FPCluster: an Efficient Out-of-core Clustering Strategy without a Similarity Metric

Douglas E. V. Pires[1,2], Luam C. Totti[1], Rubens E. A. Moreira[1], Elverton C. Fazzion[1], Osvaldo L. H. M. Fonseca[1], Wagner Meira Jr.[1], Raquel C. de Melo-Minardi[1], Dorgival Guedes[1,3]

[1] Department of Computer Science — Universidade Federal de Minas Gerais
[2] Department of Biochemistry and Immunology — Universidade Federal de Minas Gerais
[3] International Computer Science Institute, Berkeley
{dpires, luamct, rubens, elverton, osvaldo.morais, meira, raquelcm, dorgival}@dcc.ufmg.br

Abstract.

Clustering is one of the most popular and relevant data mining tasks. Two challenges for determining clusters are the volume of data to be grouped and the difficulty in defining a similarity metric applicable to the entire data set. In this work we present FPCluster, a new clustering algorithm that addresses both problems. The algorithm is based on building out-of-core frequent pattern trees, a data structure originally proposed for mining patterns. Additionally, the algorithm transparently handles missing features, a common constraint in real case scenarios. We applied FPCluster to two real scenarios: characterization of spam campaigns and clustering of protein families. We evaluated both the quality of the obtained groups and the computational efficiency of the proposed strategy. In particular, we achieved precision above 90% while the storage demand increased sub-linearly.

Categories and Subject Descriptors: H.Information Systems [**H.2. Database Management**]: H.2.8 Data Mining

General Terms: Algorithms, Data mining

Keywords: Clustering, out-of-core, protein families, spam detection

1. INTRODUCTION

The huge amount of useful information stored in large data sets derived from real scenarios always attracted great interest in areas such as data mining. In those scenarios, clustering algorithms are valuable tools for knowledge discovery and pattern detection tasks. Such algorithms generally aim to divide a set of elements into different groups according to a given established similarity criterion. However, most of these algorithms have some notorious limitations, such as the popular k-means, which does not scale well when the database exceeds the available memory, and also requires a similarity metric that can be applied between elements of the data set. In addition, it is necessary to define the number of groups to be created as an input parameter. Moreover, k-means does not support missing attributes, hence all elements must have the same dimensionality.

In this work we propose *FPCluster*, a strategy that addresses those aspects. Our algorithm uses a frequent pattern tree (FPTree [Han et al. 2000]) to capture the invariant attributes of groups of elements without any prior knowledge of the structure of data or definition of a similarity metric. It then uses a cutoff methodology to detect clusters in the tree. Additionally, the proposed algorithm offers an out-of-core strategy to enable the processing of data volumes that main-memory-based ap-

This work was supported by the Brazilian agencies CNPq, CAPES, and Fapemig. It was also supported by NIC.BR, InWeb (the Brazilian National Institute for Science and Technology for the Web), and by a fellowship from the Brazil ICSI visitor program, from Movimento Brasil Competitivo.

proaches would not be able to deal with. This work presents, as well, two distinct case studies to validate FPCluster: characterization of *spam* campaigns and clusterization of proteins into families. We conclude with an analysis of FPCluster's computational performance.

The remainder of this text is organized as follows. The next section presents a review of related work in the literature. Section 3 contains a detailed description of the proposed algorithm, and Section 4 describes the evaluation methodology and presents the experimental results obtained. The final section presents the conclusions and future directions.

2. RELATED WORK

Clustering algorithms may be divided in two broad groups with respect whether they require parameter calibration or not. FPCluster falls in the non-parametric category, *i.e.*, it does not require the definition of the number of clusters as an input parameter. We focus our discussion of related work on algorithms in that category.

There are several studies in the literature describing non-parametric clustering methods. Some approaches aim to define the number of groups and then apply parameterized techniques [Pelleg and Moore 2000]. Other approaches perform a partition of the data, dividing it into multiple clusters, merging them together until a given stopping criterion is satisfied [Frigui and Krishnapuram 1998]. Statistical methods are also popular to determine the optimal number of groups [Pelleg and Moore 2000].

More recent studies continue to explore new ways to determine clusters from irregular data sets. [Tang et al. 2010] proposed a similarity metric based on the dispersion analysis of the neighborhood, with good clustering results. [Lazic et al. 2009] reduced the clustering problem to finding the best subset of linear spaces among candidates generated from the input data.

