Dynamic Credit-Card Fraud Profiling

Marc Damez, Marie-Jeanne Lesot, Adrien Revault d'Allonnes

Dynamic Credit-Card Fraud Profiling

Marc Damez, Marie-Jeanne Lesot, and Adrien Revault d'Allonnes

LIP6, Université Pierre et Marie Curie-Paris 6, UMR7606
4 place Jussieu, Paris cedex 05, 75252, France
{marc.damez,marie-jeanne.lesot,adrien.revault-d'allonnes}@lip6.fr

Abstract. The paper proposes a scalable incremental clustering algorithm to process heterogeneous data-streams, described by both categorical and numeric features, and its application to the domain of credit-card fraud analysis, to establish dynamic frauds profiles. The aim is to identify subgroups of frauds exhibiting similar properties and to study their temporal evolution and, in particular, the emergence of fraudster behaviours. The application to real data corresponding to a one year fraud stream highlights the relevance of the approach that leads to the identification of significant profiles.

Keywords: Incremental Clustering, Data-Streams, Heterogeneous Data, Bank Fraud, Credit Card Security

1 Introduction

Credit and debit cards have become ubiquitous modes of payment, which have brought with them the important issue, both for banks and card-holders, of card fraud [1]. The ensuing economic losses have motivated a vast field of study in the machine learning community [2–4], in particular concerned with electronic business, both for supervised –fraud detection– and unsupervised –fraud characterisation– learning tasks.

This paper considers the latter, i.e. the identification of ever changing groups of similar frauds. This problem is challenging in its twofold dynamic nature. First, fraudsters are inventive and continuously adapt to circumvent anti-fraud policies, elaborating new types of frauds. Second, the data are constantly incoming, building a never-ending stream. Finally, the very nature of the heterogeneous data, described by both numeric and categorical attributes, renders part of the classic methods unusable.

In this paper, we propose a methodology to process incoming streams presenting these characteristics and study the derived profiles and the dynamics of their contents. Contrary to most clustering tasks, the method proposed in this paper is less interested in summarising the data into abstract and possibly non-existent prototypes than it is in identifying precisely observable behaviours, those which are most likely to belong to an actual type of fraudster. The proposed method is tested on a real dataset representing one year of frauds

The paper is organised as follows: after outlining related work in Section 2, we present in Section 3 the methodology we propose. Section 4 analyses the

profiles obtained using this methodology on a real dataset corresponding to a one year fraud-stream.

2 Related Work

Data representing credit-card frauds combine two characteristics that both require specific clustering algorithms, namely their data-stream and heterogeneous nature. This section outlines algorithms that have been proposed to deal with either data type.

2.1 Clustering Data-Streams

Data-Stream Characteristics: As opposed to classic data, data-streams are characterised by their production mode: the dataset is incrementally built, which means that not all data are available at once. They usually lead to very large datasets having restrictive characteristics which require specific algorithms to perform data-mining [5].

Indeed, data-streams first demand incremental algorithms that process data progressively, incorporating them as they come in to update the learnt model. Moreover, because of their production mode as well as memory constraints imposed by their quantity, data-streams usually require single-pass algorithms.

Another characteristic of data-streams is their dynamic feature: apart from the data arriving progressively, their underlying distribution generally evolves with time. This is in contradiction with the hypothesis of identically distributed data most classic data-mining algorithms rely on.

General Principles: The main existing methods for clustering such data-streams belong to the framework of incremental clustering. First introduced to address the issue of very large datasets, these algorithms decompose the dataset into samples of manageable size. They consist in iteratively processing each sample individually and merging the corresponding partial results into the final partition. In the case of very large datasets, samples are automatically extracted, e.g. randomly drawn so as to fit in memory. In the case of data-streams, where samples are imposed by the time-line, and defined as the set of data becoming available in a given time interval, samples can be seen as data buffers.

