Website Identification
DEA Internship Report

Pierre P. Senellart

Advisors: Serge Abiteboul and Grégory Cobéna

September 8, 2003

Abstract

I present in this paper a method to discover the set of webpages contained in a logical website, based on the link structure of the Web graph. Such a method is useful to identify the boundaries of what to crawl, in the context of Web archiving. For this purpose, I combine the use of an online version of the preflow-push algorithm, an algorithm for the maximum flow problem in traffic networks, and of the Markov CLuster (MCL) algorithm. The latter is used on a crawled portion of the Web graph in order to build a seed of initial webpages, a seed which is extended by the former. Experiments on subsites of the INRIA Website, which give satisfactory results, are described.

1 Introduction

Although the notion of *website* is commonly understood, there is no simple formal definition of it. The most obvious idea would be to define a *logical website* as the set of webpages hosted by a given webserver. Although this is correct for many websites, this does not reflect the intuitive notion of website:

- Some websites span over several webservers (e.g. most static pages of the INRIA Rocquencourt Website are on `www-rocq.inria.fr` whereas dynamic pages are on `osage.inria.fr`).
- Some webservers host different websites (e.g. `www.geocities.com`).
- The notion of webserver is also unclear:
  - in the context of distributed (load-balanced) webservers or mirrors: different physical machines may host the same website.
  - in the context of virtual web hosting: the same physical machine may host different webservers.

The idea to group webpages by domain name, which solves the problem of `www-rocq.inria.fr` and `osage.inria.fr`, is not really better since completely unrelated websites may be hosted on servers with the same domain name: for instance, each *login*`.free.fr` is a virtual host for websites of people or associations which have nothing in common, except being customer of Free Telecom.

There are other issues which show that the problem of website identification is not obvious. On the one hand, some websites may be seen as a whole or as a collection of different websites (e.g. the INRIA website regroups — between other — the websites of each research team).

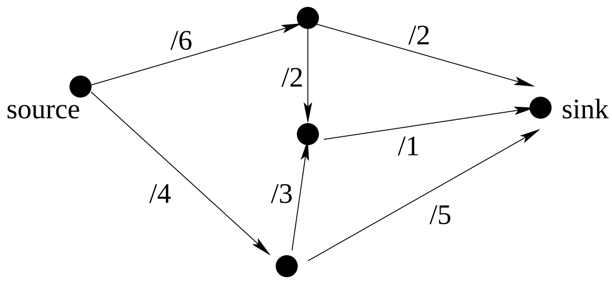

Figure 1: Sample traffic network.

This raises the question of the scale at which websites are to be found. On the other hand, the boundaries of a website are often fuzzy and subjective. It is for instance unclear what part of the websites of its members the website of a research team should contain. That implies that no automatic method for identifying logical websites will be entirely conform to everyone's notion of website. I can only aim at an automatic method which would correspond as much to one person's definition of website as one other person's would.

My interest in discovering the boundaries between logical websites comes mainly from the problem of Web archiving. The French National Library is considering the archiving of the French Web, which consists of the identification, the continuous crawling and the storage of relevant and interesting websites. Issues resulting of this approach are described in [ACMS02]. It is in particular important to be able to discover the boundaries of a website: it makes possible to archive entire websites once webpages from them are selected.

The method for website identification I present in this paper heavily relies on the (directed) graph structure of the Web, with webpages as nodes and hyperlinks as edges. The fundamental assumption is that webpages in the same website are much more connected between them than webpages from different websites. I use an adaptation and combination of two algorithms related to flow simulation in traffic networks: a *preflow-push* algorithm by Goldberg [GT88] which solves the maximum flow problem and the *Markov CLuster algorithm* (abbreviated as *MCL*) by van Dongen [vD00], a graph clustering algorithm. MCL is used to cluster a part of the Web in order to build *seeds* of websites, seeds which are extended to complete logical websites with the preflow-push algorithm. The techniques used are not based on the concept of webservers, domain names or other heuristics, like traditional website recognition methods, but on the link structure of the Web graph and secondarily in the global form of the URLs. This addresses the problems of traditional techniques described above.

I will show in Section 2 how flow simulation and the maximum flow problem may be used to identify websites in the Web graph. We will notice that in many cases, the seed of webpages the simulation starts from needs to be extended; I present in Section 3 a way to use MCL for that purpose. Experiments of my technique will be presented in Section 4. Finally, I discuss related works in Section 5. My contributions are mainly described in Sections 2.4, 3.2 and 4.

2 Flow simulation

In this section, after some necessary basics about traffic networks and the maximum flow /minimum cut problem, I will present the preflow-push algorithm and its online adaptation to the Web. This algorithm will be the base of our website identification process, used in order to extend a seed of webpages to a complete logical website.

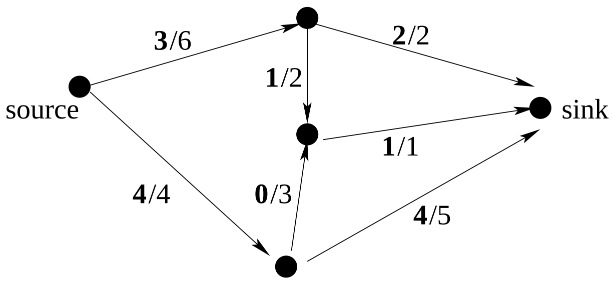

Figure 2: Maximum flow.

2.1 Maximum flow

A *traffic network* is a 4-tuple $\mathcal{T} = (S, c, s, t)$ where S is a set of nodes, $c : S^2 \to \mathbb{R}_+$ is a *capacity* function, $s \in S$ is the source and $t \in S$ is the sink. I suppose furthermore that $c(s, s) = 0$ and $c(t, t) = 0$. The underlying directed graph is (S, A) where $A = \{(u, v) \in S^2 \mid c(u, v) > 0\}$. A representation of a sample traffic network is shown on Figure 1.

A flow in $\mathcal{T}$ is a function $f : S^2 \to \mathbb{R}$, satisfying the following properties:

(i) (*Symmetry*) $\forall (u, v) \in S^2, f(u, v) = -f(v, u)$: the flow going from u to v is the opposite of the flow from v to u.