Another common limitation of clustering algorithms is not being able to processing large volumes of data. Some studies explore compression techniques, like [Sassi and Grissa 2009], which use compression techniques based on cluster quality metrics. Some classification-related works propose memory management techniques applicable to clusters, like [Yu et al. 2010], which proposed a block minimization framework due to restrictions on memory usage.

To the best of our knowledge, there are only a few works in the literature about the use of frequent pattern trees as a clustering technique. The FP-Growth algorithm was used by [Akbar and Angryk 2008] to identify sub-graphs of frequent terms occurring in documents. Then a similarity metric was proposed according to the frequency of such graphs. Another recent work uses a modification of FP-Growth to solve time scheduling problems [Shatnawi et al. 2010].

It is important to note that our approach differs from others by using frequent pattern trees to create an out-of-core clustering strategy capable of efficiently and effectively handling large volumes of data, without requiring an explicit similarity or distance metric. The details of the proposed algorithm are presented in the next section.

3. THE FPCLUSTER ALGORITHM

In this section we describe our out-of-core strategy for clustering without a similarity metric and how the principles that underlie this strategy apply to the real data sets used in the validation process. FPCluster, the proposed algorithm, identifies the invariant attributes of the transactions being clustered, organizes them hierarchically, and uses them as a discriminating criterion in the clustering step.

Clustering algorithms are usually one of the several steps of a clustering methodology. Figure 1 summarizes the steps of our proposed methodology. The process may be divided in three main phases:

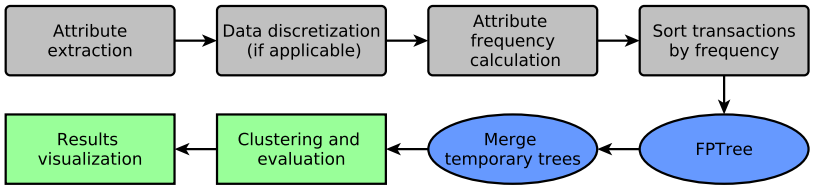

Fig. 1. *Workflow.* The steps of the proposed clustering methodology that employs FPCluster: *Pre-processing*, *FPTree generation*, *evaluation* and *visualization of the obtained results.*

the pre-processing of the data, which includes the extraction and mapping of the attributes (*i.e.*, features), the construction of the out-of-core frequent pattern tree, and the generation, evaluation and visualization of the identified clusters.

In the first phase, *Pre-processing*, a set of relevant attributes is extracted from each element of the data set to represent them as transactions. In the study of *spam campaigns*, we considered relevant, for the sake of spam detection, attributes such as the each message's language, subject, type (e.g., HTML, text, picture), and embedded URLs. In the case of proteins, some of the attributes used were the number of secondary structures, the number of amino acid residues, polar and hydrophobic, and the volume of each protein. One advantage of the FPCluster approach is that the transactions within a group could have different number of attributes (*i.e.*, have missing attributes), what would be transparently handled by the FPTree.

Based on each element's attribute set, a frequent pattern tree (FPTree) is generated. An FPTree is a prefix rooted tree, extensively used in association rule mining, where the nodes carry attributes and its frequency. Each element or transaction is represented as a path in the tree, where each node is an attribute and the attributes are organized in a non-increasing fashion from root to leaves according to their frequency. The path length (*i.e.*, the number of attributes per element) may vary, which gives FPCluster the ability to handle missing data or transactions with different dimensionality. The attributes are then sorted by their frequency in the database and inserted in a way that elements with attributes in common share a path in the tree. Consequently, globally common attributes are at the highest levels and less frequent attributes are at lower levels. Figure 2 shows a standard FPTree generation process. The next section details our out-of-core FPTree implementation strategy, designed to handle large real-world transaction data sets.

