Identifying Websites with Flow Simulation

Pierre Senellart

École normale supérieure, 45 rue d'Ulm, F-75230 Paris Cedex 05, France
INRIA Futurs, 10 rue Jacques Monod, F-91893 Orsay Cedex, France

Abstract. We 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 in the context of Web archiving and website importance computation. To identify the boundaries of a website, we 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 using the former. Experiments on subsites of the INRIA Website 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.
- Some webservers host different websites.
- The notion of webserver is also unclear in the context of distributed webservers, mirrors or virtual web hosting.

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). 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 satisfactory. We 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.
The problem of discovering the boundaries between logical websites occurs in the topic of Web archiving [1]: once webpages are selected to be archived, what is the boundary of the corresponding websites? To be able to define websites could also lead to *website importance* computation: to devise a *SiteRank* for websites, as *PageRank* [2] is defined for webpages.

The method for website identification we 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. We use an adaptation and combination of two algorithms related to flow simulation in traffic networks: a *preflow-push* algorithm by Goldberg [3] which solves the maximum flow problem and the *Markov CLuster algorithm* (abbreviated as *MCL*) by van Dongen [4], a graph clustering algorithm. MCL is used to cluster a part of the Web in order to build *seeds* of websites 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, on the global form of the URLs.

We show in Section 2 how flow simulation and the maximum flow problem may be used to identify websites in the Web graph. We notice that in many cases, the seed of webpages the simulation starts from needs to be extended. We present in Section 3 a way to use MCL for that purpose. Experiments are presented in Section 4. Finally, we discuss related works in Section 5.

2 Flow Simulation

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

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; s and t verify: $\forall u \in S, c(u, s) = c(t, u) = 0$. The underlying directed graph is (S, A) where $A = \{(u, v) \in S^2 \mid c(u, v) > 0\}$. A flow in $\mathcal{T}$ is a function $f : S^2 \to \mathbb{R}$, satisfying the following properties:

(i) (*Symmetry*) $\forall (u, v) \in S^2, f(u, v) = -f(v, u)$
(ii) (*Capacity constraint*) $\forall (u, v) \in S^2, f(u, v) \leq c(u, v)$
(iii) (*Flow conservation*) $\forall u \in S \setminus \{s, t\}, \sum_{v \in S} f(u, v) = 0$

The *maximum flow* problem is to find a flow function f such that the maximum flow value $|f|$ is maximal; $|f|$ is being defined as the sum of flows departing from the source or, equivalently, as the sum of flows arriving to the sink:

$$|f| = \sum_{u \in S} f(s, u) = \sum_{u \in S} f(u, t) \tag{1}$$

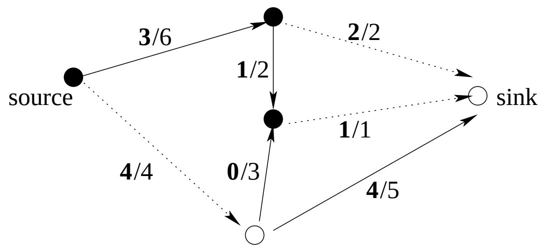

Fig. 1. Maximum flow and Minimum cut in a sample traffic network.
Plain and empty circles show the two parts of the partition.

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 [5]). 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, in the following way, as illustrated in Fig. 1: edges saturated by the maximum flow define the minimum cut.

Preflow-Push Algorithm The preflow-push algorithm, a solution to the maximum flow/minimum 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, in a preflow, a node u can have some *overflow* $o(u) = -\sum_{v \in S} f(u, v)$: it can receive more from the nodes it is pointed by than it sends to the nodes it points to. The preflow-push algorithm, 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). At each iteration, the preflow is *pushed* from a node with overflow to a lower node. 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.
This algorithm can be demonstrated to converge toward a maximum flow in $\mathcal{T}$ (cf [3]), with a complexity of $O(|S|^2|A|)$ ($|A|$ is the number of edges with non-zero capacity), whatever the strategy for selecting nodes with overflow may be.

Adaptation to the Web 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 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 is computed as a minimum cut in a traffic network whose underlying graph is the Web graph. More formally, let *Seed* be a set of seed webpages, characteristic of the website we would like to compute the borders of, and *sim* a similarity function over webpages. We 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 $u \in S \setminus \{s, t\}$, $c(u, t) = \varepsilon$

  (ε is a small value, much smaller than average similarity values).