Incremental clustering algorithms can be divided according to the way the partial results are merged, this fusion being either progressive or final. Progressive fusion means including, in the clustering step of a given sample, the results from the previous steps. Final fusion is performed at the end, when all samples have been processed. As detailed below, the same distinction can be applied to data-stream clustering algorithms.

Online Clustering: Online clustering algorithms are incremental approaches with progressive fusion: the previously seen data are summarised by extracted

clusters, possibly weighted by their sizes, and this summary is processed together with the next sample.

This approach has been applied to most classic clustering algorithms, e.g. leading to incremental variants for k-means [6], fuzzy c-means [7], fuzzy c-medoids [8] or DBSCAN [9]. Likewise STREAM [10], which is one of the first algorithms dedicated to data-streams, achieves the same kind of result with memory size limitations and theoretical guarantees on the result quality.

Two-level Approaches: The online approaches are said not to be able to fully adapt to the dynamics of the data, as they do not question previous cluster merges [11]. Therefore, two-level approaches reject progressive fusion and postpone the fusion step until the final partition is required by the user. More precisely, they combine an online part, updating compact representations of the data seen so far, with an offline part, which extracts a final partition from this compact representation. The offline step does not take into account the temporal component: no comparison can be made between two consecutive demands of the user. The two steps are also respectively called micro and macro-clustering.

Some approaches in this framework apply the same clustering algorithm to both steps, e.g. fuzzy c-means [12] or fuzzy c-medoids [8]. The centres or medoids obtained from each data buffer are in turn clustered.

Other methods perform a preclustering step of a different nature than the final one, e.g. incrementally updating quantities to compute cluster statistics such as cluster average or standard deviation. This approach is exemplified by BIRCH [13], that incrementally builds a compact representation of the dataset, based on structured summaries that optimise memory usage along user-specified requirements. CLUSTREAM [11] generalises the representation, taking into account the temporal dimension in the precluster description. These algorithms try to add a new data point to one of the preclusters. If this fails, then a new cluster is created to represent the data point. To maintain the memory size, either one of the previously identified clusters is deleted, e.g. based on a recentness criterion, or two clusters are merged, e.g. the two most similar.

This principle has also been applied to density-based clustering [14]: it also combines online micro-cluster maintenance with offline generation of the final clusters with a variant of DBSCAN. Two types of clusters are distinguished: core and outlier-clusters, which can become core-clusters if they reach a size threshold. To prevent memory overload, the outlier-clusters are periodically pruned.

The proposed methodology described in the Section 3 is similar to this one, with two main differences, as detailed below: first it relies on partitioning and not density-based clustering. Second, it uses a different representation of the so-called outlier clusters.

2.2 Clustering Heterogeneous Data

Heterogeneous data are defined as data described by both numeric and categorical attributes. The presence of categorical attributes rule out the computation

of average values and, therefore, the usage of all mean-centred clustering techniques, in particular the very commonly applied k-means and its variants.

Two main approaches can be distinguished: first, so-called relational methods which rely on the pairwise dissimilarity matrix (e.g. pairwise distances) and not on vector descriptions of the data. This type of approach includes hierarchical clustering methods, density-based methods [15] as well as relational variants of classic algorithms [16, 17].

On the other hand, medoid-based methods [18, 19] constitute variants of the mean-centered methods that do not define the cluster representative as the average of its members, but as its medoid, that is, the data point that minimises the, possibly weighted, distance to cluster members.

2.3 Linearised Fuzzy *c*-Medoids

The linearised fuzzy c-medoids algorithm [19], written *l-fcmed* hereafter, is a scalable medoid-based clustering algorithm: it can process data that are both in vast amounts and heterogeneous. Moreover, being a fuzzy variant, it offers properties of robustness and independence from random initialisation. We present it in greater detail here because the proposed method described in Section 3 depends on it.

The algorithm's inputs are $\mathcal{D} = \{x_i \,|\, i = 1, \ldots, n\}$ the dataset to be clustered, d the metric used to compare data, c the desired number of clusters, m the *fuzzifier* which sets the desired fuzziness and p the size of the neighbourhood in which medoid updates are looked for.