Finally, after constructing the tree, we identify the nodes with f or more children (fan-out) starting with a minimum depth d. Those nodes are the cutoff points of the clustering algorithm and all paths that pass through one such node represent elements in the same cluster. Thus, elements that have

Transaction data set

TID	Items
1	{a, d, c, b}
2	{f, b, e, a}
3	{g, a}
4	{g, h, c}
5	{c, b, a}

(a)

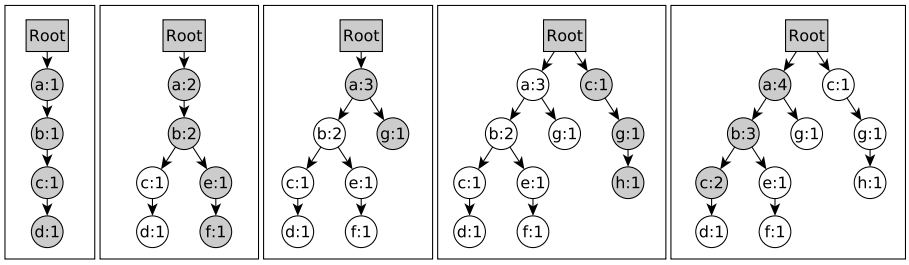

Fig. 2. *FPTree construction.* Using a transaction data set (a), steps (b) to (f) show the FPTree's structure after the insertion of each of the five transactions considered. The rectangles represent the tree root and the circles denote the attributes and their current frequency in the tree. Notice that the attributes are ordered by global frequency before insertion. In this example, no minimum attribute-frequency threshold is considered.

many frequent attributes in common are clustered together.

In the case of spam campaigns, the intuition is that *spammers* use obfuscation techniques in some of the messages attributes, such as the insertion of random strings in URLs. Such attributes make the number of children of a node raise abruptly, suggesting that such cutoff point should define a spam dissemination campaign.

While clustering proteins into families, one might expect that there would be a set of attributes that identified and characterized each protein family (set of proteins functionally or structurally related). With the knowledge of those attributes, it would be possible to analyze the differences between protein families, as well as to infer evolutionary factors that may explain their separation.

The restriction of a minimum depth d in the tree to define clusters is necessary because there may be attributes close to the root that can have a wide range of values, without necessarily being discriminative for defining spam campaigns or protein families. Moreover, it is reasonable to expect that there should be a minimum set of common attributes to identify a spam campaign or a protein family (*i.e.*, a set of invariant features that identify and characterize that group), and the size of that set might vary with the problem domain.

3.1 *Out-of-core* strategy to cluster large volumes of data

The proposed out-of-core strategy for clustering large volumes of data is described next. Algorithm 1 shows the main steps of the strategy.

Our approach makes good use the available memory by inserting transactions into the FPTree until a maximum number of nodes in memory is reached (lines 1-2). This parameter, defined by the user, limits the maximum amount of memory used by the FPTree at any point during its execution. Once this limit is reached, the information and structure of the tree in primary memory are then written to a file using a depth-first search (DFS) traversal and the memory previously occupied is released for other transactions (lines 3-5). One file is created for each sub-tree (or temporary tree) and the process repeats until the whole input file is processed. At that point, the transactions are fragmented into a set of sub-trees and a merging step is initiated. Figure 3 demonstrates how two temporary trees are combined. More specifically, from the sub-trees, all paths from root to leaves are computed, written in an auxiliary file and sorted (lines 6-7). From these paths we create a global ordering of attributes for the combined tree so that each node receives an incremental identifier denoting their order in the final DFS-fashion tree resulting from the merge of all sub-trees (line 8). This step estimates, therefore, the structure of the resulting tree.

Nevertheless, the structure of the tree in DFS makes it difficult to update node properties such as

Algorithm 1 Out-of-core FPTree

Input: $TransactionSet, MaxNumNodes, FPTree$
Output: $TreeFile$

```
1: for all Transaction ∈ (TransactionSet) do
2:     insert(Transaction, FPTree)
3:     if (NumNodes ≥ MaxNumNodes) then
4:         writeToTempFileDFS(FPTree)
5:         NumNodes ← 0
6: PathFile ← getPaths(FPTree)
7: PathFileSorted ← sortPaths(PathFile)
8: DFSFile ← getOrdering(PathFileSorted)
9: BFSFile ← DFSToBFS(DFSFile)
10: TreeFile ← DFSToBFS(BFSFile)
11: return TreeFile
```

