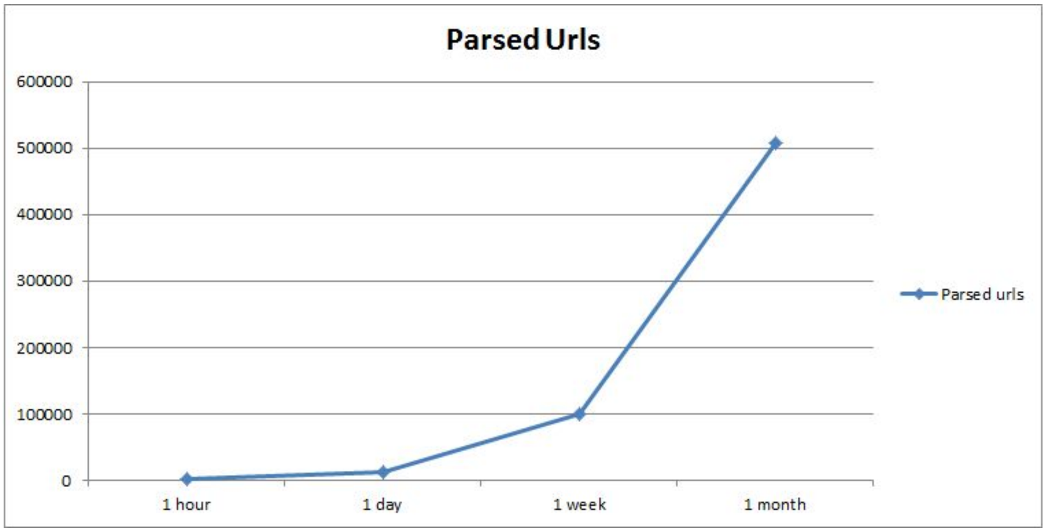
1 Seeds used

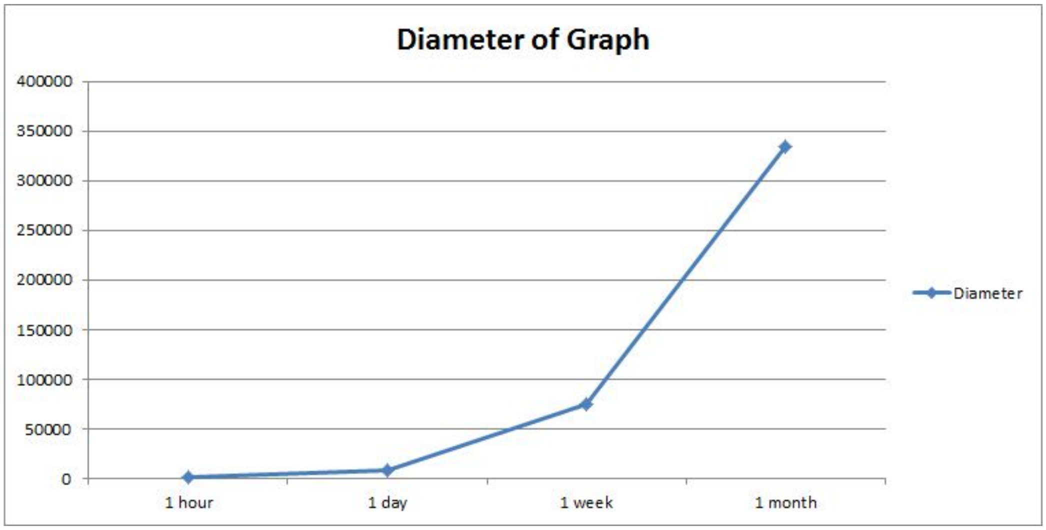
- http://www.sysbio.se/people.html
- www.synbioproject.org/
- www.systemsbiology.org/
- www.biomedcentral.com/bmcsystbiol
- syntheticbiology.org
- yeastgenome.org
- smb.org
- wiki.yeastgenome.org
- www.synberc.org/
- https://synbio.mit.edu/
- www3.imperial.ac.uk/syntheticbiology
- synbioconference.org/
- www.globalengage.co.uk/synthetic-biology.html
- pubs.acs.org/synthbio
- http://www.syntheticmicrobe.bio.lmu.de/members/phd-students/index.html