After initialisation of the cluster centres as c data in $\mathcal{D}$ the algorithm alternately updates memberships and cluster centres using the following equations:

$$u_{ir} = \left[\sum_{s=1}^{c} \left(\frac{d(x_i, v_r)}{d(x_i, v_s)} \right)^{\frac{2}{m-1}} \right]^{-1} \qquad v_r = \operatorname*{argmin}_{k \in N_p(v_r)} \sum_{i=1}^{n} u_{ri}^m d(x_k, x_i) \tag{1}$$

where u_{ir} denotes the membership degree of x_i to cluster r, v_r the cluster centres and $N_p(v_r)$ the neighbourhood of centre v_r, which looks to update medoids in their vicinity, alleviating computational costs. The latter is defined as the p data maximising membership to cluster r. Both updates are iterated until medoid positions stabilise.

3 Proposed Methodology

This section describes the methodology we propose to dynamically cluster a heterogeneous stream, as sketched in Algorithm 1. It belongs to the family of two-level approaches, performing a micro-clustering step based on a partitioning approach.

The micro-clusters, i.e. the cluster information which is updated for each new datum, are defined as cluster medoids. As in [13, 11, 14], we test whether a new data point fits an existing cluster. If the test succeeds, no update is performed;

if it fails, instead of creating a new cluster immediately, we add the data point to a buffer $\mathcal{B}$. Cluster creation then only takes place when the buffer reaches a size threshold and is the result of a partitioning algorithm applied to $\mathcal{B}$.

We propose, for this, a variant of the linearised fuzzy c-medoids [19], adding a cluster selection step to increase cluster homogeneity. This step makes it possible to identify atypical data grouped in a set of as yet unassigned data, denoted $\mathcal{U}$: similarly to [14], we distinguish outliers from core-clusters. The main difference here is that the outliers are not clusters of their own but a single set.

The next subsections respectively describe in more detail the test for assigning a point to an existing cluster, called cluster augmentation criterion, and the cluster selection criterion, as well as the global architecture of the algorithm.

3.1 Cluster Augmentation Criterion

Cluster augmentation consists in testing whether an incoming point can be assigned to one of the already identified clusters, or whether it should be buffered as a candidate clusterable. This procedure has two immediate advantages. The first is to reduce, for a given volume of data, the number of times the buffer is filled and, thus, the identification of new clusters, making the global process run faster. The second advantage is that it avoids the discovery, at a later stage, of clusters too similar to those already identified: it, indeed, ensures that the next data considered for clustering are adequately separate from the previously selected clusters. It, therefore, suppresses the need for a posterior fusion step, where the results of the buffer clustering are merged to the previously obtained clusters. This, again, alleviates the computation costs of the global method.

We propose to consider that a data point can be assigned to an existing cluster only under the condition that it does not deteriorate its compactness: addition is not aimed at generalising a cluster, rather at processing new data quickly. Therefore, we impose that a point can be added to a cluster if and only if it falls within the mean distance to the medoid at the time the cluster was selected. This can be written formally as:

$$d(x, \nu_C) \leq \frac{1}{|C|} \sum_{y^* \in C} d(y^*, \nu_C) \tag{2}$$

where y^* denotes any element in the cluster at its creation. This augmentation condition defines a local criterion that adapts to the compactness of each cluster. In particular, for a cluster containing only exact replicas of a data point, the augmentation criterion will exclusively allow the addition of more replicas. Once again, this severe criterion is intended to help the identification of fraudster behaviours, more than it is meant to offer a summarisation of the observed data.