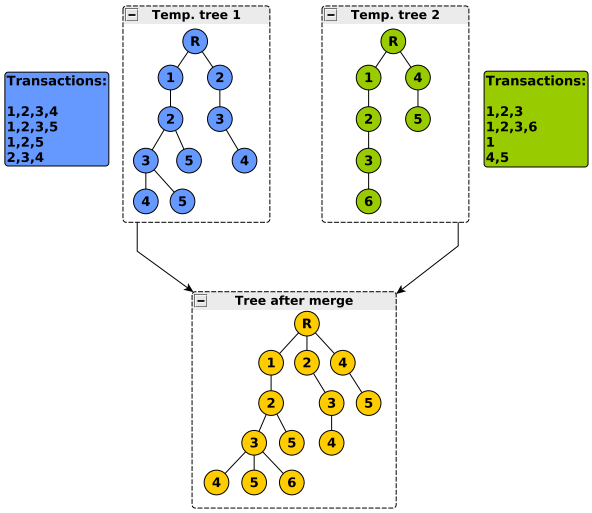

Fig. 3. Example of how two temporary trees are merged.

frequency and fan-out (used in the later stage, as a criterion to identify clusters), since this information is scattered across the multiple partial files. In other words, computing the impact of merging various branches of the sub-trees requires an alternative representation of the tree.

We solve this problem by changing the representation of the sub-trees to a breadth-first search (BFS) traversal, what is accomplished by sorting the file by the level of the nodes (line 9). Identical attributes from the same level are merged, and their combined frequency and fan-out are properly updated.

After the necessary node properties are computed, the BFS representation is converted back to DFS (line 10), simply by ordering the identifiers of each node calculated in the previous step. The result of this step is the resulting tree (line 11). The representation in DFS is more appropriate for the tree traversal used to identify the clusters, as explained before.

4. EXPERIMENTAL EVALUATION

To evaluate our proposed technique, we first discuss the methodology and data sets used, then show some performance numbers and finally present two case studies, in different application contexts.

4.1 Evaluation methodology

Our evaluation methodology focuses on both clustering quality and computational performance.

Regarding computational performance, we considered the two major dimensions that had an impact on the algorithm's running time: the amount of memory available for the construction of the tree (represented as the maximum number of nodes kept in memory) and the size of the input file (number of transactions to be processed).

In terms of clustering quality, we adopted some of the most commonly used metrics for cluster evaluation, considering the particular conditions from each application scenario. In the case of protein families, we used a well-studied data set for which the classification was known. From that classification, we evaluated the clusters obtained. Based o the previous classification available, we also computed typical metrics such as precision and recall for each clustering parameter setting (minimum height and fan-out). In the case of spam campaigns, we had no reliable previous cluster assignments

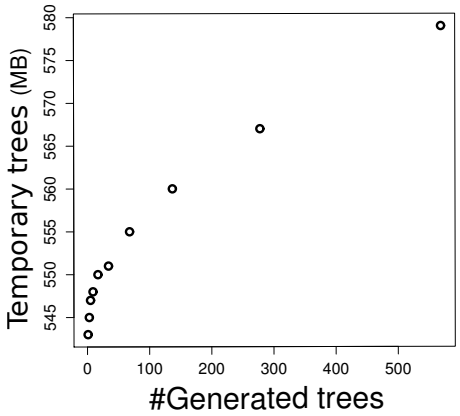

Fig. 4. Correlation between the number of temporary trees created and the total disk space used.

to validate the solution obtained, and we found no previous work that had grouped spam campaigns with good accuracy to be used as a baseline. We felt the need for some kind of direct validation and, due to the large volume of data (millions of emails per day), a sampling approach was adopted. A statistically significant set of clusters was selected and manually classified [Hamburg and Lubov 1974]. The results of this case study have a confidence of 95% with a margin of error of 5%.

4.2 Performance analysis

This section discusses the trade-offs associated with our out-of-core strategy, in particular the behavior of the algorithm as a function of memory constraints. The experiments reported here were executed on an Intel Core 2 Duo 2.13 GHz, with 2 GB of memory, with Ubuntu kernel 2.6.32 installed. A large-scale data set composed of features extracted from more than 2.3 million spam messages was used in these experiments.

One important aspect is the amount of external memory required for the out-of-core strategy when main memory is limited. When the memory available to build the FPTree decreases, there is an increase in the number of sub-trees created. Figure 4 shows the correlation between the number of sub-trees created and the total disk space used by them. It is possible to verify that there is a small impact associated with sub-tree fragmentation as the available memory to build the FPTree decreases and, consequently, the number of temporary trees increases. It is important to point out that this is a characteristic of the database considered, and the impact may vary for different databases.