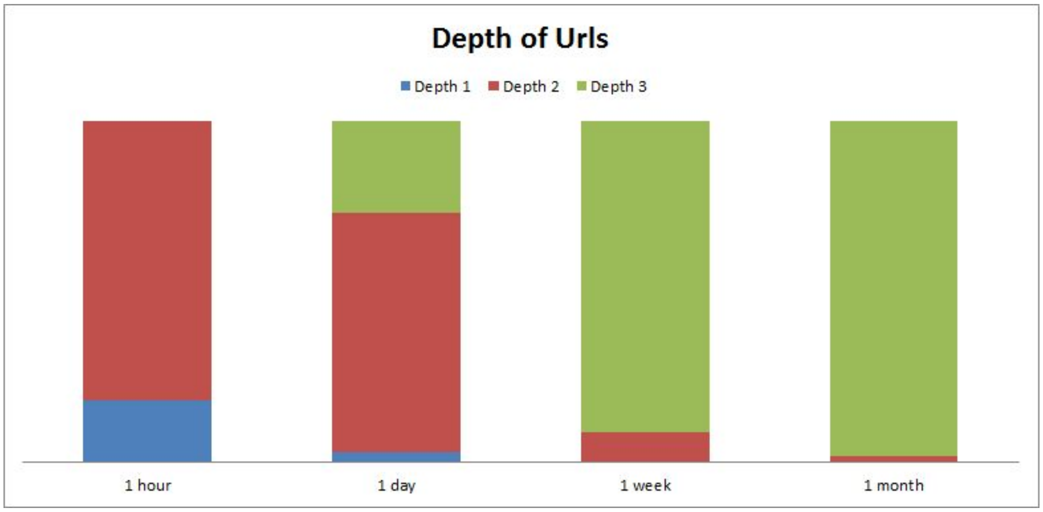
2 Query used

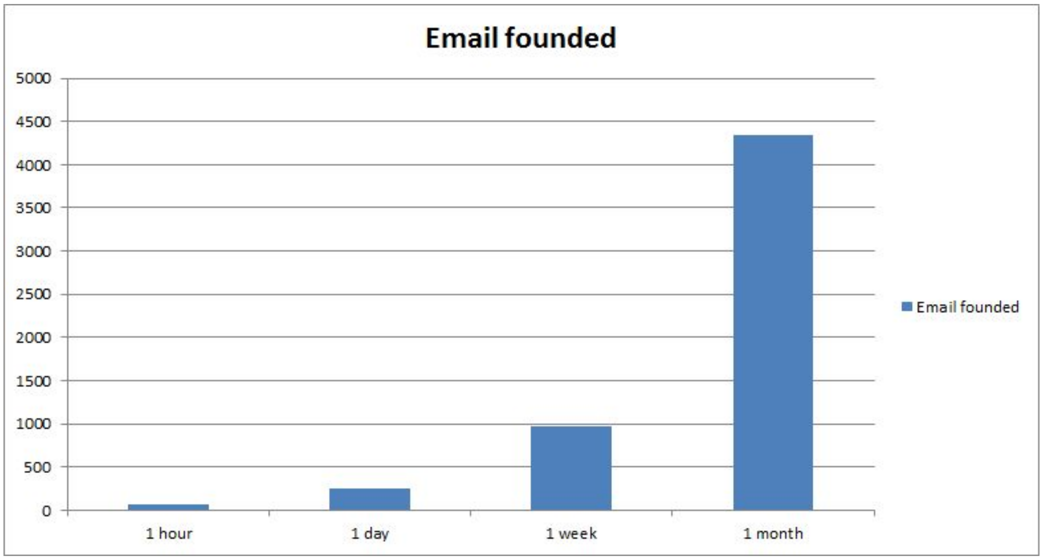
1. "phd student" && ("systems biology" || "synthetic biology" || "machine learning" || "computational biology" || "bioinformatics")
2. "student" && ("systems biology" || "synthetic biology" || "machine learning" || "computational biology" || "bioinformatics")
3. ("systems biology" || "synthetic biology" || "machine learning" || "computational biology" || "bioinformatics")
4. "post doc" && ("systems biology" || "synthetic biology" || "machine learning" || "computational biology" || "bioinformatics")