3.2 Cluster Selection Criterion: the *l-fcmed-select* Algorithm

Medoid selection is a substep that actually modifies the *l-fcmed* algorithm, leading to the variant we propose, called *l-fcmed-select*. One of its motivations comes

Algorithm 1 *$\mathcal{DS}$-l-fcmed-select*

For each new data point x

▷ Process x

if $\exists C \in \mathcal{C}$ such that x can be added to C according to Eq. 2 **then**
  Update $\mathcal{C}$: $C \leftarrow C \cup \{x\}$
else
  Update $\mathcal{B} \leftarrow \mathcal{B} \cup \{x\}$
end if

▷ Process $\mathcal{B}$

if $|\mathcal{B}| = \tau_{\mathcal{B}}$ **then**
  Apply *l-fcmed-select* to $\mathcal{B} \rightarrow (\mathcal{C}', \mathcal{U}')$
  Update $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}'$
  Update $\mathcal{U} \leftarrow \mathcal{U} \cup \mathcal{U}'$
end if

▷ Process $\mathcal{U}$

if purging criterion on $\mathcal{U}$ is fulfilled **then**
  Apply *$\mathcal{DS}$-l-fcmed-select* recursively to $\mathcal{U}$, treated as a data-stream
end if

from the issue of determining the appropriate number of clusters, c: as all partitioning clustering algorithms, *l-fcmed* always produces c clusters, whether c is relevant for the considered data or not. In order to bypass this difficulty, we propose to ask for a 'reasonable' –probably overestimated– number of clusters, and then to select only *some* of the produced clusters.

The proposed selection condition is, again, a compactness criterion: we keep only the clusters of sufficient size that exhibit a very high homogeneity, evaluated as the radius of the cluster. The selection criterion can, thus, be formalised as:

$$|C| > \tau_C \qquad \text{and} \qquad \max_{x \in C} d(x, \nu_C) \leq \xi \tag{3}$$

where τ_C is the minimal acceptable size and ξ a compactness threshold.

This algorithm does not return a data partition as some of the data, that assigned to discarded clusters, remain unattributed. More formally, *l-fcmed-select* outputs a set of clusters, $\mathcal{C} = \{C_1, ..., C_{c'}\}$, with $c' \leq c$, and a set of unassigned data $\mathcal{U}$. The latter represents atypical cases, or as yet unexplained data, that do not deserve, for the time being, medoids of their own in the clustering solution.

3.3 Global Architecture and Parameters

The global architecture of the proposed methodology, called *$\mathcal{DS}$-l-fcmed-select*, is given in Algorithm 1. The set of clusters $\mathcal{C}$, the buffer $\mathcal{B}$ and the set of unassigned data $\mathcal{U}$ are initially assigned the empty set.

When a new data point then arrives, the algorithm tries to assign it to one of the existing clusters. If this substep fails, the point is added to the buffer $\mathcal{B}$.

Once the buffer reaches a user-defined size threshold, $\tau_{\mathcal{B}}$, meaning that too many points differ from the previously identified clusters, then *l-fcmed-select* is

applied anew to the data in the buffer. Previously testing addability ensures that all obtained clusters are distinct enough from the already identified clusters.

The *l-fcmed-select* algorithm imposes handling the set $\mathcal{U}$ of unassigned data. The corresponding data are 'more atypical' than the ones in $\mathcal{B}$, insofar as they have been submitted at least once to clustering, whereas buffered data have only been tested against existing clusters. However, it may be the case that atypical behaviours eventually become less isolated, warranting a cluster of their own.

Therefore, when some purging condition on $\mathcal{U}$ is reached, the data it contains are considered once more for core-clustering. We apply $\mathcal{DS}$*-l-fcmed-select*, processing it as a fictitious data-stream. The purging condition could be linked to the $\mathcal{U}$'s size, yet this induces the risk of running back to back purges, if $\mathcal{U}$'s size does not fall far enough below the threshold. To avoid this, we harness $\mathcal{U}$'s purge to the amount of processed data, with the idea that these regular purges still attest to the aging of the data. A purge, thus, starts every time $\tau_{\mathcal{U}}$ points of the data-stream have been processed and stops as soon as $\mathcal{U}$'s size stops decreasing.