(ii) (*Capacity constraint*) $\forall (u, v) \in S^2, f(u, v) \leq c(u, v)$: the flow from u to v cannot exceed the capacity of the edge.

(iii) (*Flow conservation*) $\forall u \in S \setminus \{s, t\}, \sum_{v \in S} f(u, v) = 0$: the flow arriving to a node distinct from the source or the sink is equal to the flow departing from this node.

The *flow value* of f is defined as the sum of flows departing from the source or, equivalently, as the sum of flows arriving to the sink:

$$|f| = \sum_{u \in S} f(s, u) = \sum_{u \in S} f(u, t)$$

(The equality between these two sums is a consequence of flow conservation and symmetry).

The *maximum flow* problem is the problem of finding a flow function f such that $|f|$ is maximal. A solution of the maximum flow problem for the traffic network of Figure 1 is shown on Figure 2.

2.2 Minimum cut

A cut of $\mathcal{T}$ is a partition (S_1, S_2) of S with $s \in S_1$ and $t \in S_2$. The *minimum cut* problem is to find a cut whose *capacity*, defined by:

$$\sum_{u \in S_1} \sum_{v \in S_2} c(u, v)$$

is minimal.

There is an equivalence between the maximum flow problem and the minimum cut problem (cf [CLR90]). The maximum flow value is also the minimum cut capacity. Furthermore, a solution of the maximum flow problem can alternatively be seen as a solution of the minimum cut problem, that is a cut of $\mathcal{T}$ whose capacity is minimal, in the following way:

1. Let f be a maximum flow in $\mathcal{T} = (S, c, s, t)$
2. Let $E_f = \{(u, v) \in S^2 \mid c(u, v) > 0 \text{ and } c(u, v) < f(u, v)\}$

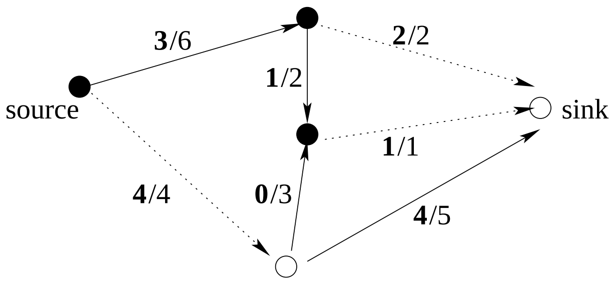

Figure 3: Minimum cut (plain and empty circles show the two parts of the partition).

3. Let $S_1 = \{u \in S \mid \text{there exists a path from } s \text{ to } u \text{ in the graph } (S, E_f)\}$
4. Let $S_2 = S \setminus S_1$
5. (S_1, S_2) is a minimum cut of $\mathcal{T}$

Conversely, if (S_1, S_2) is a minimum cut of $\mathcal{T}$ and $E_f = \{(u, v) \in S_1 \times S_2 \mid c(u, v) > 0$, there exists a maximum flow in $\mathcal{T}$ whose set of *saturated edges* (the edges whose flow equals the capacity) is E_f.

The demonstration of these results can be found in [CLR90] for instance.

Figure 3 represents the minimum cut corresponding to the maximum flow of Figure 2.

2.3 Preflow-push algorithm

The preflow-push algorithm, a solution to the maximum flow / minimal cut problem, is based on the notion of *preflow*, which relaxes the flow conservation constraint. A *preflow* in a traffic network $\mathcal{T} = (S, c, s, t)$ is a function $f : S^2 \to \mathbb{R}$ which satisfies:

(i) (*Symmetry*) $\forall (u, v) \in S^2, f(u, v) = -f(v, u)$

(ii) (*Capacity constraint*) $\forall (u, v) \in S^2, f(u, v) \leq c(u, v)$

(iii) (*Relaxed flow conservation*) $\forall u \in S \setminus \{s, t\}, \sum_{v \in S} f(u, v) \leq 0$

This definition means that a node u can have some *overflow* $o(u) = -\sum_{v \in S} f(u, v)$ in a preflow: it can receive more from the nodes it is pointed by than it sends to the nodes it points to. The preflow-push algorithms, as well as other algorithms working with preflows, maintains at each step a preflow in $\mathcal{T}$, converging finally toward a flow in $\mathcal{T}$ which is maximal.

All nodes are assigned a height (0 in the beginning for all nodes except the source). The preflow is *pushed* from nodes with overflow to lower nodes. If there are no lower nodes to unload a node with overflow, this node is *raised*. The algorithm ends when there are no nodes with overflow any longer. A more formal description of the algorithm is in Figure 4.

This algorithm can be demonstrated to converge toward a maximum flow in $\mathcal{T}$ (cf [GT88]), with a complexity of $O(|S|^2|A|)$ ($|A|$ is the number of edges with non-zero capacity), whatever the strategy for selecting nodes at step 4 of the algorithm may be. Specific strategies can be devised to improve the complexity, up to $O\left(|S||A| \log\left(\frac{|S|^2}{|A|}\right)\right)$.

2.4 Adaptation to the Web

2.4.1 A Web traffic network

I wrote in the introduction that the fundamental assumption of website identification based on the graph structure of the Web is that webpages in the same website are much more connected between them than webpages from different websites. If the Web is seen as a traffic network in which some fluid flows from a set of source nodes in the same website, the bottleneck of the flow

Preflow-push algorithm

1. All nodes are assigned a *height* h:

$$h(s) := |S|$$

$$\forall u \in S \setminus \{s\}, h(u) := 0$$

2. For all $(u, v) \in S^2$, $f(u, v) := 0$.

3. For all $u \in S$, if $c(s, u) > 0$ then $f(s, u) := c(s, u)$ and $f(u, s) := -c(s, u)$.

4. A node $u \in S \setminus \{s, t\}$ with overflow $o(u)$ is selected. If none exists, the algorithm ends.

5. If there exists $v \in S, c(u, v) > f(u, v)$ and $h(v) = h(u)-1$, the overflow of u is *pushed* toward v

$$f(u, v) := f(u, v) + \min(c(u, v) - f(u, v), o(u))$$

   Go to 4.