The merge step of temporary trees depends uniquely on the amount of temporary data to be processed. The overall space usage by the sub-trees for different memory constraints does not vary significantly and it also results in just slight variations in total execution times, but presenting a linear increase, as expected. For instance, when we clustered one-million transactions and varied the memory limit parameter from 1 to 500 nodes, FPCluster had an average execution time of 1,942 s, with a relative standard deviation of just 2% over that range.

4.3 Case study: protein families

In our first case study we aim to cluster proteins into their biological families.

We built a data set containing structures of globular proteins from two different domains and functions. Myoglobins and hemoglobins were selected, both all-alpha proteins according to SCOP [Murzin et al. 1995], which binds the prosthetic group *heme* responsible for interacting with oxygen molecules. These families are well studied in the literature and perform, respectively, storage and transport of

Table I. Comparative results between FPTree and K-means to the protein families data set.

Method	Micro Precision	Micro Recall
FPTree ($h = 5, f = 2$)	0.996	0.436
K-means ($N = 10$)	0.845	0.230
K-means ($N = 2$)	0.754	0.751

oxygen. All available structures in the Protein Data Bank [Berman et al. 2002] belonging to these two families were collected, in a total of 464 chains that were filtered and pre-processed by the *PDB Enhanced Structures Toolkit* (PDBest [Pires et al. 2007]).

After the filtering step, a set of attributes was extracted from each structure: number of secondary structures (α-helices, β-sheets, and turns), number of amino acid residues (size of the protein primary sequence), total, polar and hydrophobic volumes, and diameter. Volumes were computed using an implementation of Richards' algorithm [Tsai et al. 1999; Richards 1977], the other attributes were extracted by Perl scripts implemented to that end.

The extracted attributes were discretized and then submitted to the FPTree. Figure 5 shows the resulting tree. It can be seen that, in addition to the proper separation of the two protein classes, there were sub-groups within the families that should to be considered and explained. A manual analysis of those sub-groups showed that there was a separation between the *alpha* and *beta* chains of the hemoglobins (which are dimers) and between species in the case of myoglobin (which are monomers).

Figure 6 (graph to the left) demonstrates a tradeoff between precision and recall with respect to the cutoff point used to determine clusters, which considers minimum values of depth and fan-out. The drop in recall, and subsequent increase in accuracy, is due to fragmentation of clusters as seen in the graph to the right in the same figure.

Additionally, we performed an experiment comparing a well-established clustering algorithm in the literature (in this case, K-means) and FPCluster. Table I summarizes the evaluation of the clusters obtained by the two techniques. We show the results for the best cutoff configuration (*i.e.*, minimum fan-out of 2 and minimum height of 5) and two configurations for K-means, using as parameter the actual number of groups defined and the same number of sub-groups found in the FPTree solution. We observed that the tradeoff between precision and recall remains, and the use of the cutoff generated more homogeneous groups than K-means for the same number of groups.

4.4 Case study: spam campaigns

The spam messages analyzed in this work were collected by 10 low-interaction honeypots installed in Brazilian broadband networks from 5 different operators (cable and ADSL). A central server is also part of the architecture, configured to gather the spams captured as well as to perform a periodic monitoring of the honeypots [Steding-Jessen et al. 2008].

The honeypots were configured to simulate open proxies and open mail relays, components typically abused by spammers. A spammer who tried to abuse a honeypots to send spam would be led to believe that the messages were successfully delivered, but no spam was actually sent.

To capture the messages, the software Honeyd was used in conjunction with emulation subsystems for SMTP mail relays and HTTP and SOCKS proxies available in that software or developed in previous work [Guerra et al. 2008]. All transactions executed by the subsystems were stored in logs with information such as date and time, source IP, IP and port of intended destination, and protocol version used. All spams collected by the honeypots were downloaded at regular intervals by a central server via an encrypted tunnel [Steding-Jessen et al. 2008]. A total of 2,378,413 messages were used, corresponding to traffic observed during one day in June 2011. Those messages originated from 2,316 distinct IPs, assigned to 93 different countries (*i.e*, different Country Codes).