Overall, the proposed method relies on three sets of parameters, that is the *l-fcmed* parameters: the number of clusters c, the fuzzifier m and the neighbourhood size for the medoid update p; the cluster selection criteria: the minimal acceptable cluster size τ_C and diameter ξ; and the size thresholds: the size at which $\mathcal{B}$ is subjected to *l-fcmed-select*, $\tau_{\mathcal{B}}$ and the rate at which $\mathcal{U}$ is purged, $\tau_{\mathcal{U}}$.

4 Experimental Results

4.1 Data and Experimental Setup

We applied the proposed methodology to a real fraud-stream covering almost one year (49 weeks) and containing close to a million fraudulent transactions. Each is described by its amount in Euros, a positive real number, as well as categorical attributes, namely the country where it took place and the merchant category code, a general categorisation of transacted products.

The distance d between two transactions t_1 and t_2, represented as vectors of their features, is defined as $d(t_1, t_2) = 1/q \sum_{i=1}^{q} d_i(t_{1i}, t_{2i})$, where d_i is the distance for attribute A_i. Two cases are distinguished: d_i is either $d_i = d_{cat}$, if A_i is categorical, or $d_i = d_{num}$, if it is numeric. Each is defined as follows:

$$d_{cat}(x, y) = \begin{cases} 1 & \text{if } x \neq y \\ 0 & \text{otherwise} \end{cases} \qquad d_{num}(x, y) = \frac{|x - y|}{\max(x, y)}$$

The distance for numeric attributes is thus defined as a relative gap: the assumption being that a difference of 2€ in amount, say, should not have the same impact if the compared amounts are around 5€ or if they are closer to 1 000€.

For the algorithm parameters, we use the following setup: *l-fcmed* is applied with $c = 400$, $m = 2$ and $p = \lfloor \tau_{\mathcal{B}}/c \rfloor = 12$. Cluster selection is based on $\tau_C = 10$ and $\xi = 0.15$. Note that, due to the distance choice, this low value imposes that data assigned to a given cluster all have the same country and the same activity, variability is only (moderately) tolerated for the amount. The size threshold for

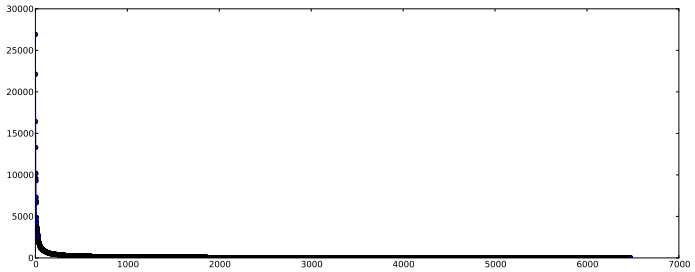

Fig. 1. Sizes of the clusters once all data has been processed, in descending order

the buffer is $\tau_{\mathcal{B}} = 5\,000$ and the set of unassigned data $\mathcal{U}$ is purged every time $\tau_{\mathcal{U}} = 50\,000$ transactions have been processed. The experiments were run using a parallel implementation of $\mathcal{DS}$-*l-fcmed-select* on a multicore cluster.

4.2 Experimental Results

With these parameters, the proposed methodology finds a total of 6 476 clusters covering a wide range of sizes and time-spans. In the following, we first analyse the global results of the final step, then comment the dynamics of fraud profiles, both globally and at a more detailed level.

Final Global Analysis: In order to study the obtained clusters when the whole data-stream has been processed, we focus on cluster sizes. Figure 1 shows the final sizes of the clusters in decreasing order.

It can be observed that, despite a very severe compactness criterion, the largest cluster groups 26 917 frauds, highlighting the redundancy of the data, in which there exists a largely dominant type of fraud. More generally, the 25% largest clusters cover 77.7% of all affected frauds. Conversely, the 25% smallest clusters only cover 3.6% of the assigned frauds. This shows that the fraud behaviours include a large number of rather rare fraud procedures, either due to atypical country, activity, amounts or a combination of the above.