6. Else, u is raised:

$$h(u) := 1 + \min\{h(v) \mid v \in S \text{ and } c(u, v) > f(u, v)\}$$

   Go to 4.

Figure 4: Preflow-push algorithm.

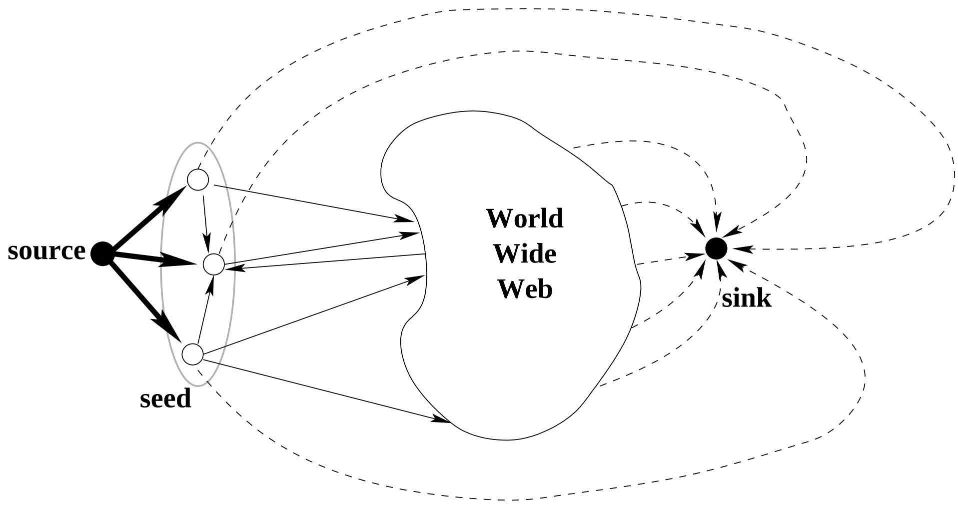

Figure 5: Web traffic network.

should not be inside the website, where there are many internal connections (and thus, a large capacity) but between the website and the rest of the Web, where the connections are much more sparse.

The idea behind using flow simulation to identify websites is that a clear cut should be visible between a “source of seed webpages” of a site and a “sink for the remaining part of the Web”, a cut which would match the borders of the website. This cut will be computed as a minimal cut in a traffic network whose underlying graph is the Web graph. More formally, I assume that I have a set of seed webpages *Seed*, characteristic of the website I would like to compute the borders of, and a similarity function over webpages *sim* and I consider the traffic network $\mathcal{T}_{Seed} = (S, c, s, t)$ where:

- S is the set of webpages in the World Wide Web, along with two virtual nodes s (a virtual source) and t (a virtual sink).
- c is defined as follows:
  (i) For all $(u, v) \in (S \setminus \{s, t\})^2$, $c(u, v) = sim(u, v)$ if there is a link from u to v, $c(u, v) = 0$ otherwise.
  (ii) For all $u \in Seed$, $c(s, u) = +\infty$
  (iii) For all $u \in S \setminus \{s, t\}$, $c(u, t) = \varepsilon$

  (The value of the capacity of an edge from s to a node u in *Seed* needs only be at least as large as the sum of the capacity of other edges starting from u. ε is a small value, much smaller than average similarity values).

The role of the virtual nodes s and t is just to make practical the concepts of “source of seed webpages” and “sink for the remaining part of the Web”. An illustration of this Web traffic network is given in Figure 5.

The choice of the similarity function is important. The most simple choice would be to use a constant function. In this case, however, a cut separating the seed webpages from the rest would be most likely minimal. Its capacity would be in proportion to the total number of links of the seed webpages pointing to other webpages; the number of these links would probably grow if the cut is put farther away. A possible way to fix the problem would be to raise the similarity of webpages near the source. It is an *ad hoc* solution, however, not very satisfying. I could also use complex semantic similarity functions, based on the contents of the webpage. Instead, I chose to use a function of the edition distance between the URLs of the webpages: even if a website span over several webservers, the URLs of the pages tend to look similar

Edition distance and similarity between URLs

1. Let u and v be two URLs.

2. Let $(u_1, \ldots, u_k)$ (resp. $(v_1, \ldots, v_l)$) be the list of maximal substrings of u (resp. v) not containing the chars '`/`', '`:`', '`?`', '`=`', '`&`', '`#`', ordered in the way they appear in u (resp. v).

3. $ed(u, v)$ is the minimum number of modifications, additions and deletions of list elements necessary to transform $(u_1, \ldots, u_k)$ into $(v_1, \ldots, v_l)$ (it can be computed by dynamic programming).

4. $sim(u, v) = e^{\frac{-ed(u,v)^2}{2\sigma^2}}$ where σ is a typical standard deviation for the edition distance over the set of considered URLs (the value of σ was determined experimentally by computing the standard deviation of the edition distance between pairs of crawled URLs; I had $\sigma \approx 7.24$).

Figure 6: Edition distance and similarity between URLs

(e.g. `http://osage.inria.fr/verso/Gemo/PUBLI/index.php` and `http://www-rocq.inria.fr/verso/Gemo/Projects/index.html` are part of the same website). The edition distance between URLs ed and the similarity function sim are then defined as shown in Figure 6.

sim does not need to be exactly the same as defined, of course, but the shape of the Gaussian curve corresponds to what I was looking for: a maximum value in 0, with a slow decrease for lower values, then (after σ) a stronger attenuation toward 0. ε was assigned a matching heuristic value, in my case $e^{\frac{-50^2}{2\sigma^2}}$ as 50 is much larger than typical values of edition distance between URLs.

2.4.2 On-line preflow-push

Off-line and on-line Classical graph and network algorithms are *off-line*: they require that the entire matrix is stored, so that computations can be made on it. In the context of the Web, *on-line* algorithms may be more interesting. An on-line algorithm on the Web graph is an algorithm which does not require the storage of the entire matrix of the graph, and in which computations are made progressively, at the same time webpages are crawled. For instance, Google's PageRank [PBMW98] is normally computed by an off-line algorithm: robots crawl the Web regularly in order to build the Web matrix and once the crawling is done, computation is performed. Abiteboul et al propose in [APC03] an on-line algorithm to compute it. This has several advantages, peculiarly using much less resources and being more reactive to the changes of the Web.