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. Instead of an *ad hoc* solution raising the similarity of webpages near the source or a complex semantic similarity function, we chose to use a function of the edit distance between the URLs of the webpages: even if a website span over several webservers, the URLs of the pages tend to look similar. If $ed(u, v)$ is the edit distance between URLs u and v and σ is a typical standard deviation for the edit distance over the set of considered URLs, the similarity function *sim* is defined empirically to be:

$$sim(u, v) = e^{-\frac{ed(u,v)^2}{2\sigma^2}} \tag{2}$$

On-line Preflow-Push 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.

The preflow-push algorithm can be made on-line in a straightforward way: webpages with overflow are progressively crawled and dealt with (pushed or raised). We 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.

Several strategies adapted to crawling can be chosen. We 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 in the experiments we 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).
Applying this version of the preflow-push algorithm on a seed of characteristic webpages of a website gives quite good results on small or medium-sized, well-organized, websites. When the website is very large or not well organized, however, the algorithm only retrieves a small proportion of the webpages of the real website (cf Table 1). The problem is that a small seed is not sufficient to discover the entire website. Thus, we need a way to extend automatically the seed before carrying out the flow simulation.

3 Extension of the Seed

We use MCL, a graph clustering algorithm by van Dongen [4], to extend the seed used by the online push-preflow algorithm. After presenting it very briefly, we describe how it can be used in the context of the Web, and the role it plays in the website identification process we present in this paper.

MCL (Markov CLuster Algorithm) Let G be a (possibly weighted) directed graph and M_G its associated matrix. Let $\mathcal{M}_G$ be the matrix obtained from the transpose of M_G by normalizing each column. $\mathcal{M}_G$ is a column stochastic matrix. 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 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 obtaining a graph clustering algorithm, but is not sufficient in itself. 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).
Note that this algorithm is not guaranteed to converge. There are even examples of non-convergence. However, 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 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 [4].
The complexity of the MCL algorithm, as it is presented above, is $O(|S|^3)$ where $|S|$ is the number of nodes in G. Simple pruning of each column of the matrix may drastically lower the complexity to $O(|S|)$. Experiments

show no significant effect of the pruning on the convergence and results of the algorithm.

Flow Simulation from MCL Clusters Ideally, a graph clustering algorithm such as MCL would be applied to a large portion of the Web graph "as is", thus discovering the different logical websites of the whole Web, 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. 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, we do not use MCL to extract the logical websites, but to extend the seed the on-line preflow-push starts from. The entire process is shown on Fig. 2.

Process for Identifying a Website W

1. Find 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); crawl and build the corresponding subgraph G_S of the Web graph.
2. 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).
3. Find the obtained cluster K which is the most relevant to W. It may be identified by finding the cluster which contains the largest number of URLs containing some given keyword.
4. 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.

Fig. 2. Website identification process.

4 Experiments

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

The GEMO *Website* Our main experiment was on the website of our research team, 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. Following the process described in Fig. 2, a MCL clustering was performed on the underlying undirected graph. 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.
Table 1 shows some data about the website found by the algorithm, along with results of other simpler methods:

Table 1. Gemo website, according to different methods.

	Number of Pages	Precision	Recall
Flow Simulation	8	87.5%	1.3%
MCL	320	99.7%	33.0%
MCL + Flow Simulation	788	90.4%	86.4%
http://www-rocq.inria.fr/verso/*	221	100%	44.4%
http://*.inria.fr/verso/*	683	100%	68.6%

- **Flow Simulation**: direct on-line preflow-push, starting from the website entry page.
- **MCL**: clusters from MCL, without flow simulation
- **MCL + Flow Simulation**: process described above
- http://www-rocq.inria.fr/verso/*: "naive" recursive crawl of the hierarchy of URLs starting from the website entry page.
- http://*.inria.fr/verso/*: recursive crawl of the hierarchy of URLs starting from /verso/ on every webserver of the `inria.fr` domain. This method use the human knowledge that dynamic and static webpages of the Gemo website are hosted on different webservers.

Precision and recall, as presented in Table 1, are defined as follows:

$$precision(W) = \frac{\text{number of relevant webpages in } W}{|W|} \tag{3}$$

$$recall(W) = \frac{\text{number of relevant webpages in } W}{\text{total number of relevant webpages}} \tag{4}$$

The notion of relevant webpages is somewhat subjective, therefore a 100% precision or recall is not a realistic objective.
As noted in Section 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. The complete process, however, 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 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 we used. Even *ad hoc* techniques like the last one (proposed by a member of the group) do not retrieve as much relevant webpages.