Moreover, at the end of the process there remain 21.7% of the whole as unassigned frauds, i.e. too atypical to be clustered according to the criteria imposed on the clustering result.

Global Dynamic Analysis: We first analyse the dynamics of fraud profiles globally, examining the distribution of each profile time-span. These lengths are defined as the number of weeks between dates for the first and last frauds assigned to each cluster. We should point out that our analysis of the dynamics is more concerned with the evolution of the contents, the frauds, then it is with the displacement of the clusters, as is usually the case.

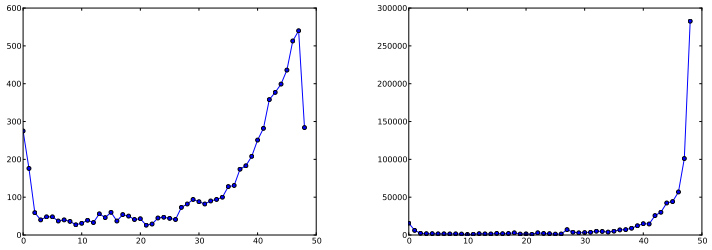

Fig. 2. Time-span histograms: (left) number of clusters covering each number of weeks, (right) cumulative size of clusters covering each number of weeks

Figure 2 shows the repartition of these durations, both in terms of the number of clusters, on the left, as well as the number of frauds they represent, on the right. Two types of clusters co-exist: a first category, representing 46.9% of all clusters, groups long-lasting profiles that nearly span over the whole year, with periods longer than 40 weeks. This comes along with a dominance in terms of data quantities, as they represent 81.6% of the assigned fraudulent transactions.

A second type of profiles can be described as 'flash profiles': these correspond to behaviours that are very precisely dated, disappearing within a single week or, possibly, up to two or three weeks. This category represents 9.1% of the clusters, and still appear to be significant in terms of number of frauds. More precisely, the average size of the 275 clusters observed only during a single week is 57, with sizes ranging from 10 to 3001.

Figure 3 provides another view on the dynamics of credit-card fraud profiles. The full line on top shows the evolution of the number of active clusters each week, where a cluster is called active at a given date if it is created or if at least one transaction is assigned to it at that date. The number of active clusters is globally increasing over the entire period, excepting the final period which is shorter than the other weeks and not augmented by new data. The ratio between new and existing clusters is clarified by the two additional dashed lines, where the one nearer the bottom shows cluster creation per week and the one closer to cluster activity traces the number of augmented clusters in the period. Cluster creations tend to slow down significantly after each peak and the range between consecutive maximum and minimum shrinks as time goes. This is explained by the fact that most relevant profiles are found and that new ones do not appear all the time. Cluster augmentation, for its part, explains most of the activity, as the closeness between plots shows, since the more clusters there are, the more likely a fraud will fit one. It should also be observed that the number of active clusters is always much lower than the total number of clusters.

Detailed Dynamic Analysis In order to offer a more detailed analysis, Figure 4 shows the size evolutions of the 40 largest clusters, where the ranking is based on the clusters' final sizes, as in Figure 1.

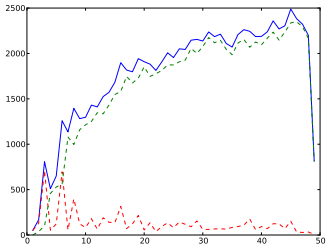

Fig. 3. Number of active (full line), augmented (top dashed) and new (bottom dashed) clusters per week

Four main types are observed, relative to growth speed: the curves can be linear, corresponding to a constant speed, concave, as in a deceleration behaviour, convex, showing an acceleration, or, as already discussed, flash profiles. We discuss each in turn below.