The preflow-push algorithm has two features which makes possible to use an "on-line" version of it, when browsing the Web:

- The strategy for selecting nodes with overflow is not imposed.
- There is no need to have the entire graph stored in memory; in particular, for some node u, there is no need to know the set of nodes pointing to u when accessing u.

Thus, the preflow-push can be made on-line in a straightforward way: webpages with overflow are progressively crawled and dealt with (pushed or raised). I have to maintain the height and overflow of each node, as well as, for all edges where there is flow, the value of the flow through this edge. The latter component can unfortunately grow quite large, potentially in proportion of the number of edges in the graph. At the end of the algorithm, nodes on the same side of the cut as the source are extracted: they form the logical website found by the process.

Crawling strategies Several strategies adapted to crawling can be chosen. I decided to use a greedy one: the node with maximum overflow is selected at each step (a list of candidates nodes ordered by their overflow is maintained). Minor modifications were made, which do not ensure the correctness of the algorithm any longer but which did not have any noticeable impact on it in the experiments I made: the height of the source is decreased (which accelerates the rise of nodes that have to push flow toward the source) and the algorithm may stop before the convergence is obtained (after a timeout or when all nodes overflow are under a threshold value).

Experimental results Applying this version of the preflow-push algorithm on a seed of characteristic webpages of a website gives quite rood results on small or medium-sized, well-organized, websites. When the website is too large or not well organized, however, the algorithm only retrieves a small proportion of the webpages of the real website (cf Section 4.1 and Table 1). The problem lies in the size of the seed, and in the small number of outgoing links in it. Thus, I need a way to extend automatically the seed before carrying out the flow simulation.

3 Extension of the seed

MCL will be the tool needed to extend the seed used by the online push-preflow algorithm. After presenting it, I will describe how it can be used in the context of the Web, and the role it plays in the website identification process I present in this paper.

3.1 MCL (Markov CLuster algorithm)

MCL is a graph clustering algorithm by van Dongen, described in detail in [vD00]. I briefly present it here, as long as several other interesting results found in this reference.

3.1.1 Presentation

Let G be a (possibly weighted) directed graph and M_G its associated matrix (the (i, j) element of M_G is 0 if there is no edge from i to j, the weight of the edge otherwise).

Let $\mathcal{M}_G$ be the matrix obtained from the transpose of M_G by normalizing each column. If d is the diagonal matrix containing the sum of the values of each column, $\mathcal{M}_G$ is defined as:

$$\mathcal{M}_G = M_G^T \cdot d^{-1}$$

$\mathcal{M}_G$ is a column stochastic matrix, that is, each column of $\mathcal{M}_G$ sums to 1. In the context of a random walk on G, $\mathcal{M}_G$ can be seen as the transition matrix of the corresponding Markov chain: the j^{th} column of $\mathcal{M}_G$ contains the probabilities of transition from j to each different node.

The first successive powers $\mathcal{M}_G^2$, $\mathcal{M}_G^3$... (which are also column stochastic) present interesting behaviors: the (i, j) element, which corresponds to the transition probabilities in multiple steps, is all the higher as i and j are in the same dense region. Such feature is very interesting for devising a graph clustering algorithm, but this trend disappears afterwards. Basics of Markov theory say that, if $\mathcal{M}_G$ is irreducible (which happens if G is undirected and connected) and has non-zero values on the diagonal (loops can be added to G for this purpose), the limit matrix $\lim_{k\to+\infty} \mathcal{M}_G^k$ exists and has identical columns (which corresponds, by the way, to the PageRank values [PBMW98]). Thus, at the limit, local heterogeneity of the values of the matrix, corresponding to local density of the graph, disappears. The basic idea of MCL is to amplify the behavior which appears for the first powers of the matrix, by alternating expansion (matrix multiplication) with inflation (rescaling with a power coefficient greater than 1, to increase the heterogeneity of the values).

MCL algorithm

1. Add loops (edges from a node to itself) to every node in G.
2. Build $\mathcal{M} = \mathcal{M}_G$, the Markov matrix associated to G.
3. Do
   (a) $\mathcal{M} := \mathcal{M}^2$
   (b) $\mathcal{M} := \Gamma_r \mathcal{M}$

   While $\mathcal{M}$ is not idempotent under both expansion and inflation
4. Interpret $\mathcal{M}$ as a clustering of G in the following way: the clusters are the maximal subgraphs of G containing a strongly connected component C of G and all nodes x from which there is a path in G to a node in C. These clusters may overlap.

Figure 7: MCL algorithm.

The inflation operator Γ_r on square matrices of dimension n is defined for every $r > 1$ as:

$$\forall (p, q) \in [1..n]^2,\ (\Gamma_r N)_{p,q} = \frac{(N_{p,q})^r}{\sum_{i=1}^{n} (N_{i,q}^r)}$$

r is called the *inflation parameter*. The farther r is to 1, the more heterogeneity is introduced. A typical value is 2.

3.1.2 Algorithm

The MCL algorithm on graph G is then a simple alternation of expansion and inflation (cf Figure 7).

There are some variations to this algorithm: the inflation parameter, as well as the expansion power, may be different (but always greater than 1) at each iteration. In the following, however, the expansion power will always be 2, and the inflation parameter will be a constant, usually 2 when not specified differently.

Note that this algorithm is not guaranteed to converge. There are even examples of non-convergence. However, on most real-life examples (in particular if the graph is undirected), experiments show the convergence. There are two important theoretical results in respect to this:

- The MCL process with standard parameters converges quadratically in the neighborhood of each nonnegative idempotent column stochastic matrix for which every column has one entry equal to 1 and all others equal to 0.
- If $\mathcal{M}_G$ is diagonally similar to a symmetric matrix (this is in particular the case if G is undirected), the iterands of the MCL process with standard parameters are eventually diagonally similar to a positive semi-definite matrix. This property gives the means to associate an overlapping clustering to each iterand, in a way described in [vD00].