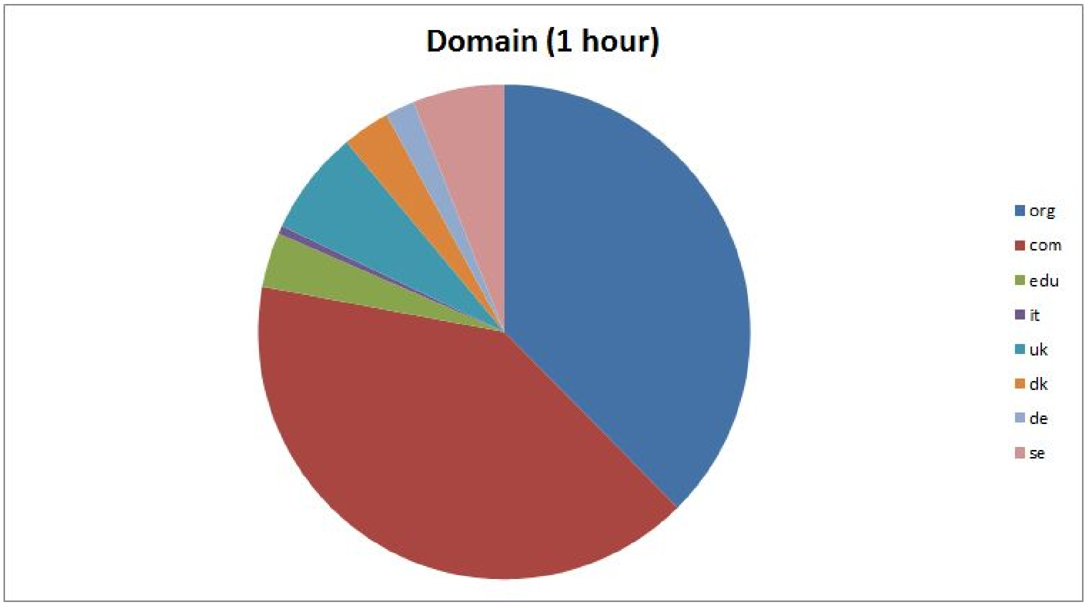
3 Results

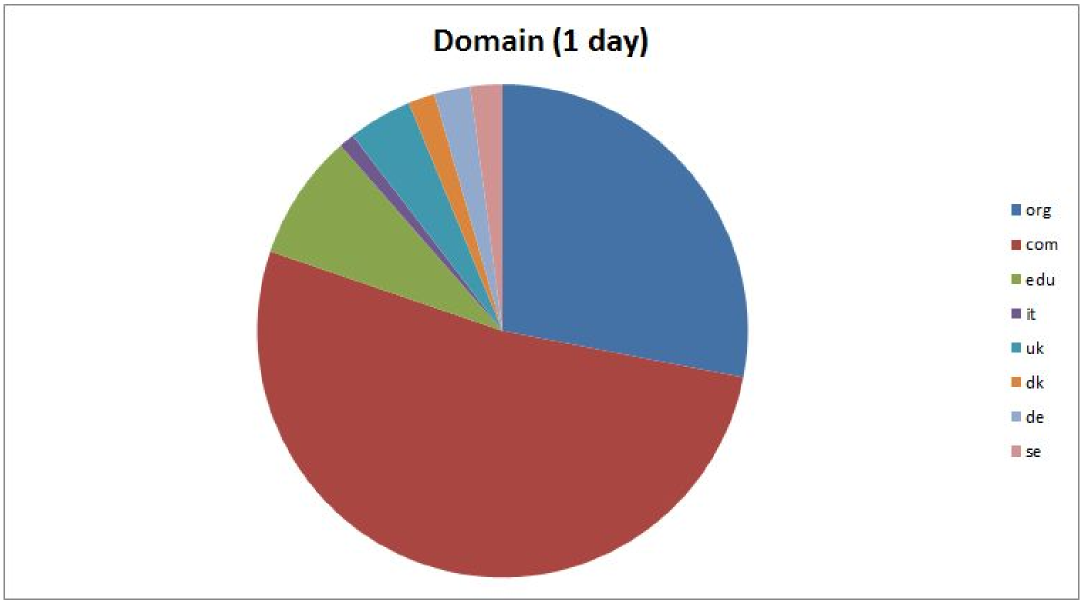

Parsed Urls
600000
500000
400000
300000
200000
100000
0
1 hour
1 day
1 week
1 month
Parsed urls