The constant increase profiles, globally exemplified by ten of the eleven largest clusters, represent fraud behaviours that are observed all year with very little variations in terms of relative distribution, i.e. with an approximate constant number of additional representatives each week. These are the 'fraud basics' or classic behaviours fraudsters know they can rely on. It should be pointed out that the constant increase is correlated to the fact that these clusters are the largest ones.

The second profile type, characterised by concave curves, presents a deceleration or even an abrupt halt in exploitation. This type is illustrated by clusters 21, 25 or 27. It corresponds to obsolete types of frauds on their way out. It would be interesting to check with credit-card fraud experts whether these observations can be related to the introduction of specific anti-fraud policies.

The third type is complementary: it groups convex curves, more precisely a parabolic increase in size or even a sudden augmentation, like cluster 13. It would likewise be relevant to discuss these results with experts to see if these observations correlate with credit-card policy modifications.

Over the given temporal window, some clusters combine a sudden increase and progressive deceleration, as clusters 35 and 38 do. These hybrid profiles introduce some doubt as to whether clusters of the third type are going to slow down at some point.

The last type is the already discussed very short-lived clusters , e.g. cluster 23. It is interesting to note that some of these are indeed large enough to join the 40 largest clusters, in which most are represented over a much longer time-span.

5 Conclusion and Future Work

A methodology to process incoming streams of heterogeneous data and study the derived profiles and their dynamics is proposed in this paper, together with

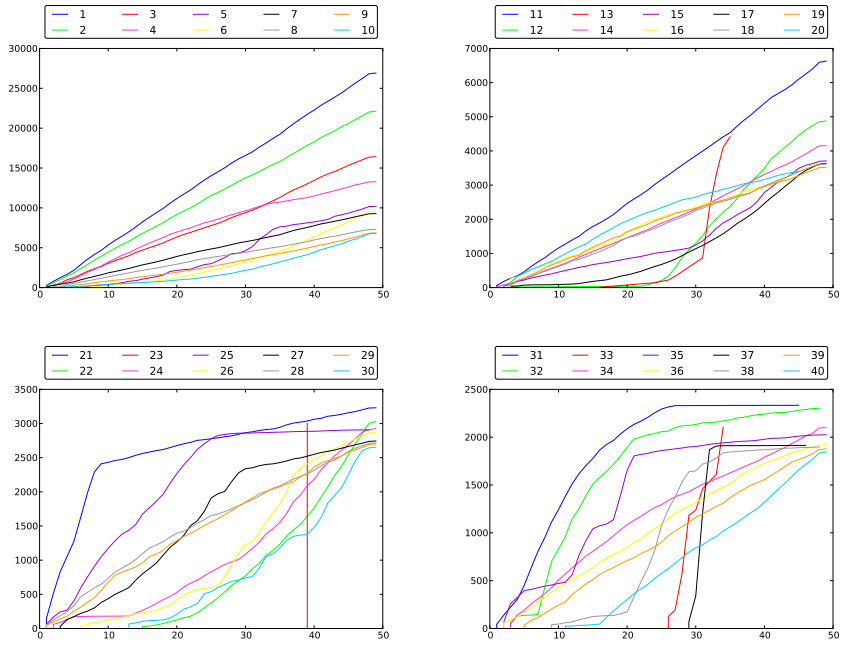

Fig. 4. Evolution of the size of the 40 largest clusters, where the cluster id equals its rank in terms of final size.

its application to a real data set of credit-card frauds. The analysis of the results shows the relevance of the approach, which makes it possible to establish distinct types of fraud profiles depending on their temporal evolution and to identify specific fraudster profiles. The proposed algorithm efficiently processes data streams of heterogeneous nature, in a parallel implementation. It is based on incremental micro-clustering performed by a variant of the *l-fcmed* algorithm, defined as its combination with a cluster selection step guaranteeing highly compact clusters. The processing decomposes the data into three subsets, namely clusters, data to-be-clustered in a buffer and atypical data, remaining unassigned. The purge of the unassigned data, and to a lesser extent the clustering of buffered data, make it possible to manage some data beside the current flow of the data stream. This is particularly helpful in the case of credit-card fraud profiling, where frauds can be declared a long time after the transaction has been recorded.