Figure 8 gives an example of a graph clustering with MCL, on a simple undirected graph.

Overlaps can happen, which is not really annoying in the case of website identification (some webpage may well belong to different websites), but it is fairly rare and it never happened in the experiments described in section 4.

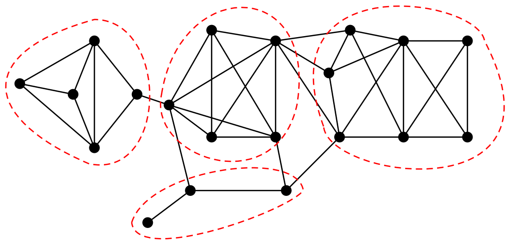

Figure 8: Example of graph clustering with MCL, taken from [vD00].

3.1.3 Scaling

The complexity of the MCL algorithm, as it is presented above, is $O(k|S|^3)$ where $|S|$ is the number of nodes in G and k is the number of iterations. Typically, k is somewhere between 10 or 100 and thus can be considered as a constant. A complexity of $O(|S|^3)$ is of course not adapted to very large graphs, but different optimizations can be made, specially by making the iterands of the process sparse. Simple pruning of each column of the matrix may drastically lower the complexity to $O(|S|l^2)$ if l is the pruning constant (the number of non-zero values kept in each column). The convergence and results of the algorithm are naturally affected by pruning, but experiments show that changes are not very significant and clustering results are still satisfactory.

3.2 Flow simulation from MCL clusters

Ideally, a graph clustering algorithm such as MCL would be applied to the Web graph "as is", thus discovering the different logical websites, each as a different cluster. Clusters could then be clustered furthermore, to extract potential subsites. But MCL is an off-line algorithm. Applying it to the Web would require to have the entire Web graph retrieved and stored, which is impracticable, or at least to have a large portion of it stored, which is feasible but much resource consuming. The process itself would require huge computing power, even if its complexity can be made linear. It takes roughly a day for a $100,000$ nodes graph on a typical personal computer; it would need at least $50,000$ times as much computing power for the entire Web graph. Furthermore, MCL only gives good results (and has sound mathematical foundations) on undirected or almost undirected graphs. It is not at all the case of the Web graph, which is essentially directed. For all these reasons, I will not use MCL to extract the logical websites, but to extend the seed the on-line preflow-push will work from.

The entire process for discovering the borders of a website W is shown on Figure 9.

Tasks 2 and 3 are the most time-expensive ones, but the obtained clustering may be reused for various sites which have a large part in the webpages crawled. For example, if the webpages hosted by a webserver with the `inria.fr` domain name are crawled and clustered, the output can be used for identifying the different sites associated with INRIA.

4 Experiments

In this section, I will describe some experiments on subsites of the INRIA website used to validate our website identification process.

I implemented a multi-threaded crawler (used for flow simulation and for the Web graph extraction) and an on-line version of the preflow-push algorithm in Java 1.4; I used van Dongen's MCL implementation in C, available at `http://micans.org/mcl/`; finally, Perl and shell scripts and XSLT stylesheets were written to format the inputs to the programs and to parse and analyze the results. Experiments were run on a Pentium III 600 MHz PC running Linux, with

Website identification process

1. Identify a superset S of a large part of W. S may be, for instance, the set of webpages hosted by the main webserver for W (or the set of webpages hosted by webservers with the same domain name).

2. Crawl S and build the corresponding subgraph G_S of the Web graph.

3. Cluster G_S using MCL on the underlying undirected graph G'_S (there is an edge (u, v) in G'_S if and only if either (u, v) or (v, u) is in G_S). If some of the resulting clusters are too large, MCL can be applied recursively on them, with different values of the parameters.

4. Find the obtained cluster K which is the most relevant to W. It may be identified as the one which contains a *characteristic* webpage of W (but preferably not the entry point to the website which may be classified in another cluster, given that it is "at the border" of the website). It may also be identified by finding the cluster which contains the largest number of URLs containing some given keyword.

5. Use the on-line preflow-push algorithm with K as a seed. The resulting set of pages is the logical website for W found by the process.

Figure 9: Website identification process.

1 Gb RAM and a broadband access to the Internet. The time bottleneck for flow simulation came from the response times of the webservers (all the more so since I avoided multiple quick successive requests to the same webserver, in order not to overload it).

4.1 The Gemo website

4.1.1 Description of the experiment

My main experiment was on the website of my research team at INRIA, Gemo. Its entry point is `http://www-rocq.inria.fr/verso/` (Verso is the former name for Gemo) and it spans over several webservers, of the `inria.fr` domain and of other domains (for the personal pages of its members who have several affiliations, for instance).

A large part of the webpages hosted on webservers of the `inria.fr` domain was crawled (a breadth-first crawl was performed during 3 days, starting from `http://www.inria.fr`; the last pages to be crawled were at a distance of 7). The resulting subgraph of the Web graph contained 87, 140 nodes and 709, 371 directed edges (that is, on average, 8.14 internal links per webpage). Following the process described in Section 3.2, a MCL clustering was performed on the underlying undirected graph (it took about a day of computation). The most relevant cluster was identified as the one with the largest number of URLs containing "verso". Finally, flow simulation with preflow-push gave the resulting logical Gemo website (in a few hours).

Table 1 shows some data about the website found by the algorithm, along with results of other simpler methods:

- **Flow simulation**: direct on-line preflow-push, starting from `http://www-rocq.inria.fr/verso/`
- **MCL 1.2**, **MCL 2.0**: clusters from MCL, without flow simulation, for two values of the inflation parameter

	Size	Prec.	Recall	THESUS keywords
Flow Simulation	8	87.5%	1.3%	`xml`, `web`, `project`
MCL 1.2	320	99.7%	33.0%	`gemo`, `report`, `server`
MCL 1.2 + Flow Simulation	788	90.4%	86.4%	`gemo`, `report`, `server`
MCL 2.0	249	99.6%	24.9%	`report`, `gemo`, `server`
MCL 2.0 + Flow Simulation	285	94.4%	27.1%	`server`, `report`, `gemo`
`http://www-rocq.inria.fr/verso/*`	221	100%	44.4%	`diapositive`, `texte`, `suivant`
`http://{www-rocq,osage}.inria.fr/verso/*`	683	100%	68.6%	`report`, `diapositive`, `gemo`

Table 1: GEMO website, according to different methods.