The results of our 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.

Another important point is that our method is entirely dependent of the clustering obtained by MCL, in particular if there is no clear “most relevant cluster” for a website (either because the website is split into many clusters or because it is merged with other logical websites). We had this issue on the Gemo website on another crawl of the inria.fr domain, one year later: due to server reconfigurations, a large proportion of the Gemo website was disallowed to crawlers. The remaining webpages did not form a distinct MCL cluster. Still, human merging and splitting of a few MCL clusters would be enough to obtain a proper seed on which flow simulation can be run.

Flow Simulation from Random MCL Clusters 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 to the 147, 784 nodes INRIA subgraph as before. As the main cluster was huge (43, 811 nodes), we clustered it furthermore. We obtained 4, 303 clusters, whose size went from 1 to 16, 941 pages.

4 clusters were then chosen at random among clusters whose size is greater than 10 (too small clusters often do not have much interest). The preflow-push algorithm was applied to each of them, resulting in 4 sets of URLs described in Table 2 by their size, a title (manually selected by looking at the set of URLs) and the precision and recall in regard to the title, manually computed (as precisely as it could be).

Table 2. Sets of URLs obtained by flow simulation on random MCL clusters.

Size	Title (human-generated)	Precision	Recall
24	Presentation of the RAP Project Proposition	100.0%	100.0%
37	Praxitele Transportation System	94.6%	97.2%
262	Ocaml - Practices and Principles	99.6%	95.6%
56	OSCAR 2004 Workshop	94.6%	100.0%

The very high precision values are in part artificial: since the title was chosen by looking at the results of the website identification process, it is normal that most webpages fit closely to the description given by the title. The high recall values give a good validation that our process effectively returns (nearly) complete logical websites.

5 Related Work

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 in an URL. Links between pages are usually only taken into account in an elementary way, such as in [6] where *clan graphs* 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 [7], Mathieu proposes a way to partition the Web by using the fact that the matrix of the Web graph in which URLs are lexically ordered is nearly block diagonal. Each block seems to correspond to a logical website, heavily connected inside and sparsely connected with other webpages.
The maximum flow problem in traffic networks is a classical and much studied algorithmic problem. Karzanov developed the notion of *preflow* [8] and Goldberg invented the preflow-push algorithm [3], which works better in this respect. Flake et al [9] use a modified version of the preflow-push algorithm on the Web graph in a similar way as we do, for identifying Web communities. Beside the purpose, our approach differs in the on-line adaptation of the preflow-push algorithm, in the choice of the capacity of the edges and in the use of an extended seed.
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). [10] and [11] 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) [4], presented in Section 3, does not require such conditions.

6 Conclusion

We 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 CLuster algorithm. The experiments realized on this process show quite satisfactory results. In particular, the technique presented here showed to be superior to naive methods and to either graph clustering or flow simulation techniques alone.
The first obvious perspective 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. Finally, a semi-automatic method, with the possibility of splitting and merging selected MCL clusters, would allow a more precise selection of the website to identify.
The process we 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.

Acknowledgments Thanks to Serge Abiteboul and Grégory Cobéna for their advice on this research topic.

References

1. Abiteboul, S., Cobéna, G., Massanes, J., Sadrati, G.: A first experience in archiving the French Web. In: Proceedings of the European Conference on Digital Libraries. (2002)
2. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project (1998)
3. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum-flow problem. Journal of the ACM **35** (1988) 921–940
4. van Dongen, S.M.: Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht (2000)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. The MIT Electrical Engineering and Computer Science Series. The MIT Press / McGraw-Hill Book Company (1990)
6. Terveen, L., Hill, W., Amento, B.: Constructing, organizing, and visualizing collections of topically related Web resources. ACM Transactions on Computer-Human Interaction **6** (1999) 67–94
7. Mathieu, F.: Mesures d'Importance à la PageRank. PhD thesis, Université Montpellier II (2004)
8. Karzanov, A.V.: Determing the maximal flow in a network by the method of preflows. Soviet Mathematics Doklady **15** (1974) 434–437
9. Flake, G., Lawrence, S., Giles, C.L.: Efficient identification of Web communities. In: Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA (2000) 150–160
10. Matula, D.W.: k-components, clusters, and slicings in graphs. SIAM Journal on Applied Mathematics **22** (1972) 459–480
11. Hartuv, E., Shamir, R.: A clustering algorithm based on graph connectivity. Information Processing Letters **76** (2000) 175–181