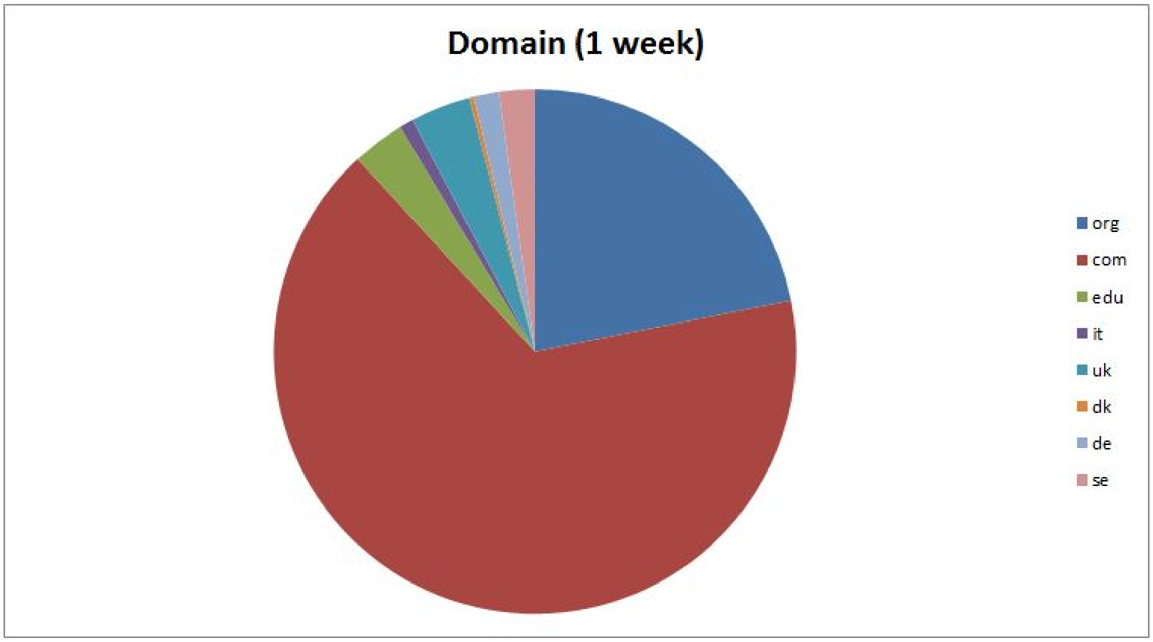

Diameter of Graph
400000
350000
300000
250000
200000
150000
100000
50000
0
1 hour
1 day
1 week
1 month
Diameter

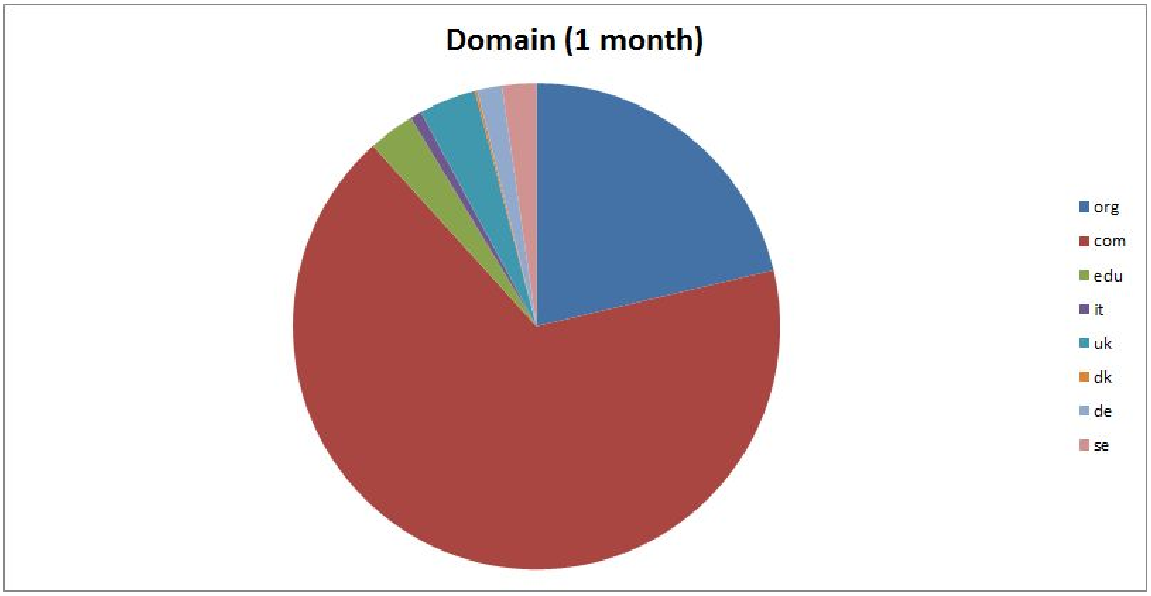

Depth of Urls
Depth 1
Depth 2
Depth 3
1 hour
1 day
1 week
1 month


Email founded
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
1 hour
1 day
1 week
1 month
Email founded


Domain (1 hour)
org
com
edu
it
uk
dk
de
se


Domain (1 day)
org
com
edu
it
uk
dk
de
se





The number of the **connected component** is always 15, probably because there aren't common link between the subgraphs.