- **MCL 1.2 + Flow Simulation**, **MCL 2.0 + Flow Simulation**: process described above, for two values of the inflation parameter
- `http://www-rocq.inria.fr/verso/*`: “naive” recursive crawl of the hierarchy of URLs starting from `http://www-rocq.inria.fr/verso/`
- `http://{www-rocq,osage}.inria.fr/verso/*`: recursive crawl of the hierarchy of URLs starting from `http://www-rocq.inria.fr/verso/` and from `http://osage.inria.fr/verso/`. This method use the human knowledge that `osage.inria.fr` hosts the dynamic pages of the INRIA website; therefore, it is not entirely fair to compare it to automatic methods.

For each method, Table 1 shows:

- the number of pages in the website;
- the *precision* and *recall* of the result W, defined as follows:

$$precision(W) = \frac{\text{number of relevant webpages in } W}{|W|}$$

$$recall(W) = \frac{\text{number of relevant webpages in } W}{\text{total number of relevant webpages}}$$

  The notion of relevant webpages is somewhat subjective, therefore a 100% precision or recall is not a realistic objective. The concepts of precision and recall come from information retrieval, where it is considered that, in most cases, values of 80% or 90% are very good, since they correspond to the relative precisions or recalls between the judgments of two human beings.
- the three most frequent keywords for the set of URLs given by THESUS. THESUS, presented in [HNVV03], is a project for the semantic characterization and clustering of sets of Web documents, which exploits the text around webpages' incoming hyperlinks. An on-line demo can be found at `http://www.db-net.aueb.gr/thesus/services.jsp`. The list of keywords provided by THESUS should describe the set of documents. Agreement between the *semantic* keywords of THESUS and the topic of the website delimited by my *structural* website identification technique would be a kind of cross-validation between the two methods.

4.1.2 Discussion of the results

As noted in Section 2.4.2, flow simulation alone retrieves a very small portion of the relevant webpages, whereas MCL effectively retrieves many more webpages, which are nearly all relevant (that is, the precision is very high). The recall for MCL clusters is still low, however; this is why the online preflow-push is applied afterwards. For an inflation parameter of 1.2, the complete process still gives a good precision, over 90%, and especially gives a high recall, much higher than all the other methods. This shows the interest of the combination of flow simulation and graph clustering techniques, over each technique alone. The naive technique naturally has a perfect precision (since every webpage of the hierarchy `http://www-rocq.inria.fr/verso/*` is of course part of the GEMO website) but a rather low recall: there is indeed a need for more elaborate website identification methods, such as the one I used. Even *ad hoc* techniques like the last one, which uses human knowledge, do not retrieve as much relevant webpages. THESUS keywords also tend to validate the process, since `gemo` appears in the top position (`report` and `server` are also good keywords for the GEMO website, since a large part of is is composed of a publication server which lists GEMO reports).

The results of my experiment on the GEMO cluster are thus very satisfactory. It is to be noted, though, that this does not represent the relative performance of the different techniques on every website. On smaller or more organized ones, the online preflow-push algorithm alone may be sufficient. On many websites, the naive recursive crawl of the hierarchy of URLs may even be perfect. Still, a large part of the Web is composed of not-so-well organized websites, spanning over several webservers, like the GEMO website. For those, the use of the process described here may be very useful.

4.2 Flow simulation from random MCL clusters

4.2.1 Description of the experiment

Another interesting experiment is to look at the clusters found by MCL in order to see if each of them, after flow simulation, corresponds more or less to a logical website. We first applied MCL (with inflation parameter 1.2) to the $87,140$ nodes INRIA subgraph as before. As the main cluster was huge ($49,170$ nodes), I clustered it furthermore, with an inflation parameter of 2.0. I obtained $5,247$ clusters, whose size went from 1 to $8,126$. The distribution is shown on Figure 10 on a log-log scale. It follows more a less a Zipfian law: the probability that a cluster has size s is proportional to $\frac{1}{s^\alpha}$ for some α. There are some outliers to this law, mostly for very big clusters (perhaps that means that those clusters could be clustered furthermore).

5 clusters were then chosen at random among clusters whose size is greater than 50 (too small clusters often do not have much interest). The preflow-push algorithm was applied to each of them, resulting in 5 sets of URLs described in table 2 by:

- their size;
- the most common URL prefixes;
- the three most frequent THESUS keywords;
- a title, manually selected by looking at the set of URLs;
- the precision and the recall in regard to the title, as far as it could be manually computed; the previous remark about the subjectivity of these notions holds.

4.2.2 Discussion of the results

First, the very high precision values are in part artificial: the title was chosen by looking at the results of the website identification process; therefore, it is normal that most webpages fit closely to the description given by the title. However, the title itself is in all but one case a label for a natural subsite of the INRIA website. In the remaining case (the fourth one), two natural subsites are merged. Precision values are thus an indication of the coherence of the results all the same.

Size	Common URL prefixes	THESUS	Title (human-generated)	Prec.	Recall

1083	`http://cdserv4.inria.fr/Volumes/DISC001/out/`	`data`
		`query`
		`database`
			Digital Symposium Collection 2000	99.9%	78.8%

126	`http://www-sop.inria.fr/croap/CFC/stalmarck/`	`intros`
		`auto`
		`ha`
			Letouzey's Stålmarck proof in Coq	100%	100%

99	`http://www-rocq.inria.fr/scilab/doc/Scilabpratique/`	`previous`
		`les`
		`matrice`
			Scilab par la pratique	100%	100%

236	`http://www-sop.inria.fr/mimosa/fp/Bigloo/`
	`http://www-sop.inria.fr/mimosa/personnel/Manuel.Serrano/`
	`http://www-sop.inria.fr/mimosa/rp/`
		`class`
		`scribe`
		`public`
			1. Manuel Serrano
			2. Mimosa Reactive Programming
				99.1%	51.0%

51	`http://www.inria.fr/rrrt/`	`rr`
		`rapport`
		`recherche`
			1988 research reports	96%	27%

Table 2: Sets of URLs obtained by applying the preflow-push algorithm to five random MCL clusters.