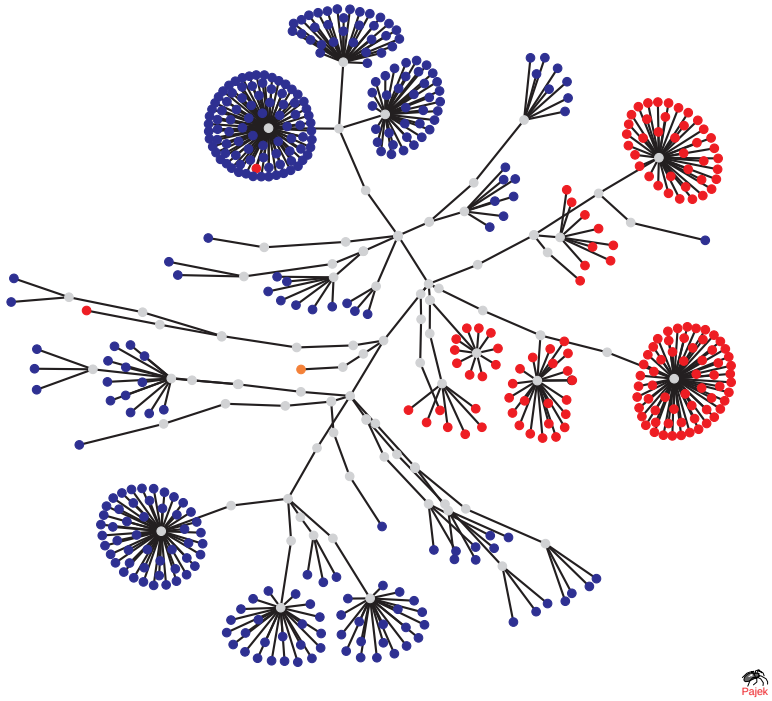

Fig. 5. FPTree for the two globular protein families, myoglobin (in blue) and hemoglobin (in red).

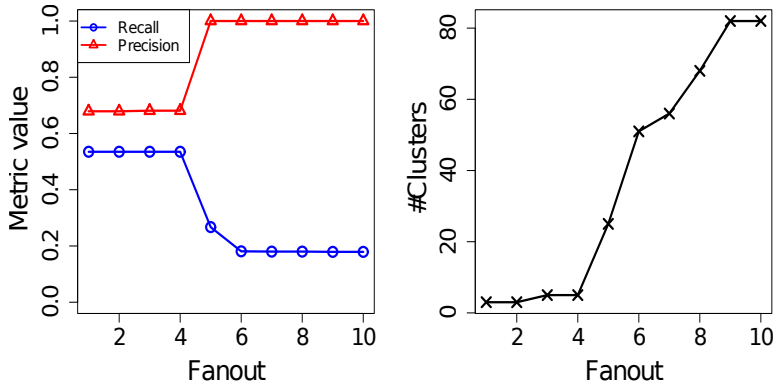

Fig. 6. Tradeoff between precision and recall in the cutoff criterion to obtain the clusters for the 4^{th} level of the tree.

As expected, the resulting FPTree showed that there are many cutoff points that, in fact, corresponded to obfuscation strategies of spam campaigns. Several e-mails share the same attributes in the tree, to the point where there is a sharp increase in the fan-out due to the presence of an obfuscation attribute, such as a portion of a URL containing a random sequence of characters.

We calculated clustering evaluation metrics for the sampled data, more specifically Weighted Purity and Entropy, which are equal to 0.939 and 0.025, respectively. We also calculated Micro Precision and Recall, and obtained 0.939 and 0.415, respectively. In summary, FPCluster achieved good values of accuracy at the expense of recall. In practice, it means that FPCluster has good performance in clustering similar elements, however it divides these elements into multiple sub-groups. Such excessive number of clusters may be a result of frequent attributes that should not be taken into account in the

algorithm, because they create alternate paths in the tree, segmenting one group into several smaller ones.

In addition, significant work remains to be done towards establishing good cutoff parameters for the algorithm. Due to the manual validation of the results, a calibration would be very costly and is left as future work.

An interesting result observed is that the non-parametric nature of the algorithm makes it very flexible for identifying campaigns. Messages were properly grouped in the same campaign, even when they contained URLs pointing to different domains, after all redirections were considered. This suggests that a spammer was promoting different URLs in the same campaign, possibly selling a spamming service and using the same strategies and templates to multiple clients. Any clustering algorithm that restricted campaigns rigorously by the URL of each message (a common form of analysis in that field) would certainly fail when trying to group those messages.

5. CONCLUSIONS AND FUTURE WORK

Clustering large volumes of data is a challenging task due to several aspects. In this paper we address two such challenges, namely the efficient use of limited computing resources (for example, through effective out-of-core strategies), and the definition of a similarity or distance metric appropriate to a whole data set.