Ongoing work aims at combining these micro-clustering results with a macro-clustering step, e.g. through a hierarchical method, to get a more synthetic view of the clusters, even if the detailed analysis also provides relevant and meaningful information. Future work includes comparisons with existing algorithms, as well as the discussion with experts in anti-fraud policies to obtain a semantic validation of the observed results. Another perspective is to take more information into account in fraud comparisons, in particular related to transaction sequences.

Acknowledgements This work was supported by the project eFraudBox funded by ANR - CSOSG 2009.

References

1. Banque de France: Annual Report of the Observatory for Payment Card Security. `http://www.banque-france.fr/observatoire/home_gb.html` (2011)
2. Bolton, R.J., Hand, D.J.: Statistical fraud detection: a review. Statistical science **17** (2002) 235–255
3. Phua, C., Lee, V., Smith, K., Gayler, R.: A comprehensive survey of data mining-based fraud detection research. Artificial Intelligence Review (2005)
4. Laleh, N., Azgomi, M.A.: A taxonomy of frauds and fraud detection techniques. Information Systems, Technology and Management Communications in Computer and Information Science **3** (2009) 256–267
5. Aggarwal, C.C., ed.: Data Streams: Models and Algorithms. Springer (2007)
6. Farnstrom, F., Lewis, J., Elkan, C.: Scalability for clustering algorithms revisited. SIGKDD Explorations **2** (2000) 51–57
7. Hore, P., Hall, L., Goldgof, D.: Single pass fuzzy c means. In: Proc. of the IEEE Int. Conf. on Fuzzy Systems, FUZZ-IEEE'07. (2007) 1–7
8. Labroche, N.: New incremental fuzzy c medoids clustering algorithms. In: Proc. of the NAFIPS'10. (2010) 145–150
9. Ester, M., Kriegel, H.P., Sander, J., Wimmer, M., Xu, X.: Incremental clustering for mining in a data warehousing environment. In: Proc. of the 24th Very Large DataBases Conference, VLDB'98. (1998) 323–333
10. O'Callaghan, L., Meyerson, A., Motwani, R., Mishra, N., Guha, S.: Streaming-data algorithms for high-quality clustering. In: Proc. of the 18th Int. Conf. on Data Engineering, ICDE. (2002) 685–694
11. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: Proc. of Very Large Data Bases, VLDB'03. (2003) 81–92
12. Hore, P., Hall, L., Goldgof, D., Cheng, W.: Online fuzzy c means. In: Proc. of NAFIPS'08. (2008) 1–5
13. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: an efficient data clustering method for very large databases. In: Proc. of the ACM Int. Conf on Management of Data, SIGMOD'96, ACM Press (1996) 103–114
14. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: Proc. of the 6th SIAM Int. Conf. on Data Mining. (2006)
15. Sander, J., Ester, M., Kriegel, H.P., Xu, X.: Density-based clustering in spatial databases: the algorithm DBSCAN and its application. Data Mining and Knowledge Discovery **2** (1998) 169–194
16. Hathaway, R., Bezdek, J.: Nerf *c*-means: non euclidean relational fuzzy clustering. Pattern Recognition **27** (1994) 429–437
17. Hathaway, R., Bezdek, J., Davenport, J.: On relational data versions of *c*-means algorithms. Pattern Recognition Letters **17** (1996) 607–612
18. Kaufman, L., Rousseeuw, P.: Finding groups in data, an introduction to cluster analysis. John Wiley & Sons, Brussels, Belgium (1990)
19. Krishnapuram, R., Joshi, A., Nasraoui, O., Yi, L.: Low complexity fuzzy relational clustering algorithms for web mining. IEEE Transactions on Fuzzy Systems **9** (2001) 595–607