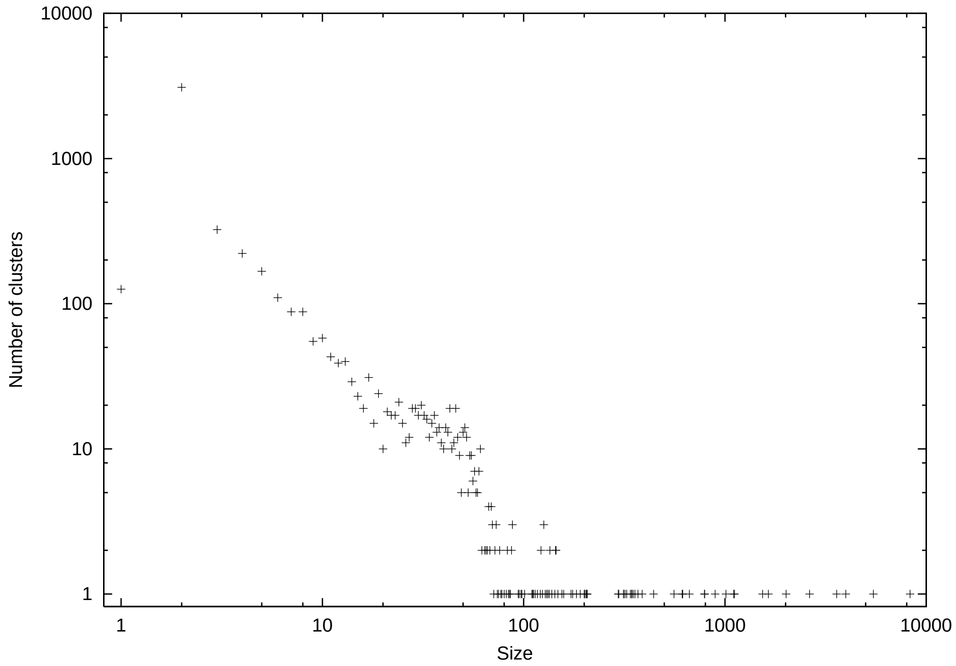

Figure 10: Distribution of the size of the clusters found by MCL.

The second and third clusters correspond perfectly to two logical websites. This is not very surprising, since these websites are much organized; here, a simple naive recursive crawl would have been enough. The first cluster is almost as satisfying, although the recall is a bit smaller than 80%. The fourth cluster is composed of two independent parts of the Mimosa project website, the webpage of a researcher (along with the programs he developed) and the webpage on Reactive Programming. The explanation of the phenomenon is quite simple: the original MCL cluster contained webpages from the two websites. With flow simulation, the two websites were (nearly) independently extended. It is a limitation of my process: flow simulation can only add webpages to the seed provided by MCL. If there was an incoherence in the MCL cluster, it remains in the extended set of webpages. A likely consequence of the presence of two different websites, not heavily linked between one another, is the low recall score. Finally, the last cluster, on 1998 INRIA research reports, has a very low recall: it is due to the organization of the corresponding website, with one index page linking to all other research reports pages, which are not linked between them. Such a website can thus easily be broken into different MCL clusters that flow simulation will not merge. Perhaps this raises the need of a way to merge closely related MCL clusters, a topic I did not have the time to study.

THESUS keywords are here almost always irrelevant. This comes from the fact that THESUS looks for keywords also around incoming links which are internal to the website, thus the keywords `intros`, `previous` which are purely navigational. It would be interesting in this case to be able to take into account external incoming hyperlinks only. I did not have the chance to access the internals of THESUS to test how that would modify the results.

Despite the aforementioned limitations of my process this experiment on random MCL clusters shows, it is rather satisfactory too. In particular, the high precision values corresponding to the high coherence of the websites found, which are almost always natural subsites of the INRIA website, is a proof of the relevance of MCL clusters in the problem of website identification. The problems raised suggest interesting directions to improve the quality of the website identification process.

The experiments presented here were only made on subsites of the INRIA website, simply because we had a direct access to it and the crawl did not require too much time and resources.

Further validation of the process would require to apply it to other websites. Still, the large number of crawled webpages and their variety is already a sign that the process may be used successfully on a large part of the World Wide Web.

5 Related works

5.1 Website identification

In most works where the notion of website appears, it is taken to be the pages hosted by a given webserver, or a lexical hierarchy of URLs (e.g. the set of URLs that share a common prefix), in addition to heuristics such as the recognition of `/~user/` part of an URL. Links between pages are usually only taken into account in an elementary way, such as in [THA99] where *clan graph* are introduced to find closely connected pages. In that paper, websites are still assumed to be on a single webserver and much importance is given in the form of the URLs. In [MV02], Mathieu and Viennot look at the matrix of the Web graph in which URLs are lexically ordered; this matrix is nearly block diagonal. Each block seems to correspond to a logical website, heavily connected inside and sparsely connected with other webpages. The authors do not suggest a way to exploit this observation and such a way would probably impose that websites are composed of contiguous URLs.

The topic of community identification (identifying sets of webpages which deal with the same topic) on the Web presents similarities with website identification. However, most approaches to the former [GKR98],[DH99] focus on *authorities* in the community. A good authority, as defined by Kleinberg in [Kle99] is a web page linked by many good *hubs*, which in turn are web pages pointing to many good authorities. Kleinberg proposes the HITS algorithm to compute the authority and hub scores of webpages. Whereas the notion of authority is relevant to the identification of communities, where individual interesting webpages are looked for, it is not the case for website identification, where all webpages of a website, even the least authoritative ones, are looked for. Another community identification technique, based on flow simulation, is discussed below.