This work proposes FPCluster, a new out-of-core clustering algorithm based on frequent pattern trees, which does not require the definition of a global similarity metric in advance. The strategy proposed proved to be effective and generic, being evaluated experimentally in two quite different contexts: clustering spam campaigns and protein families.

We distinguish at least three future work directions: the first one is the development of an incremental version of FPCluster; a second one would aimed at parallelizing it; and finally, we want to study and evaluate different cutoff criteria and their impact on the clustering quality.

REFERENCES

Akbar, M. and Angryk, R. A. Frequent pattern-growth approach for document organization. In *Proceeding of the 2nd international workshop on Ontologies and information systems for the semantic web*. Napa Valley, California, USA, pp. 77–82, 2008.

Berman, H. M., Battistuz, T., Bhat, T. N., Bluhm, W. F., Bourne, P. E., Burkhardt, K., Feng, Z., Gilliland, G. L., Iype, L., Jain, S., Fagan, P., Marvin, J., Padilla, D., Ravichandran, V., Schneider, B., Thanki, N., Weissig, H., Westbrook, J. D., and Zardecki, C. The protein data bank. *Acta Crystallographica Section D: Biological Crystallography* 58 (Pt 6 No 1): 899–907, 2002.

Frigui, H. and Krishnapuram, R. A robust competitive clustering algorithm with applications in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. 21, pp. 450–465, 1998.

Guerra, P. H. C., Pires, D. E. V., Guedes, D., Wagner Meira, J., Hoepers, C., and Steding-Jessen, K. A campaign-based characterization of spamming strategies. In *Proceedings of the 5th Conference on e-mail and anti-spam (CEAS)*. Mountain View, CA, 2008.

Hamburg, M. and Lubov, A. *Basic statistics: A modern approach*. Harcourt Brace Jovanovich, 1974.

Han, J., Pei, J., and Yin, Y. Mining frequent patterns without candidate generation. In *Proceedings of the ACM SIGMOD Conference*, W. Chen, J. F. Naughton, and P. A. Bernstein (Eds.). pp. 1–12, 2000.

Lazic, N., Givoni, I. E., Frey, B. J., and Aarabi, P. Floss: Facility location for subspace segmentation. In *Proceedings of the International Conference on Computer Vision*. pp. 825–832, 2009.

Murzin, A. G., Brenner, S. E., Hubbard, T., and Chothia, C. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology* 247 (4): 536–40, 1995.

Pelleg, D. and Moore, A. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, pp. 727–734, 2000.

Pires, D. E. V., da Silveira, C. H., Santoro, M. M., and Meira Jr, W. PDBEST: PDB enhanced structures toolkit. In *Proceedings of the 3rd International Conference of Brazilian Association for Bioinformatics and Computational Biology*, 2007.

Richards, F. M. Areas, volumes, packing and protein structure. *Annual Review of Biophysics and Bioengineering* vol. 6, pp. 151–76, 1977.

Sassi, M. and Grissa, A. Clustering large data sets based on data compression technique and weighted quality measures. In *Proceedings of the 18th international conference on Fuzzy Systems*. Jeju Island, Korea, pp. 396–402, 2009.

Shatnawi, S., Al-Rababah, K., and Bani-Ismail, B. Applying a novel clustering technique based on fp-tree to university timetabling problem: A case study. In *Proceedings of the International Conference on Computer Engineering and Systems*. Vol. 63. pp. 314–319, 2010.

Steding-Jessen, K., Vijaykumar, N., and Montes, A. Using low-interaction honeypots to study the abuse of open proxies to send spam. *INFOCOMP Journal of Computer Science* 7 (1): 45–53, 2008.

Tang, D., Zhu, Q., Cao, Y., and Yang, F. An efficient clustering algorithm for irregularly shaped clusters. *IEICE TRANSACTIONS on Information and Systems* 93 (2): 384–387, 2010.

Tsai, J., Taylor, R., Chothia, C., and Gerstein, M. The packing density in proteins: standard radii and volumes. *Journal of Molecular Biology* 290 (1): 253–66, July, 1999.

Yu, H.-F., Hsieh, C.-J., Chang, K.-W., and Lin, C.-J. Large linear classification when data cannot fit in memory. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. Washington, DC, USA, pp. 833–842, 2010.