5.2 Flow simulation

The maximum flow problem in traffic networks is a classical and much studied algorithmic problem. See for instance [CLR90] for a general presentation of the question. The most famous algorithm to solve it is the Fold-Fulkerson method [FF62] but it is not the most efficient one and it is not convenient to adapt it to work on-line, as I need. Karzanov developed the notion of *preflow* [Kar74] and Goldberg invented the preflow-push algorithm [GT88], which works better in this respect. Flake and al [FLGC02, FLG00] use a modified version of the preflow-push algorithm on the Web graph in a similar way as I do, for identifying web communities. Beside the purpose, my approach differs in the on-line adaptation of the preflow-push algorithm (Flake and al use a fixed-depth crawl whereas I do a progressive crawl with no *a priori* limits) and in the choice of the capacity of the edges (a constant capacity or a function of the edition distance between URLs). Since the goals are not the same, I cannot either compare the efficiency of the two methods.

5.3 Graph clustering

Graph clustering (the discovery of heavily connected subsets of nodes in graphs) is a rather young subfield of *cluster analysis* (the discovery of "natural" subsets of a set of elements). Traditional cluster analysis techniques use a vector representation of the elements, in which each coordinate represent a characteristic, and tend to draw boundaries in the corresponding vector space (which may or may not be a metric space). Graphs, however, are not described naturally in terms of vectors of characteristic but rather in terms of connectivity, paths between nodes, etc. That is why there is a need for original techniques for graph clustering. [Mat72] and [HS00] propose different algorithms for graph clustering, based on strong local properties which do not seem to fit well to the case of the Web graph, since websites are seldom strictly tightly connected. MCL (Markov CLuster algorithm) [vD00] does not require such conditions. Clusters in the graph are identified as the limit of a Markov process on the graph matrix, which corresponds to an alternation of flow expansion and flow inflation in a corresponding traffic network. It is to be noted that all these graph clustering algorithms only deal with undirected graphs (or, in the case of MCL, nearly undirected graphs), whereas the Web is intrinsically directed. I am not aware of any graph clustering algorithms designed to work on directed graphs. Still, as I use MCL in combination with direct flow simulation, which takes into account the fact that the fact is directed, I do not lose this information.

6 Conclusion

I presented in this paper a website identification process, based on a combination of flow simulation and graph clustering. The preflow-push algorithm, which solves the maximum flow problem in a traffic network, was adapted to the case of the World Wide Web. Logical websites are discovered by computing the minimum cut between a set of seed webpages and the rest of the Web, a seed which is computed using the Markov CLustering algorithm. The experiments realized on this process show quite satisfactory results. In particular, the technique presented here is at least in some cases superior to naive methods and to either graph clustering or flow simulation techniques alone.

The first obvious perspectives on this topic would be to improve the performance of the process, both in its execution time and in the quality of its results. Currently, the graph clustering needs to be computed on an off-line, crawled, subgraph of the Web, which can take a few days for a large graph, in order not to overload the corresponding webservers. It would thus be very useful to be able to realize an on-line computation of MCL. The adaptation is not obvious, especially because of the behavior of the inflation operator, which cannot be easily expressed in terms of classical linear algebra operators. Other improvements could be made on the online preflow-push algorithm, in particular with the choice of an efficient crawling strategy. As for the quality of the logical websites found by the process, several directions were suggested in Section 4.2.2: merging of related MCL clusters, removing of irrelevant webpages from the clusters.

The process I described in this paper is purely *structural*, based on the graph structure of the World Wide Web (and, for a lesser part, on the form of the URLs). Even if this can give good results in a large variety of cases, it is likely that no entirely structural methods will succeed in discovering the boundaries of every website. The use of *semantic* methods, based for instance on the content of the webpages, would probably be an efficient complement of the process presented here.

References

[ACMS02] Serge Abiteboul, Grégory Cobéna, Julien Massanes, and Gerald Sadrati. A first experience in archiving the French Web. In *Proceedings of the European Conference on Digital Libraries*, 2002.

[APC03] Serge Abiteboul, Mihai Preda, and Grégory Cobéna. Adaptive on-line page importance computation. In *Proceedings of the 12th international World Wide Web Conference*, Budapest, Hungary, 2003.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* The MIT Electrical Engineering and Computer Science Series. The MIT Press / McGraw-Hill Book Company, 1990.

[DH99] Jeffrey Dean and Monika R. Henzinger. Finding related pages in the World Wide Web. In *Proceedings of the 8th international World Wide Web Conference*, Toronto, Canada, 1999.

[FF62] Lestor R. Ford, Jr. and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, 1962.

[FLG00] Gary Flake, Steve Lawrence, and C. Lee Giles. Efficient identification of Web communities. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 150–160, Boston, MA, August 20–23 2000.

[FLGC02] Gary William Flake, Steve Lawrence, C. Lee Giles, and Frans Coetzee. Self-organization of the Web and identification of communities. *IEEE Computer*, 35(3):66–71, 2002.

[GKR98] David Gibson, Jon M. Kleinberg, and Prabhakar Raghavan. Inferring Web communities from link topology. In *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia*, pages 225–234, Pittsburgh, Pennsylvania, June 1998.

[GT88] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, 35(4):921–940, October 1988.

[HNVV03] Maria Halkidi, Benjamin Nguyen, Iraklis Varlamis, and Michalis Vazirgiannis. THESUS: Organizing web document collections based on semantics and clustering. *VLDB Journal Special Issue on the Semantic Web*, September 2003.

[HS00] Erez Hartuv and Ron Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(4–6):175–181, December 2000.

[Kar74] A. V. Karzanov. Determing the maximal flow in a network by the method of preflows. *Soviet Mathematics Doklady*, 15:434–437, 1974.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[Mat72] David W. Matula. k-components, clusters, and slicings in graphs. *SIAM Journal on Applied Mathematics*, 22(3):459–480, May 1972.

[MV02] Fabien Mathieu and Laurent Viennot. Structure intrinsèque du Web. Research Report 4663, INRIA, December 2002.

[PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[THA99] Loren Terveen, Will Hill, and Brian Amento. Constructing, organizing, and visualizing collections of topically related Web resources. *ACM Transactions on Computer-Human Interaction*, 6(1):67–94, 1999.

[vD00] Stijn Marinus van Dongen. *Graph Clustering by Flow Simulation.* PhD thesis, University of Utrecht, May 2000.